A New Approach towards Non-holonomic Path Planning of Car-like Robots using Rapidly Random Tree Fixed Nodes(RRT*FN)

A Thesis Submitted to
The University of Kent
for The Degree of
Doctor of Philosophy
in Electronic Engineering

Sotirios Spanogianopoulos

Nov 2017

Abstract

Autonomous car driving is gaining attention in industry and is also an ongoing research in scientific community. Assuming that the cars moving on the road are all autonomous, this thesis introduces an elegant approach to generate non-holonomic collision-free motion of a car connecting any two poses (configurations) set by the user. Particularly this thesis focusses research on "path-planning" of car-like robots in the presence of static obstacles.

Path planning of car-like robots can be done using RRT and RRT*. Instead of generating the non-holonomic path between two sampled configurations in RRT, our approach finds a *small incremental step* towards the next random configuration. Since the incremental step can be in any direction we use RRT to guide the robot from start configuration to end configuration.

This "easy-to-implement" mechanism provides flexibility for enabling standard planners to solve for non-holonomic robots without much modifications. Thus, strength of such planners for car path planning can be easily realized. This thesis demonstrates this point by applying this mechanism for an effective variant of RRT called as RRT - Fixed Nodes (RRT*FN).

Experiments are conducted by incorporating our mechanism into RRT*FN (termed as RRT*FN-NH) to show the effectiveness and quality of non-holonomic path generated. The experiments are conducted for typical benchmark static environments and the results indicate that RRT*FN-NH is mostly finding the feasible non-holonomic solutions with a fixed number of nodes (satisfying memory requirements) at the cost of increased number of iterations in multiples of 10k.

Thus, this thesis proves the applicability of mechanism for a highly constrained planner like RRT*-FN, where the path needs to be found with a fixed number of nodes. Although, comparing the algorithm (RRT*FN-NH) with other existing planners is not the focus of this thesis there are considerable advantages of the mechanism when applied to a planner. They are a) instantaneous non-holonomoic path generation using the strengths of that particular planner, b) ability to modify

on-the-fly non-holomic paths, and c) simple to integrate with most of the existing planners.

Moreover, applicability of this mechanism using RRT*-FN for non-holomic path generation of a car is shown for a more realistic urban environments that have typical narrow curved roads. The experiments were done for actual road map obtained from google maps and the feasibility of non-holomoic path generation was shown for such environments. The typical number of iterations needed for finding such feasible solutions were also in multiple of 10k. Increasing speed profiles of the car was tested by limiting max speed and acceleration to see the effect on the number of iterations.

Acknowledgements

I would like to express my sincere gratitude to my supervisors Dr. Konstantinos Sirlantzis and Dr. Gareth Howells for their continuous encouragement, keen interest and constant guidance throughout the course of my research work. My conversations with them have been a source of great encouragement, inspiration and learning.

I would also like to acknowledge School of Engineering and Digital Arts support for without their funding I could not have completed this thesis.

I would like to express my gratitude to my parents. Especially to my mother, Ms Paraskevi Spanogianopoulou, who has always been there for me and I am thankful for everything she has done. To my father, Mr Ioannis Spanogianopoulos, who has been a source of encouragement.

This work was part of the SAVEMORE project co-funded by the European Regional Development Fund and the School of Engineering and Digital Arts, University of Kent, UK. SAVEMORE was selected for funding under the Interreg IVA France (Channel) England programme.

To xxx:

Dedication goes here

Contents

Abstract **i**

Acknowledgements **iii**

List of Tables **xi**

List of Figures **xii**

List of Publications **xvi**

1 Introduction **1**

1.1 Tools for autonomous navigation of a robot 2

1.1.1 3D object intersection checking 2

1.1.2 Collision-checking of any robot motion 2

1.1.3 Planners 4

1.2 Necessity of a new approach 5

1.2.1 Can be used by most planners 5

1.2.2 Facilitating global path planning 6

1.2.3 Giving path planners ability to rapidly handle unforeseen obstacles on-the-fly . . . 7

1.2.4 Enabling feasibility for existing computing technology constrained by memory and power . . . 7

1.3 Scope and objectives of this research . . . 8

1.4 Structure of Thesis . . . 9

2 State-of-the-art review **11**

2.1 Review of a sampling based path planner . . . 11

2.1.1 Rapidly-exploring Random Tree (RRT) . . . 12

2.1.2 RRT* . . . 14

2.1.3 Selecting RRT as our test planner . . . 14

2.2 Existing approaches using RRT for non-holonomic systems . . . 15

2.2.1 Car-like robot . . . 16

2.2.2 Other complex non-holonomic systems . . . 16

2.3 Safe path planning . . . 17

2.3.1 Indoor environments . . . 17

2.3.2 Planning on rough terrains . . . 18

2.3.3 Online RRT . . . 19

2.4 Further improvements introduced in RRT . . . 20

2.5 Other factors affecting path planning . . . 22

2.5.1 Sensing unforeseen changes . . . 22

2.5.2 Addressing motion uncertainty in car-like robot 23

2.6 Methods Based on Fuzzy Logic 23

2.6.1 Sensor-based Methods 24

2.6.2 SLAM-based Methods 27

2.7 Latest developments 30

3 Motivation and Contribution **40**

3.1 Need of path planning process in autonomous navigation 40

3.1.1 Simple path planning for any robot 41

3.1.2 Car-like robot – a robot with non-holonomic constraints . . . 41

3.1.3 Framework for autonomous navigation of car-like robot using way points 43

3.2 Revisiting current state-of-art approaches 45

3.3 Technical characteristics of mechanism augmenting planners for non-holonomic path generation 46

3.3.1 Computationally inexpensive 46

3.3.2 Instantaneous 46

3.3.3 Simplicity in integrating with already existing path planners . 47

3.4 Achievements and contribution 47

3.4.1 Interweaving incremental collision checking with generator . . 47

3.4.2 Strengths of most existing planners can now be applied for non-holonomic path planning 48

3.4.3 Could be easily scaled for dynamic environments with existing motion planners 48

3.5 Practical aspects 49

4 Augmented Path Planner: RRT* Fixed Nodes for Non-Holonomic robot (RRT*FN-NH) 50

4.1 A review of RRT*FN 51

4.1.1 Overall structure 51

4.1.2 Sampling of new node 52

4.1.3 Connection of new node to tree 52

4.1.4 Rewiring of nodes 53

4.1.5 Maintaining constraint of maximum fixed nodes 53

4.1.6 Summarising into an algorithm 54

4.1.7 Problems in Extending RRT*FN for non-holonomic systems . 55

4.2 Problem Statement 56

4.3 Non-holonomic path generation using small dt 58

4.3.1 Modelling kinematics of car-like robot 59

4.3.2 Random non-holomic instantaneous path generator 60

4.4 RRT*FN with non-holonomic constraints 61

4.4.1 Simple modifications to RRT*FN 61

4.4.2 Highlights on RRT*-FN-NH 62

5 Experiments and Results **64**

5.1 Specific details on implemented approach 65

5.1.1 Minimising steering angle change 66

5.1.2 Collision checker 66

Benchmark Environments 67

Google Maps 67

5.2 RRT*FN-NH in traditional environments 67

5.2.1 Initial parameters 68

Random seed as a parameter 68

5.2.2 Feasibility of RRT*FN-NH in finding solutions 69

5.2.3 Study of narrow passage example 70

5.2.4 Study with environment requiring multiple manuevers 74

5.3 RRT*FN-NH on Google Maps 76

5.3.1 Car speed profiles 77

Slow speed 78

Medium speed 78

Fast speed 79

5.3.2 Insights 79

6 Conclusion and Future Work **86**

6.1 Achievements 87

6.2 Summary 88

6.3 Possible future advancements of our work 89

6.4 Autonomous robot navigation and obstacle perception 90

6.5 Applications of autonomously-navigating car-like robots 93

List of Tables

5.1 Experimental data of RRT*FN with Nonholonomic constraints 70

5.2 Experimental data of RRT*FN-NH Vs. RRT* for narrow passage environment 72

5.3 Experimental data of RRT*FN-NH Vs. RRT* for car needing multiple manuevers to reach goal 74

5.4 Speed profile of car in pixels, pixels/sec, pixels/sec^2 77

5.5 Results in pixels and sec for slow speed profile 78

5.6 Results in pixels and sec for medium speed profile 79

5.7 Results in pixels and sec for fast speed profile 79

List of Figures

1.1 Illustration of collision-free determination of robot motion 10

2.1 Flowchart of Rapidly-exploring Random Tree (RRT) 13

2.2 Diagram of the overall control system [1] 24

2.3 Block diagram of the overall system [2] 25

2.4 A schematic representation of a car-like robot making a detour from a path towards its primary destination to opportunistically gather additional information about a secondary target (indicated by a blue star) once the presence of the latter has been detected at distance R [3]. 27

2.5 Double integrator system: the snapshots depict the path at different time instant [4]. 27

2.6 Simulated detection of two cars crossing each others. (a) Simulated environment : the robot equipped with a laser range finder detects a car moving from left to right and a second car moving from right to left. (b) Dynamic occupancy grid: red is high, blue is low probability of occupation. The space behind the cars has low probability of occupation. (c) Clustering: different colours characterise objects and occluded or free space [5]. 28

2.7 An example of a generated path [1] 28

2.8 Variables involved in trajectory tracking behaviour, using Bayesian inference [6]. 29

2.9 An example minimum-distance path (bold line) found by non-holonomic RRT* after 1000 vertices [7]. 34

2.10 An example CLiFF-RRT* path generated in the more complex maze scenario [8]. 37

2.11 Theta*-RRT trees in two example environments [9]. 38

4.1 Flowchart of overall structure of RRT*FN. The bold text represents the logical function (fn) stated in RRT*FN algorithm 52

4.2 Flowchart of sampling a new node that can be connected to RRT*FN 53

4.3 Flowchart of connecting a new node to tree 54

4.4 Flowchart of rewiring node of tree to optimise its neighbourhood . . . 55

4.5 Flowchart of various tactics used to maintain fixed number of nodes . 56

4.6 Symbols used in describing the control vector $\mathbf{u}$ and the configuration $\mathbf{q}$ of the car-like robot. 58

4.7 Applying control vector $\mathbf{u_{i-1}}$ to $\mathbf{q_{i-1}}$ results in new pose $\mathbf{q_i}$ of the robot 58

5.1 A path Γ_{best} returned by RRT*FN NH 71

5.2 Path Γ_{best} returned for narrow passage example with Random Seed # 100 72

5.3 Path Γ_{best} returned for narrow passage example with smaller total time and larger total length with Random Seed # 80 73

5.4 Path Γ_{best} returned for Random Seed # 50, where car needed multiple manuevers to reach goal 75

5.5 Path Γ_{best} returned for Random Seed # 30, where car needed multiple manuevers to reach goal 75

5.6 Path Γ_{best} returned for Random Seed # 60, where car needed multiple manuevers to reach goal 76

5.7 Path Γ_{best} returned for Map 1 in slow speed profile of car 82

5.8 Path Γ_{best} returned for Map 2 in slow speed profile of car 83

5.9 Path Γ_{best} returned for Map 1 and Map 2 in medium speed profile of car 84

5.10 Path Γ_{best} returned for Map 1 and Map 2 in medium speed profile of car 85

List of Publications

The research in this thesis have produced the following publications:

Journal:

[J2] **Spanogianopoulos, S.**and Sirlantzis, K. Car-Like Mobile Robot Navigation: A Survey. In: Tsihrintzis, George and Virvou, Maria and Jain, Lakhmi C, eds. *Intelligent Computing Systems: Emerging Application Areas*, Springer-Verlag Berlin Heidelberg, 2016.

Conference:

[C1] **Spanogianopoulos, S.**and Sirlantzis, K. Path Planning of Car-like Robot using RRT*FN. *12th International Conference on Ubiquitous Robots and Ambient Intelligence*, 2015.

Chapter 1

Introduction

During the past few years there has been significant progress in navigation applied to outdoor robots with several industrial applications in well defined environments. At the same time there also exists literature for making autonomous systems reliable in much less structured environments. However, these are mostly not for car-like robots that are non-holonomic in nature.

Driving a car autonomously in real world environments has been recently picked up by many industries and is widely studied in research literature. Currently, its picking up pace in the market to have autonomous cars. The most crucial ability required to enable autonomy is to predict the future *safe* motion of the car under the presence of obstacles moving unpredictably. Basically, the fundamental ability to plan the motion of the car in an intuitive way is the focus of this dissertation.

Once a reasonable representation of the environment is obtained, the vehicle needs to be controlled to follow a certain path. Path execution by the robot has three main stages: navigation, path planning and guidance. The navigation module is usually responsible for the localization of the vehicle within a given map. The path planning module deals with defining global as well as local paths and the guidance module is responsible for keeping the car on the defined path within acceptable errors. The application of such a techniques has many applications in areas such as robotics,

manufacturing, pharmaceutical drug design, computational biology and computer graphics.

In this Chapter, first we discuss the fundamental tools required to enable car to move autonomously, then discuss the necessity of the new approach.

1.1 Tools for autonomous navigation of a robot

There already exists popular and general tools published in literature that are used for facilitating autonomous navigation of a robot.

1.1.1 3D object intersection checking

The objects in 3D physical space are described by the position and orientation (collectively called as *pose*). Representing such objects in frame of reference is widely studied and well known in literature [10, 11]. These objects can either be a robot or a group of robots, or obstacles or any combination thereof.

Checking if two objects (robot with any obstacle) of known geometry are intersecting or not (i.e. in collision or not) is also well studied in literature [12, 13].

1.1.2 Collision-checking of any robot motion

With these fundamental tools of representing objects with poses and intersection checking for known environments the motion of any object (robot) can be computed as follows:

- **Discretisation:** Represent the motion of object as sequence of the object at corresponding poses at some future time.

- **Environment snapshot at a time** Knowing position of object for any time t, and also the position of the obstacles estimated at the same time t, a snapshot of the environment is known. At this step the information of geometries of object and obstacles are known in cartesian space or robot's workspace.

- **Collision check of object:** For every snapshot computed, standard intersection checking between is performed between object and all the obstacles to know if the object will not collide with any obstacle at that time. If no snapshot contains intersection then the entire motion of the object is collision-free.

Figure 1.1 indicates the above process for a four wheeled robot that is moving in a simple straight line motion. Figure 1.1(a) shows how a continuous straight-line path is divided into set of discrete poses at particular future time. Note that the red obstacle is shown at time t_1. To know if the trajectory is collision-free the respective poses of the four wheeled robot needs to be tested with the red obstacle at the respective time intervals; for example, in Figure 1.1(b) the four wheeled robot pose at time t_4 and red obstacle position at time t_4 is considered. Based on standard intersection algorithms the collision is determined at time t_4 between the four wheeled robot and red obstacle. Figure 1.1(d) shows the result of querying the current trajectory for collision-free.

The collision avoidance can be possible by knowing the pose at which collision takes place and finding the right maneuvers based on the intersection information. However, when there are multiple obstacles finding maneuvers can become tedious and hence a common methodology of sampling based approaches are used to find collision-free paths.

Figure 1.1(e) further illustrates that a graph of randomly chosen paths connected with each other may lead in finding a collision free path by using the collision information. Also there is a possibility that all the randomly sampled paths chosen may not have a feasible path to be extracted, especially if environment grows com-

plex. However, as the number of randomly sampled paths increases, so does the probability of finding a collision-free path.

1.1.3 Planners

A motion of an object (robot) can be detected collision free as mentioned in the previous subsection. However, it might happen that the entire motion is not collision free but rather a part of it is. This will be true for many different robot motions. Now, those collision free part of robot motions can be reused to compute a diverted collision-free motion for that object. Making such decisions are a part of algorithms called *planners*.

There are three types of planners: a)Path Planner, b) Reactive Planner and c) Motion planner. All these algorithms accept two standard inputs: i) Start pose and ii) Goal pose of the robot. The obstacles poses and geometries are also needed as inputs but are usually interface to algorithm by a means of *sensor* that provides raw information about them.

Path Planning is a process of finding collision-free motion of robot from start to goal pose, where the obstacles are usually assumed to be static. In this, the major challenge is, if the environment is congested with obstacles, finding the shortest path or time-optimal path becomes a heuristic concern for the planner. Some widely known path planners that need obstacle geometry information are [14,15] and unknown obstacle geometry are [16]. Nowadays, sampling based approaches[17–22] are widely used for representing robot free-space.

Reactive Planning is a local process of avoiding obstacles that are usually in motion. Usually this is combined with path planning process to handle moving obstacles on the fly while it follows the global path as indicated by the path planner. Typical examples of such work done are in [23, 24].

Motion Planning is the process of finding a collision-free motion (trajectory) of robot from start to goal pose taking into account the future predicted motions or perceived motions of obstacles. Thus, conceptually same as path planning but works with an extra parameter the future motions of obstacles. Note that earlier the steps were described to determine collision free motion also applies here (e.g. [25, 26]).

1.2 Necessity of a new approach

Most approaches in literature (for e.g., [27, 28]) for non-holonomic path generation require complex computations that may not generate random path instantaneously. Moreover, path planning process is avoided [29] in autonomous navigation by replacing it with reactive techniques.

Path planning offers several advantages:

1. Formulating a set of multiple non-holonomic paths that can be chosen on the fly based on situation
2. Can have different path quality such as smoothness, less jerks, constrained speed and acceleration.
3. Can be re-planned on existing set of non-holonomic paths already precomputed

Next we reason strongly the necessity of a new approach.

1.2.1 Can be used by most planners

Most sampling based planners require the kinematics of robot in coming up with non-holonomic motion that connects between two nearby sampled poses. Using complex non-holonomic generators can not only slow down the path planning process but

also may not be able to simply find a feasible intuitive trajectory in constrained time.

Instead if an approach exists that relaxes on one of the sampled configuration from two as proposed by path planner but finds a nearby reachable pose, not only, it will be instantaneous, but also, it will satisfy non-holonomic constraints.

Relaxing assumption can be incorporated by many planners and this thesis uses a variant of RRT to show the elegant way of generating non-holonmic motion of car-like robot in combination with planning process.

1.2.2 Facilitating global path planning

Global path planning is a process where the robot moves from start to goal configuration knowing all possible obstacles future motions that it will encounter. However, mostly global path planning is difficult to achieve when a car-like robot moves in unforeseen environment. Instead local planning using way-points as shown in Chapter 1 can be used to handle unforeseen changes. However, there have been reports [30] that autonomous car driving failures occur due to the mistake of other car driven manually.

If all cars driving on the road are autonomous (no manual driving), then a central planner can exists that can collectively pre-plan the path of all the cars. This will need the ability of a global path planner. Moreover, as moving obstacles can most likely be cars on highways or roads, often stopping or passing by other cars in reactive manner may seem to be dangerous. So a global planner is required that plans collectively for all cars so that the possibility of unforeseen obstacles decreases and can be known beforehand.

The global path planning can be done for every car in serial or parallel fashion with the simple assumption that all cars are autonomous.

1.2.3 Giving path planners ability to rapidly handle unforeseen obstacles on-the-fly

Although all car driving may be autonomous there can be pedestrians which could contribute to unforeseen changes that the car-like robot may need to handle. If the non-holonomic path generation is instantaneous the different non-holonomic motions may be generated and added/concatenated to the set of current non-holonomic pre-planned paths by global planning process. Fast collision checking can be done and recomputed non-holonomic connected path can be used on-the-fly to avoid unforeseen obstacles.

1.2.4 Enabling feasibility for existing computing technology constrained by memory and power

There exists planners [31] that limit the memory usage of planners making it easy to deploy on embedded systems that are low powered thus having a longer life with battery. However, most of these planners have not been shown to work with non-holonomic constraints.

While there exists efficient path planners for an unconstrained robot, those approaches can not be used for car-like robot because computing non-holonomic constraints is complicated and may be infeasible on a low end embedded system. However, if the non-holonomic path generation requires less computing power then this approaches can be easily used.

1.3 Scope and objectives of this research

The primary goal of this research is to facilitate planning of car-like robots autonomously. Since a car-like robot is non-holonomic, the scope of this research is focused on tackling non-holonomic constraints posed by car-like robots, which have kinematics described by differential equations. Such robots in physical space can be any non-holonomic vehicles such as cars, jeeps, mini-trucks, etc. Heavy-duty trucks with wide load can also be considered, provided the kinematics of attached wheeled carrier with load is formulated. These vehicles have in common that they have a throttle for speed control and a steering wheel to control their direction of motion.

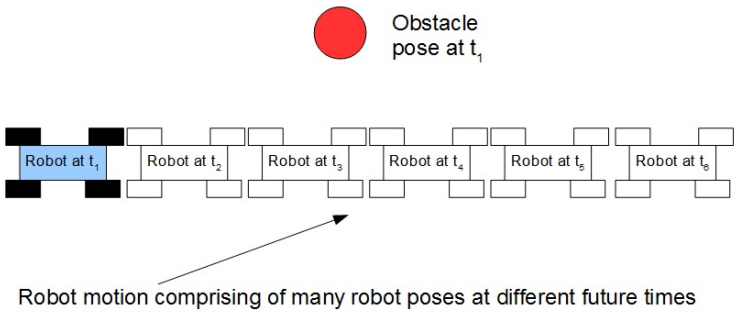
This research is tested heavily for static environments where obstacles can be large or small, collectively forming a narrow passage or complex environment for the car to manuever. Further, to test the quality of the path, real road-like narrow free space is obtained from google maps onto which the non-holonomic motion of the car is found. The current scope was limited for static environments in 2D simulation environment. Live and raw sensor data is not taken into account for collision checking purpose which may pose a challenge in establishing real-time constraints. Although the real-time constraint is usually dependent on non-holonomic path generation computation and collision checking, this research has aimed to minimise the use of collision checking by using incremental planning strategy and making the non-holonomic path generator instantaneous.

The main objective of this thesis is to facilitate non-holonomic motion planning of car-like robots using elegant and well-established planners developed mostly for holonomic robots. As an example, there exist RRT*-Fixed Nodes [31] approach for holonomic robots that tries to find a feasible path under constant memory constraints. Such planner can be very useful while deploying for mobile robots where the computing platforms are limited with constant memory and power source.

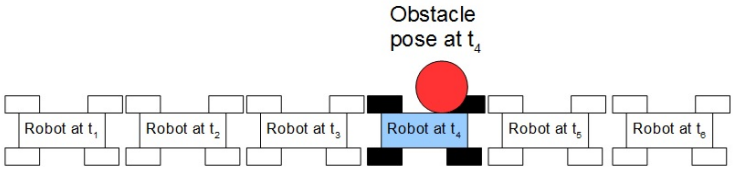
Other objective of this thesis is mainly focused on enabling RRT*-Fixed Nodes (this variant of RRT) for non-holonomic robots. Same strategies mentioned for planning under constant memory constraints is applied and tested for a car-like robot. While original algorithm RRT*-FN can not be tested for car-like robots, a comparison study is done by building RRT* (called RRT*-Non Holonomic) for non-holonomic robots with the new introduced algorithm (RRT*-Fixed Nodes-Non Holonomic) based on RRT*-Fixed Nodes. So RRT*-Non Holonomic is unbounded by memory requirement whereas RRT*-Fixed Nodes-Non Holonmic algorithm had constant memory constraint. The test parameters path length, total time of path and computing time (in terms of iterations) are reported in this thesis for benchmark environments, traditionally used for path planning of holonomic robots. Also to rapidly handle unforeseen changes the non-holonomic path generator needs to be instantaneous is another critical objective that this research tries to focus on.

1.4 Structure of Thesis

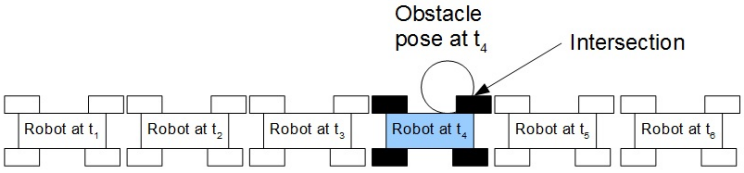
Chapter 2 surveys the path planning literature available for holonomic robots, non-holonomic robots and current challenges faced to achieve safe motion in dynamic environment with unforeseen changes. It mostly concerned with RRT based variant planners as they facilitate incremental step based path/motion planning. Chapter 3 discusses the motivation and contributions made by this approach. It reasons out a need of new approach and provides necessary requirements on achieving the task of autonomous car driving using state-of-art path planners. Chapter 4 discusses on enabling the state-of-the-art planner called RRT*-Fixed Nodes for non-holonomic car-like robots via the approaches suggested in literature. Chapter 5 shows experiments and results on the path finding feasibility under memory constraints and number of iterations available. Chapter 6 concludes the research and provides future insights on further work.



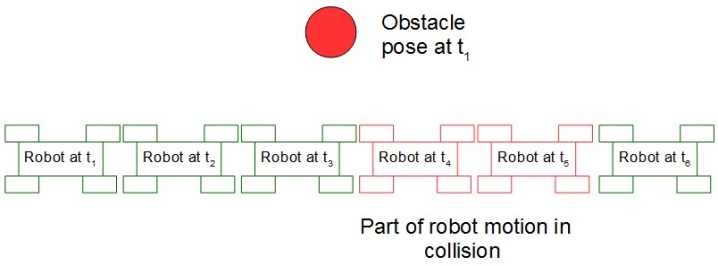
(a) Discretization of robot motion



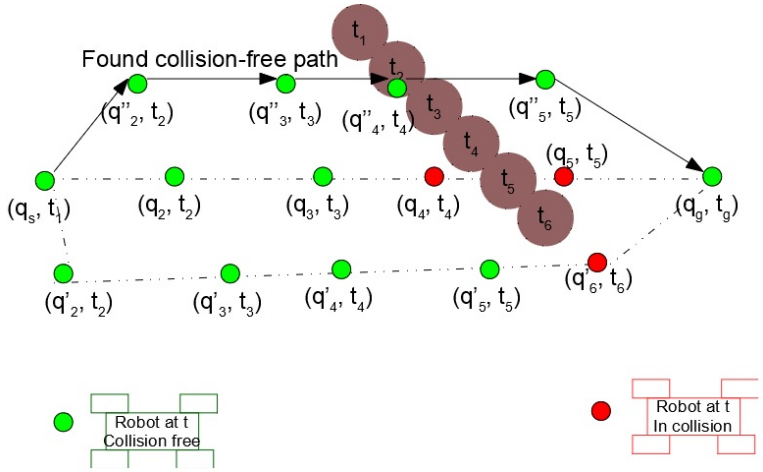
(b) Environment snapshot at future time t_4



(c) Intersection checking determines this configuration will be in collision



(d) Final result with green poses indicating collision-free part motion



(e) Finding a feasible collision free motion to the goal by randomized sampling

Figure 1.1: Illustration of collision-free determination of robot motion

Chapter 2

State-of-the-art review

Since this thesis focuses on path planning aspect involving RRT for autonomous navigation of car-like robot, this Chapter first covers existing literature similar to approaches that use RRT for non-holonomic system planning. Later, also the practical aspects of autonomous navigation of car-like robot addressed in literature are briefly covered to provide an introductory view.

2.1 Review of a sampling based path planner

There exists many literature [16, 32–38] that are fundamentally based on sampling based planners mainly Probabilistic Roadmap Method (PRM) [17] and Rapidly-exploring Random Tree (RRT) [19].

A very popular and successful family of navigation algorithms is based on the Rapidly-exploring Random Tree (RRT) path planning method. An interesting feature of this algorithm for path planning is that a path planned with RRT does not need local planner to find a way from a configuration to another. It also allows the RRT to rapidly explore in the beginning, and then converge to a uniform coverage of the space.

2.1.1 Rapidly-exploring Random Tree (RRT)

RRT tries to address the motion planning of any kind of robot by sampling in the planning or configuration space of the robot. *Sampling* means given a state of robot, it tries to vary all the parameters within the state to find different collision-free positions and orientations of a robot

Some notions used in describing the complexity of space planning of the robot :

- d : degrees of freedom of robot (number of independent joint variables)
- Q_d: the configuration space of robot, i.e., d-dimensional space with each axis = an independent joint variable
- $\mathbf{Q_{obs}}$: the region in Q_d such that a point on $\mathbf{Q_{obs}}$ representing the state of the robot is in collision with an obstacle in the environment
- $\mathbf{Q_{free}}$: is the remaining collision free region in Q_d
- τ: the tree with root node as q_{init} connected by more nodes. Thus, $\tau = (V, E)$ is a specialized graph with vertices V lying in $\mathbf{Q_{free}}$ and E is a subset of $V \times V$. The branches of the tree are termed as continuous path segments.

Figure 2.1 shows the flowchart of the algorithm. It consists of following functions:

- **InitialiseTree**: Initialises the tree with the root as current robot pose (state) or starting robot pose.
- **Sample**: This function randomly samples the configuration space Q_d and comes with q_{rand}.
- **Nearest**: Finds the nearest robot pose q_{near} to q_{rand} to add to tree τ
- **Steer**: Usually concatenation of a set of one-step hyper straight-line paths connecting the q_{near} to q_{rand} as a single path Γ

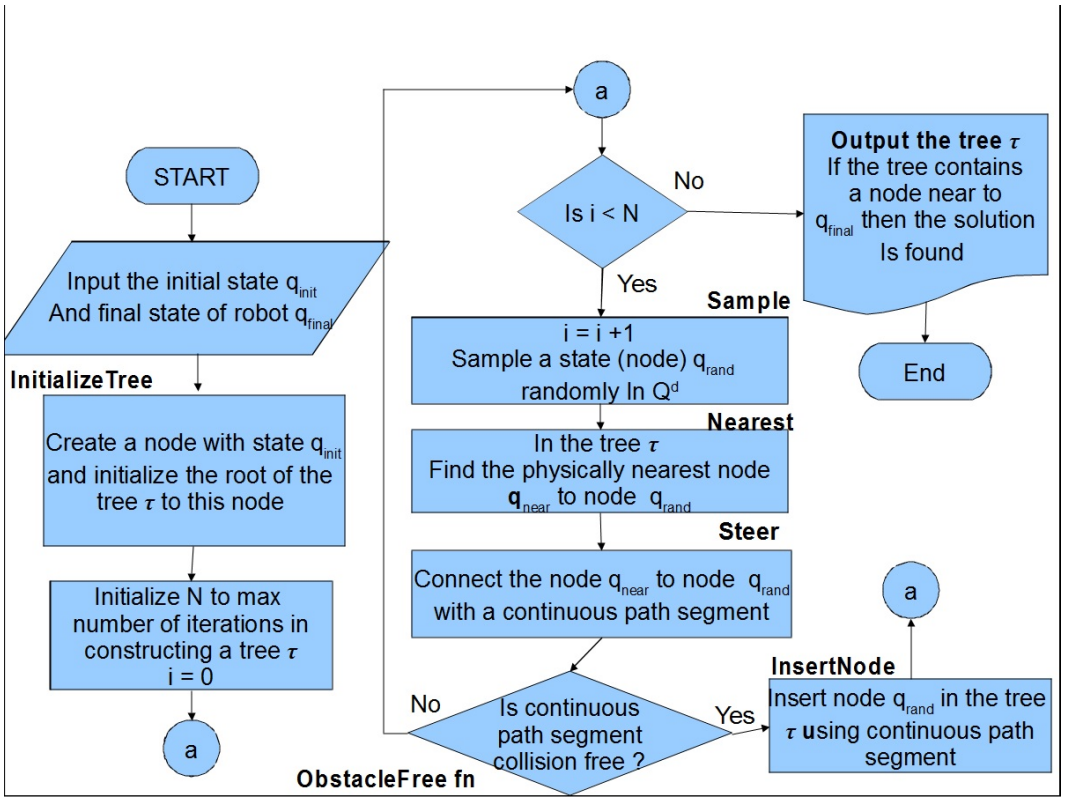


Figure 2.1: Flowchart of Rapidly-exploring Random Tree (RRT)

- **ObstacleFree**: Check if that concatenated single-step hyper straight-line paths lies in $\mathbf{Q_{free}}$
- **InsertNode**: Once the Γ is collision-free add that q_{rand} node to the tree.

Above steps are repeated for N iterations and at most N new nodes can be added to the tree such that the entire tree lies in $\mathbf{Q_{free}}$.

Some of the advantages of RRT include collision checking as and when required in an incremental fashion, can be applied to high Degree Of Freedom robots, and is probabilistically complete. Some disadvantages include lack of exploration in unconnected regions, may not effectively work for simple environments, does not have weighted heuristics in selecting optimized paths.

2.1.2 RRT*

RRT* is a variant of RRT that accommodates heuristics in planning process, Given a corresponding cost function c, optimal path planning seeks to find a feasible path ρ_{best} from initial state to final state, such that,

$$c(\rho_{best}) = minc(\rho) : \rho is feasible \tag{2.1}$$

RRT* introduces two more concepts: a) local neighbourhood of newly added node and b) rewiring of local neighbourhood of newly added node such that cost to initial state is decreased. The tactic used in rewiring of branches of trees reflects the performance of RRT*. More details about the implementation can be found in [39].

2.1.3 Selecting RRT as our test planner

There exists many branches in path planning algorithms, such as, A* [40], PRM [16], and RRT [19]. While A* algorithm is dedicated to find shortest path by overestimating heuristics and using the environment as a grid like structure, it does not support path planning of robots with kinematic constraints directly. Also, making grid like structure for the environment is not ideal as the car might stop in between two grids (each grid usually represents a car pose).

While PRM might be more effective in supporting continuous space exploration with kinematics of the robot, it faces difficulty while scaling up for the dynamic environments. Since these planners have some indirect limitations, the RRT was chosen as our test planner.

The RRT is popular in sampling based algorithms for path planning of robots. It also has a vast variants to effectively explore the configuration space. The primary reasons behind choosing RRT are:

- An incremental planner has more adaptability towards dynamic scenarios of the environment (e.g. [41, 42]). Since car needs to navigate in dynamic environments this planner fits well.

- Simple iteration based probabilistic algorithm allows to study how this approach can find feasible collision-free paths in reasonable time [43]. Thus, the quality of path increases as the iterations increases. This suits our car driving scenario where as the car-like robot is moving the path can still be improved by increasing the number of iterations.

- RRT has an elegant and a simple way of implementation provided the next configuration (pose) of the robot can be computed using its kinematics. Thus, deployment of it on embedded system of car becomes simple and feasible.

- RRT very commonly used for path planning is used mostly for holonomic robots [44]. To extends its capabilities for non-holonomic robots is an intriguing choice to extend this path planner for car-like robots.

2.2 Existing approaches using RRT for non-holonomic systems

Car-like mobile robot navigation has been a challenging field in the academic research over the last few decades. As these robots are mainly aimed for outdoor activities, corresponding navigation algorithms should be able to account for the constraints imposed by the non-holonomic type of movement allowable for car-like mobile robots.

2.2.1 Car-like robot

In [45], a five degrees of freedom dynamic car model is used that considers skidding and sliding. After exploring the configuration space using RRT, the RRT then converges to a uniform coverage of the configuration space by breaking the large Voronoi areas.

In [46] authors present a new method for improving the trajectories based on the Voronoi Fast Marching Method (VFM). The proposed method is suitable for improving the smoothness. Further in [47] authors present the application of Voronoi Fast Marching (VFM) to non-holonomic mobile robot path planning.

Authors in [48] presented several enhancements that improve the quality of the generated path in comparison with the simple adaptation of the Single-query, Bi-directional, Lazy roadmap (SBL) algorithm [48], which successfully builds upon the traditional Probabilistic Roadmaps (PRM), solving also the planning problem in the context of non-holonomic constraints of car-like robots.

2.2.2 Other complex non-holonomic systems

In [27] authors extend the RRT algorithm to handle a large class of non-holonomic dynamical systems. While addressing computational complexity and asymptotic optimality of the system motion, the approach seeks connections within bounding boxes and computes the shape and orientation of these boxes for a large class of dynamical systems based on differential geometry using the ball-box theorem, where shapes can be constructed and joined in a “fuzzy” manner, i.e. without definitive boundary constraints.

Further, in [28] the authors tries to solve the problem of computing a complete motion to the goal within a limited time. There are also other notable approaches

(for e.g. [49]) that do path planning in context of non-holonomic constraints.

Addressing the narrow passage problem[1] while path-planning is crucial for RRT to support for non-holonomic constraints. As an example, in [50], the obstacle vector information is used to grow the tree in some nine possible ways in difficult areas of configuration space (C-space). A modification to a greedy algorithm was made for calculating the planner path, such that it would take as big a step length as possible, as long as it is less than some maximum step length specified.

2.3 Safe path planning

During the past few years there has been significant progress in navigation applied to outdoor robots with several industrial applications in well defined environments. At the same time there is still a need of fundamental breakthroughs in autonomous systems to make them reliable in much less structured environments.

Safe path planning is a key concern in robotics especially when humans are involved. There exists literature to address this concern and is discussed in this section.

2.3.1 Indoor environments

Authors in [45] are using a five degrees of freedom dynamic car model to better place car configuration, considering skidding and sliding. They use the Rapidly-exploring Random Tree (RRT) planner to quickly explore the whole configuration space.

In [51] authors address the problem of safe path planning for their car-like robot that involves non-holonomic constraints and their planner uses an ideal indoor 2D world map, where obstacles are represented by polygonal lines. Then, they used the

[1] A narrow passage problem is visualised as a corridor like environment where the width of mobile robot is slighltly smaller than the distance between corridor walls.

Rapidly-exploring Random Trees (RRT) method, which is an incremental method to quickly explore the whole configuration space, in order to visit the unexplored parts of the space by breaking the large Voronoi areas.

In [52] authors try to address the problem of testing complex reactive control systems and validating the effectiveness of multi-agent controllers. More specifically, they consider the application of the Rapidly-exploring Random Tree (RRT) algorithm to the testing and validation problem and they propose three modifications in order to improve its results.

2.3.2 Planning on rough terrains

As the car-like robots are used outdoors, in [53] a novel RRT algorithm on rough terrains (RRT- RT) has been developed using the Roughness based Navigation Function (RbNF), which is a numerical function that provides the cost-to-go values for each terrain location. Simulation results show that the RRT-RT planner explores the terrain in an efficient manner, and generates final paths that slightly deviate from paths obtained by Dijkstra's algorithm.

In [54] authors propose a new method for navigating a car-like vehicle within an unstructured environment and solving the following problems: precise parking maneuvers, narrow turns and long distance navigation. Using a path planning technique an implicit graph is expanded on the fly by an A* search algorithm. Also adding a feed forward term makes the controller react more quickly and accurately, since reaction of the vehicle to steering input is modelled separately from controller offset introduced by noise. The configuration space obstacles are also computed from an obstacle map acquired from a high definition laser range scanner and search is restricted to the collision free subset of the configuration space.

The paper in [55] presents a path planning algorithm for handling systems with constraints on controls or the need for relatively straight paths for real-time actions.

The initial phase of the algorithm finds an efficient path using guided Expansive Spaces Trees (guided ESTs) and focuses on a randomized search on the low cost region while expanding a tree.

2.3.3 Online RRT

Literature [56] widely addresses offline based path planning, where the collision-free trajectory of the robot is planned offline and then given to the actual robot to execute the trajectory. Offline path planning works if the obstacle geometry and motions are known beforehand. However, usually a car driving on road autonomously may not have known information beforehand about obstacles or environment.

The work presented in [57] proposes on-line use of the RRT on robotic vehicles. By providing a path and a speed command to the controller for tracking the path, the vehicle is able to avoid obstacles and stay in lane boundaries under the different conditions of urban driving. The proposed RRT obtains the dynamically feasible trajectory by running a forward simulation of the closed-loop system, consisting of the vehicle model and the controller.

In [58] authors used a Partial Motion Planning (PMP) approach in order for the algorithm to operate in changing environments under minimal time constraints. Inevitable Collision State (ICS) [59] property was used, which firstly proves that a trajectory is continuously safe while the states safety is verified discretely only, and secondly, it permits a practical computation of safe trajectories by integrating a dynamic collision detection module within the Rapidly-Exploring Random Tree (RRT).

In [60] a method is presented for sensor-based exploration of unknown environments, which proceeds by building a data structure called SRT (Sensor-based Random Tree). The SRT structure represents a roadmap of the explored area with an associated safe region, and estimates the free space as perceived by the robot during

the exploration. The technique which is used for this case is called SRT-Radial and deals with non-holonomic constraints using two alternative planners.

The paper in [61] describes a navigation algorithm for dynamic and uncertain environment. Assuming that moving obstacles are supposed to move in typical patterns that can be pre-learned, in this literature the planner (based on an extension of RRT) takes into account the likelihood of the obstacles trajectory and the probability of collision.

In [62] numerous extensions made to the standard RRT algorithm that enable the on-line use of RRT on robotic vehicles. The sampling is done using the environmental structure to reduce the time in finding trajectories with various maneuvers.

2.4 Further improvements introduced in RRT

Making RRT efficient has also been a prime goal of researchers. In [63], the obstacles in the configuration space are taken into account and a general framework for minimizing the effect of inappropriate sampling is developed based on the visibility region of the nodes in the tree.

Also, in [64] a new sampling scheme is developed for a variant of the dynamic-domain RRT that iteratively adapts the sampling domain for the Voronoi region of each node during the search process. The boundary domain of a given node (i.e. its associated radius) is adapted as a function of the number of expansion attempts and failures from that node. Also, to ensure the probabilistic completeness of the algorithm, a lower bound on the possible radius values of the nodes is formulated when that nodes are extended.

In [65], a variant of the Rapidly-Exploring Random Tree (RRT) path planning algorithm is presented, which is able to explore narrow passages or difficult areas more effectively. More specifically, authors in [65] used some obstacle hints for directions

to grow the tree for path planning in order to find difficult areas of configuration space (C-space). They presented nine possible ways to expand a tree, in which the orientations to grow are either the same as the source configuration or random orientations.

The paper in [66] presents a utility-guided algorithm for the online adaptation of the random tree expansion strategy. As the dimensionality of the configuration space increases, the performance of the tree-based planners that use uniform expansion degrades. The proposed algorithm is based mainly on RRT and guides the expansion towards regions of maximum utility based on local characteristics of state space.

In [67] authors analyze the weaknesses of RRT as the obstacles in the configuration space are not taken into account and/or the sampling region is inappropriately chosen. They propose a general framework for minimizing these weaknesses by considering the visibility region of the nodes in the tree. By developing a simple new planner that defines a boundary domain for a boundary point as the intersection of the Voronoi region of that point and an n-dimensional sphere centered at that point.

Authors in [68] analyze the influence of a parameter introduced in [67], which relies on a new sampling scheme that improves the performance of the RRT approach, and propose a new variant of the dynamic-domain RRT, which iteratively adapts the sampling domain for the Voronoi region of each node during the search process. In particular, authors propose to adapt the boundary domain of a given node (i.e. its associated radius) as a function of the number of expansion attempts and failures from this node.

In [69] a new method called Transition-based RRT (T-RRT) for path planning in continuous cost spaces is presented, which combines the exploration strength of the RRT algorithm that rapidly grow random trees toward unexplored regions of the space, with the efficiency of stochastic optimization methods that use transition

tests to accept or to reject a new potential state.

In [70] authors address the problem of parallelizing the Rapidly-exploring Random Tree (RRT) algorithm on large-scale distributed-memory architectures, using the message passing interface. The parallelization schemes they compared are the OR parallel RRT, the distributed RRT, and the manager–worker RRT.

2.5 Other factors affecting path planning

In literature the other relevant parameters that affect path planning process are discussed here.

2.5.1 Sensing unforeseen changes

Most sensors generate very rudimentary data about obstacles that can not be directly used to get geometry information or their future motions.

There exists approaches [71, 72]), that digitise obstacles using laser scanner, stereo-vision, camera, etc. There exists literature to identify specific types of objects with different complexities, such as, human [73,74], vegetation [75], objects that can cause a mobile robot to drop off [76], etc. Using machine learning techniques [77–79], there exists notably [80, 81], that can be used to detect various kinds of objects.

For detecting future obstacle motion, there exists short term prediction mechanisms [82, 83] commonly used for reactive planning [84, 85] . Also long term prediction of future obstacle motion can be computed using various approaches [86–89] by tracking obstacle motion [82, 86, 90–94].

2.5.2 Addressing motion uncertainty in car-like robot

A car-like robot may not exactly follow the planned path due to slipping of wheels, tyres alignment, motors actuation are never same even with same inputs, etc. However, in literature there already exists appraoches that handle different kind of uncertainties resulting from different factors, such as, slipping of wheels [95], stabilizing a non-holonomic system [96], etc. There also exists a literature [97] that mathematically solves the problem of uncertainty using equations specific to robot.

Using known environment features the uncertainty in robot motion could be reduces is well studied in [98] to generate a map. If there is uncertainty in map generation process there exists approaches (e.g., [99]) to deal with uncertainity in a map. In [100] introduces the notion of a robust path that guarantees a non-holonomic mobile robot to reach its goal based on landmarks.

Some researchers addressed uncertainties in path planning algorithm, e.g., [101–103].

2.6 Methods Based on Fuzzy Logic

In [1] a navigation scheme that contains complex pattern, non-uniform illumination, and strong reflection based on a distributed active-vision network-space system (DAVNSS), is presented. This system is subject to three fuzzy variable-structure decentralized controls (FVSDCs), which includes trajectory tracking and obstacle avoidance. Two distributed wireless charge-coupled-device (CCD) cameras individually driven by two stepping motors are constructed to capture the dynamic pose of the car-like wheeled robot (CLWR) and the obstacle. The control system includes quad processors with multiple sampling rates, while a personal computer (PC) is employed to receive the image of the CLWR or obstacle by a wireless transmitter

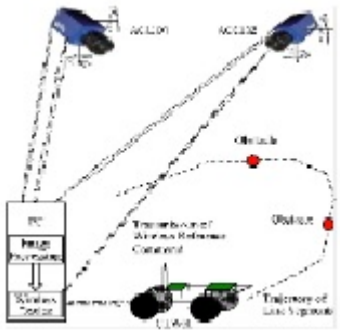


Figure 2.2: Diagram of the overall control system [1]

and then to plan three reference commands for the CLWR and the cameras. Next, a six-step image-processing routine and the calibration between the world coordinate and the image plane coordinate using multilayer perceptrons (MLPs) are established, while in the final step the radial distortion of ACCD is reduced for better localization and tracking.

In [2] a different approach is presented for a navigation system, which includes a path tracking and an obstacle avoidance apparatus for a car-like wheeled robot (CLWR) within an Internet-based smart-space (IBSS) using fuzzy-neural adaptive control (FNAC). This method relies on two distributed charge-coupled device (CCD) cameras, which capture both the dynamic pose of the CLWR and the obstacle. Based on the control authority of these two CCD cameras, a suitable reference command has been planeed, which contains the desired steering angle and angular velocity for the FNAC built into the client computer. The FNAC method that the authors presented in [24] contains also a neural network consisting of a radial basis function (RBFNN) to learn the time-related uncertainties due to the fuzzy-model error, which stem from wireless network delays and CLW slippage.

2.6.1 Sensor-based Methods

Another approach presented in [104] investigates the use of Dynamic Window Approach (DWA) to solve the high speed autonomous navigation problem for mobile

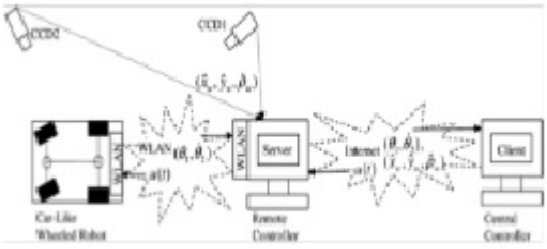


Figure 2.3: Block diagram of the overall system [2]

robots in unknown and unstructured environments. Since the DWA algorithm considers periodically a short time interval when computing the next motion command and based on the fact that the obstacles in the closer environment of the robot impose restriction on the translational and rotational velocities, authors define a Dynamic Window (DW) in order to limit the accelerations executable by the motors. In addition, in order to reduce the time of the motion command selection, they use the sensory data from the environment directly in the obstacle avoidance process without the grid cells building in the velocity space, while in order to determine the Distance To Collision (DTC), they have adopted an analytic solution for polygonal robot. Regarding the experimental results of the algorithm, the obstacle avoidance tests using the extended DWA for different environments (simple and cluttered) at high speeds indicated a good performance and efficiency.

Another approach which relies on a sensor based algorithm for car-like robot based on GVG theory is presented in [105]. For generating the completed GVG, the car-like robot goes through each edge and vertex of GVG in two tangent directions.. In addition, the authors proposed backward motion for direction changes at boundary points, also with favorable results (no collision) in unknown environments. In [106] a new algorithm is presented that enables a car-like robot to explore an unknown planar workspace, based on Generalized Voronoi Graph (GVG) theory. More specifically, since GVG is a set of points in the plane equidistant to two obstacles, the robot of the proposed system has three degrees of freedom and hence the authors defined a rod-GVG edge as the set of the points equidistant to three obstacles. In

[107], an intelligent scaled car-like mobile robot that possesses the capability of autonomous driving in an extra-road environment and fully autonomous parking on standard parking lots is presented. In particular, authors describe a low weight and low cost complex mobile robot that is able to navigate across a previously unknown terrain combining some mechanical, sensorial, computing and communication modules (rather than implementing a new sophisticated algorithm). An algorithm associated with the autopilot of the system was also implemented, in order to make the mobile robot completely autonomous; many of these functions are written in MATLAB, and therefore available for analysis and modification using open-source modules (xPC toolbox). The robot described in [3] is equipped with a sensor that can alert it if an anomaly appears within some range while the robot is moving. In that case, the robot tries to deviate from its computed path and gather more information about the target without incurring considerable delays in fulfilling its primary mission, which is to move to its final destination. The originality of this approach is to take a "semi-corrective" action, i.e. deviating while attempting to further define the problem, akin to a car stepping out of its lane when flashing lights appear ahead – not changing lanes yet, just gaining a view of the obstacle. This model relies on a sampling-based planner called SYCLOP, which works by automatically defining a decomposition of the workspace, creating an adjacency and abstraction graph, and searching that graph for a high-level guide. Then, a low-level planning layer computes the actual dynamically feasible paths and informs the upper layer for how to assign informative weights to the edges of the abstraction graph.

A different approach is presented in [4], where authors present a new trajectory deformation scheme in order to improve path deformation. During the course of execution, the still-to- be-executed part of the motion is continuously deformed in response to sensor information (internal and external) acquired on-line, thus accounting for the incompleteness and inaccuracies of the a priori world model.

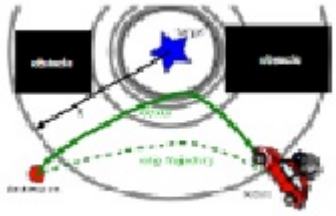


Figure 2.4: A schematic representation of a car-like robot making a detour from a path towards its primary destination to opportunistically gather additional information about a secondary target (indicated by a blue star) once the presence of the latter has been detected at distance R [3].

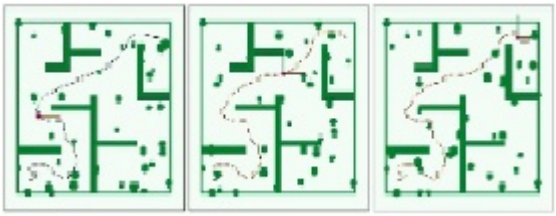


Figure 2.5: Double integrator system: the snapshots depict the path at different time instant [4].

In [5], the Probabilistic Velocity Obstacle (PVO) provides a probabilistic estimation of the occupied free space around the robot and of the velocity with which the objects are moving. The observations of the mobile robot update a 4D probabilistic occupancy grid (incl. space and velocity), and the probability of collision in time is estimated for each reachable velocity of the robot. The proposed system shows that is able to take directly into account limited range and occlusions, uncertain estimations of velocity and position of the obstacles, allowing the robot to navigate safely toward the goal.

2.6.2 SLAM-based Methods

In [108] the authors address the problem of on-line path following for a car working in unstructured outdoor environments. More specifically, the partially known map of the environment is updated and expanded in real time by a Simultaneous Localization and Mapping (SLAM) algorithm. This information is used to imple-

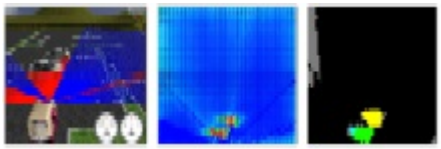


Figure 2.6: Simulated detection of two cars crossing each others. (a) Simulated environment : the robot equipped with a laser range finder detects a car moving from left to right and a second car moving from right to left. (b) Dynamic occupancy grid: red is high, blue is low probability of occupation. The space behind the cars has low probability of occupation. (c) Clustering: different colours characterise objects and occluded or free space [5].

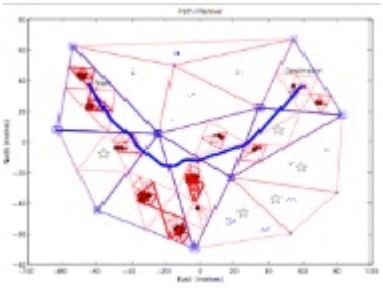


Figure 2.7: An example of a generated path [1]

ment global path planning based on a new method which constructs a cost graph using the D* search algorithm. In this stage, uncertainty is incorporated in the cost function, and since the continuity of the path is crucial for car type robots, the algorithm chooses only the continuous-curvature local paths. Finally, an improved feedback linearization control algorithm is used to guide the car along this computed reference path.

In [6] authors present a bi-steerable car, which allow steering by turning either the back or the front wheels. Using this car, they address the integration of the four essential autonomy abilities (i.e. simultaneous localisation and environment modelling, motion planning and motion execution) into a single application. Then they build a kind of simplified occupancy grid on the environment and they apply the motion planner adopted for the CyCab, expressed as a Bayesian inference problem. Bayesian methods are also used for trajectory tracking

In [109] a new kind of public transportation system is presented, which relies on a

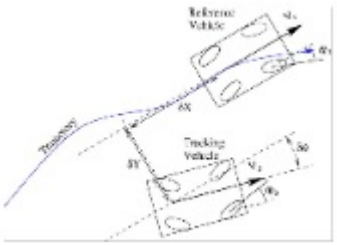


Figure 2.8: Variables involved in trajectory tracking behaviour, using Bayesian inference [6].

particular double- steering kinematic structure (as described above). The authors in this work address the integration of these four essential autonomy abilities into one application, applying a reactive execution of planned motion. In addition, they address the fusion of controls, issued from the control law and the obstacle avoidance module, using probabilistic techniques. The planner first builds a collision-free path without taking into account the non-holonomic constraints of the system. Then, this path is approximated by a sequence of collision-free feasible sub-paths computed by a suitable steering method and then is smoothed properly.

In [110] authors propose a dense stereo V-SLAM algorithm that estimates a dense 3D map representation which is more accurate than raw stereo measurements. The proposed system is composed of two main parts. First a sparse V- SLAM system based on an EKF is calculated, which takes the resulting pose estimates in order to compute a locally dense representation from dense stereo correspondences. The state vector of the EKF contains all landmark positions, the current camera pose and a subset of past camera poses. To tackle the computational complexity problem inherent to EKF SLAM, authors utilize a sub mapping method called conditionally independent sub maps. After incorporating new observations and updating the EKF state vector a new camera pose is obtained. This allows the dense part to be continuously updated.

In [111] a strategy to turn a car-like mobile robot in a restricted environment using a Simultaneous Localization and Map Building (SLAM) algorithm is presented. More

specifically, in the first step of the proposed method, the environment's information and the vehicle's pose (position and orientation) estimation is provided to the vehicle by a SLAM algorithm, which is implemented on an Extended Kalman Filter (EKF), extracting the lines and corners (convex and concave) from the environment. In the next phase, a turning algorithm, which is based on a semi-circle trajectory, following with direction switching, plans from the vehicle's initial pose the first semi-circle trajectory with respect to the environment until it reaches a neighborhood of the closest geometric map feature provided by the SLAM system state. Then, a next semi-circle trajectory is planned in the opposite direction to the previous trajectory. The proposed algorithm continues until the vehicles reaches the desired orientation, while a kinematic trajectory controller drives the vehicle through the generated paths.

Authors in [112] present a new SLAM method, called L-SLAM. It is a low dimension version of the FastSLAM family algorithms, which reduces the dimensionality of the particle filter that FastSLAM algorithms use, while achieving better accuracy with less or the same number of particles. The key idea they used is to sample only the robot's orientation on each particle, in contrast to the FastSLAM algorithms that sample the orientation along with the position of the robot.

2.7 Latest developments

In [113] authors focus on the distance metric, since they believe that it is a key component in RRT-based motion planning. The reason behind it is that the distance metric affects the coverage of the state space, the path quality and the planning time. Therefore, in order to speed up the planning time, they introduced a learning approach for approximating the distance metric for RRT-based planners. Specifically, they made some extensions to a previous work and they approximated the cost-to-go metric by a simple, offline-learned regression model with constant-time

inference. The proposed metric distance can estimate the cost of local paths through a novel extender, which is called POSQ, solving the two-point boundary value problem (2P-BVP) and producing smooth cusp-free trajectories. The new POSQ can connect any pair of 2D poses and produces RRT trees that covers the entire state space. In addition, the proposed POSQ extender makes no linearization or approximation, while at the same time is efficient to compute it. Furthermore, it is capable to produce smoother paths in shorter time with smaller trees than motion primitives. In general, the main contributions of this method could be briefly summarized as the presentation of a comprehensive comparison to an Euclidean distance baseline, the four alternative regression models (neural network regression, LWPR, SVM regression, and random forest regression), and a method for learning offline the distance pseudo-metric for the case of the POSQ extender using a set of domain-specific features and a simple basis function model with constant-time inference.

A new technique that exploits the reachable volumes is presented in [114], with which it is possible to efficiently restrict the sampling to feasible or reachable regions of the planning space, even in cases where a high degree of freedom is required, or when highly constrained environments create problems to the planning process. Based on that, they have developed a method to apply reachable volumes to tree-based planners such as Rapidly-Exploring Random Trees (RRTs), in order to adjust the stepping reachable volume samples so that to generate nearby samples that are also in the reachable volumes. In particular, they have developed a reachable volume RRT called RVRRT that can solve problems with high degree of freedom, while it can be applied in problems with many constraints. For this reason, they implemented a reachable volume stepping function, a reachable volume expand function, and a distance metric based on these operations. Additionally, they developed a reachable volume local planner to ensure that local paths satisfy some constraints from different methods, such as PRMs, showing that the RVRRTs can solve many and constrained problems with up to 64 degrees of freedom. Also, they tested it to

unconstrained problems and they claim that their method can be applied to problems with as many as 134 degrees of freedom, while at the same time the proposed RVRRTs can solve them more efficiently than existing methods, requiring fewer nodes and collision detection calls. In [115] authors present a new motion planning technique called Batch Informed Trees (BIT*), which is based on unifying graph- and sampling-based planning algorithms. In their method they assume that the recognition of a set of samples can describe an implicit random geometric graph (RGG), based on which they are able to combine the ordered nature of graph-based methods with the scalability of sampling-based algorithms, such as the Rapidly-exploring Random Trees (RRT). To manage it, the new BIT* technique uses a heuristic to efficiently search a series of increasingly dense implicit RGGs, taking into consideration also the previous information. Authors describe this process as an extension to incremental graph-search techniques, like the Lifelong Planning A* (LPA*), which is applied to continuous problem domains. Also they state that their method could be considered as a generalization of the existing sampling-based optimal planners, as it tends to be probabilistically complete and asymptotically optimal. The proposed BIT* method was tested under different cases, and on these problems, BIT* was able to find better and faster solutions than RRT, RRT*, Informed RRT*, and Fast Marching Trees (FMT*).

The method proposed by [116] intends to fill a gap which is caused when a Bi-directional search strategy is applied to increase the success and convergence rates of sampling-based motion planning algorithms. More specifically, it is evident that although there are many methods in this field, actually there are very few approaches that try to merge the bi-directional search and the asymptotic optimality into existing optimal planners, such as PRM*, RRT*, and FMT*. Therefore, in this work a bi-directional, sampling-based, asymptotically optimal algorithm is proposed, which extends the Fast Marching Tree (FMT*) algorithm to a bi-directional search. Therefore the new algorithm is called Bi-directional FMT* (BFMT*), and its intension

is to preserve the key properties and asymptotic optimality through convergence in probability. More particularly, the BFMT* method performs a two-source, lazy dynamic programming recursion over a set of randomly-drawn samples, generating two search trees. The first one in the cost-to-come space from the initial configuration, while the second one in the cost-to-go space from the goal configuration. As authors state, this is the first tree-based, asymptotically-optimal bi-directional sampling-based planner, which converges to an optimal solution at least as fast as the state-of-the-art methods, such as FMT*, PRM*, and RRT*.

Authors in [117] proposed a method based on machine learning (ML) in order to estimate the relevant region of a motion planning problem during the exploration phase of sampling-based path planners. Their method relies on the fact that the incremental sampling-based algorithms collect a lot of data about the planning problem as iterations progress. Therefore, they utilize this information in order to provide informative labels of the collected samples (obstacle or free) and to associate approximate cost values with each sample. Then, employing active learning and by inferencing based on the collected data, these labels are used to guide the selection of future samples towards the favorable region of the search space without invoking the computationally expensive collision checking and local steering procedures. In order to succeed it, the proposed algorithm guides the exploration so that it draws more samples from the relevant region as the number of iterations increases. In order to do it, it first predicts if a given sample is collision-free (classification phase) without calling the collision-checker, and then it estimates if it is a promising sample. In this case, it is checked if it has the potential to improve the current best solution (regression phase), without solving the local steering problem. Finally, the proposed exploration strategy is integrated to the RRT# algorithm, in order to guide the future exploration of the search space. In another work [7], a new non-holonomic distance function for unicycle-type vehicles was proposed, in order to extend the optimal path planner RRT* and be able to handle non-holonomic

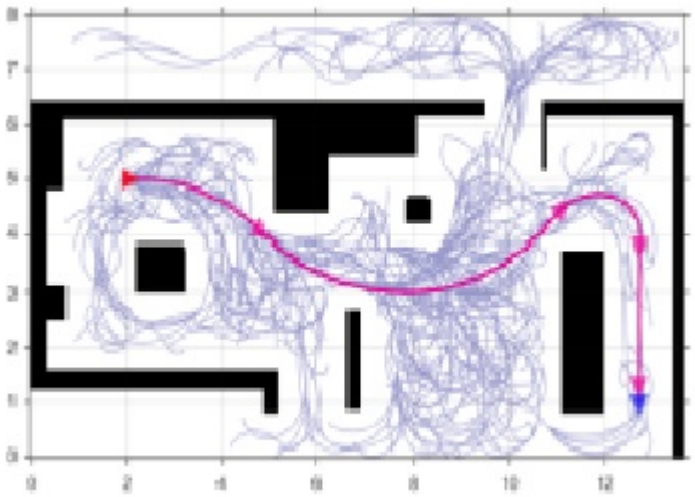


Figure 2.9: An example minimum-distance path (bold line) found by non-holonomic RRT* after 1000 vertices [7].

constraints. What makes the proposed distance function appealing is the fact that it is also a control-Lyapunov function (CLF) for the system, which makes it a natural measure of cost-to-go. This parameterized closed-form distance function can be applied from a pose (position and orientation) to a target pose for unicycle-type vehicles, while the shape of the resulting path can be controlled by the free parameters of the distance function. Therefore, as the authors state, it is possible to construct feedback policies that stabilize the system to a target pose, and also to generate the optimal path that respects the non-holonomic constraints of the system via the non-holonomic RRT*. The result of this process is the composition of the Lyapunov function, which provides stabilizing feedback and the cost-to-go to the final destination in the neighborhood of the planned path. This is the most important result, since it adds flexibility and robustness to the planning process, as well as higher efficiency.

In [118] authors describe a process which focuses more on Inverse Reinforcement Learning (IRL) for path planning, because IRL can enable robots to learn the cost functions through a demonstration process instead of hard-coding them. Based on this fact, they developed the Rapidly Exploring Learning Trees (RLT*), which is

capable to learn the cost functions of optimal Rapidly Exploring Random Trees (RRT*) through a demonstration process. In this way, they apply an inverse learning method to complex tasks, relying mainly on the Approximate Maximum Margin Planning (AMMP), a variant of Maximum Margin Planning (MMP), along with the sample-based planning algorithm RRT* cost function. In addition, a new caching scheme was introduced, which is very efficient and can reduce the computational cost when RRT* is used within AMMP. After extensive experiments, authors mention that the proposed RLT* algorithm can learn effectively the cost function in robotic tasks with obstacles and motion constraints under different social navigation scenario, using either simulated or real-robot data, achieving better performance at lower computational cost than existing methods.

A novel approach for collision-free global navigation for continuous-time multi-agent systems is presented in [119]. The proposed approach is quite generalized and could be used to perform collision-free navigation in 2D and 3D spaces, either in case of narrow passages or in crowded regions. The proposed method first precomputes the multiple bridges in the narrow or tight regions in the current workspace using kinodynamic RRT algorithms. The computed bridges have certain geometric properties that enable the calculation of a collision-free trajectory for each agent, which is also guaranteed by the fact that authors have defined specific criteria, with which when an agent enters a bridge with a velocity which satisfies them, then the new method can always compute a collision-free trajectory that lies within the bridge. In this way, the trajectory is computed using simple interpolation at runtime, and after combining the pre-computed and interpolated bridges trajectories with local multi-agent navigation algorithms, it is possible to compute global collision-free paths for each agent. Combining the above-mentioned features and methods, the proposed method can combine the performance benefits of coupled multi-agent algorithms with the precomputed trajectories of the bridges, in order to handle better different challenging scenarios, such as 3D benchmarks with narrow passages.

It is also mentioned that this approach can perform global navigation for 50-100 agents on a single CPU core, either in 2D or in 3D workspaces, presenting linear processing time in most cases and scenarios.

A work that focuses more on autonomous vehicles is presented in [120]. Authors first briefly describe the high interest of the research community in this field and especially the active development of systems such as the ADAS (Advanced Driver Assistance Systems), which points towards fully autonomous vehicles. Although in this field the motion planning is very important key technology for fully autonomous vehicles, such as in cases where they operate in constrained narrow spaces like a parking lot, it is well-known that in these environments the motion planning is very challenging because it requires many changes in forward and reverse directions, as well as adjustments of position and orientation. Therefore, they have proposed an efficient motion planning algorithm, which relies on Rapidly-exploring Random Trees (RRT) and it can specify the desired orientation of the vehicle during the tree expansion. Their method is called the desired orientation RRT (DORRT), and in order to overcome the above limitations, authors used a tangential vector space, which indicates the desired orientation and enables to model some non-holonomic constraints for the vehicle and some geometric constraints for the obstacles. For the model creation, the proposed method relies on a magnetic-field-based model, which can determine the preferred direction of a vehicle based on the non-holonomic constraints of the vehicle and the geometric constraints of obstacles. The proposed method was tested in narrow parking spaces, verifying its efficiency and performance.

A method which relies on Gaussian mixture fields is presented in [8], where authors present a new mobile robot motion planning approach based on kinodynamic constraints. Under this concept, kinodynamic constraints exploit learned perception priors in the form of continuous Gaussian mixture fields, creating statistical multi-modal motion models of discrete objects or continuous media. This representation creates a Circular Linear Flow Field (CLiFF) map, which associates a

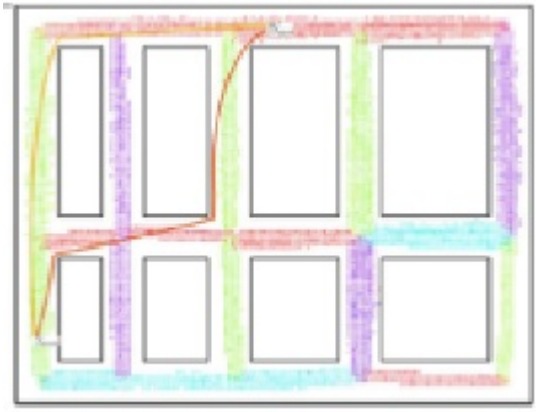


Figure 2.10: An example CLiFF-RRT* path generated in the more complex maze scenario [8].

Gaussian mixture model (GMM) to each location whose components encode multiple weighted flow directions. Specifically, this model captures the dependency between motion speed (a linear variable) and direction (a circular variable) using semi-wrapped Gaussian mixture models, creating motion models in environments that encode the dynamics of air or pedestrian flows. Then, these mixture components guide a sampling and rewiring process in an RRT* algorithm using a steer function for non-holonomic mobile robots. From the combination of the above-mentioned techniques, their method is called CLiFF-RRT*. Following with many experimental setups and using three alternative baselines, authors verify that this combination allows the planner to generate efficiently high-quality solutions in terms of path smoothness and path length, while it controls vehicle motions through the multi-modal representations of Gaussian mixture fields.

The limitations of RRT and RRT* are described and analyzed in the work of [9], who focus more on these planning techniques in high- dimensional systems such as wheeled robots with complex non-holonomic constraints. As they mention, in this applications the planning times can scale poorly for these robots, leading to the usage of hierarchical techniques that grow the RRT trees in more advanced ways. Following this direction, authors proposed a new technique, called Theta*-RRT, which hierarchically combines any-angle search with RRT motion planning for non-

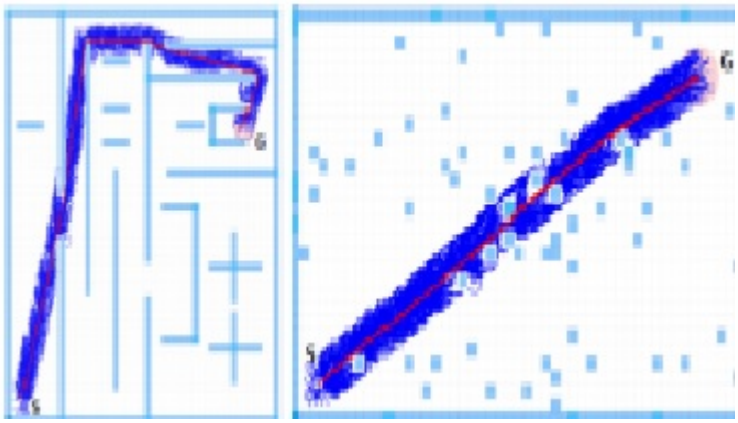


Figure 2.11: Theta*-RRT trees in two example environments [9].

holonomic wheeled robots. Their method is a variation of RRT and improves its efficiency in high-dimensional spaces by generating a trajectory which expands a tree of geodesics toward sampled states. These sampled states present distributions which summarize the geometric information of the any-angle path using steer functions instead of random control propagations. In this way, they are able to gain more knowledge of the non-holonomic constraints of the system and to ensure that their method will succeed high planning efficiency and high trajectory quality. For this reason they state that their proposed method can retain the probabilistic completeness of RRT for all small-time controllable systems that use an analytical steer function, and in order to proof it, they validated their method in various scenarios, such as in the case of a differential drive system and in the case of a high-dimensional truck-and-trailer system. As the claim, the proposed Theta*-RRT method produced shorter trajectories much faster than other baseline planners, such as RRT, A*-RRT, RRT* and A*-RRT* without losing the smoothness.

A method focusing again in high-dimensional configuration spaces is presented in [121], which tries to dig deeper in the field not of sampling-based planners such as RRT-Connect, but the search-based methods such as A*. Authors analyze the systematic nature of search-based algorithms, claiming that although it often leads to consistent and high-quality paths, it also enforces strict conditions for the connection

of forward and backward searches. Based on this claim, they argue that the currently developed admissible heuristics for the connection of forward and backward searches present high computational complexity, therefore in their work try to differentiate from this line and exploit the recent advances in search with inadmissible heuristics, in order to develop a new algorithm. Their algorithm is called A*-Connect and it relies heavily on RRT-Connect. The proposed method uses a fast approximation of the front-to-front heuristic to lead the forward and backward searches towards each other. Specifically, it runs bi-frontal searches either from the start or from the goal configuration, differentiating from the single frontier searches. In addition, it runs a front-to-back and a fast-to-compute front-to-front heuristic search in parallel from each side, exploiting the functionality of the Multi-Heuristic A* (MHA*) framework. Actually, the framework MHA* provides a multi-heuristic search that allows the usage of multiple arbitrarily inadmissible heuristics while at the same time it can preserve the completeness and bounded suboptimality conditions. By using multi-heuristic searches, authors can apply a bidirectional multi-heuristic search in a principled manner, and therefore the A*-Connect is capable to find solutions much faster than a unidirectional search. Even in this case, authors state that A*-Connect can still guarantee the bounded suboptimality, which was proven by the performed experiments and their evaluation process. Authors tested it in various domains, such as in manipulation and navigation, and they compared it with other popular sampling-based methods and state-of-the-art bidirectional search algorithms.

Chapter 3

Motivation and Contribution

Driving car-like robot autonomously is a challenging issue that requires a fundamental need of path planning process for planning complex manuvers in presence of complex scenarios. Although a simple reactive scheme suits best to avoid unforeseen obstacles, it may not be able to plan for complex manuvers and may require a forced stop. So for online path planning the author recommends to work (change) with a pre-computed set of non-holonomic paths generated in pre-processing phase.

In this Chapter, a need of new approach is reasoned aiming specifically at offline or/and online path-planning process that should be needed for autonomous navigation of car-like robot. Further it will also provide insight on fundamental components that are needed to facilitate path planning process.

3.1 Need of path planning process in autonomous navigation

Path planning of robot consists of finding trajectory from some starting pose to the goal pose. Traditionally, the robotics literature [122, 123] assumes there exist a configuration space of the robot called *C-Space*, where a point in configuration

space represents the robot pose.

The C-space is divided into C-free space and C-obstacle space. For all robot poses that collide with obstacles belong to *C-obstacles* and for poses that are collision free belongs to *C-free space*. It is often needed to find a path in C-space that completely lies in C-free space. There can be many paths that can connect the start pose to goal pose and selecting the best path under optimality constraints such as shortest distance, shortest time or both also is a part of path planning process.

3.1.1 Simple path planning for any robot

Algorithm 1 represents a common approach to solve path planning problem. Usually finding C-free space as a continuous region can be computationally expensive, so instead sampling is done to find collision free C-points in popular path planning approaches like Probabilistic Road Map (PRM) [17] and Rapidly-exploring Random Tree (RRT) [18].

Algorithm 1: Path_Plan

1: Given inputs start pose $\mathbf{q_s}$ and goal pose $\mathbf{q_g}$ of the robot and N be the number of tries to connect trajectory from $\mathbf{q_s}$ to $\mathbf{q_g}$
2: Trajectory $\Gamma = \emptyset$
3: **for** i = 1 : N **do**
4: Explore the C-space for collision free C-space
5: Start growing the trajectory using kinematic model of the robot in newly found collision-free C-space from start pose to some unexplored poses
6: **if** $\mathbf{q_g}$ is in the connected free C-space from $\mathbf{q_s}$ **then**
7: Add that grown trajectory in to set Γ
8: **end if**
9: **end for**
10: **return** Γ_{best} from set Γ using the heuristics set by user

3.1.2 Car-like robot – a robot with non-holonomic constraints

The non-holonomic system is defined as "a constrained system whose state depends on the path taken in order to achieve it" in [124]. Usually such constraints expresses

kinematic models of that system using differential equations.

For a car-like model the non-holonomic constraints are shown below in kinematic model of the car. The configuration (or pose) of car-like robot can be expressed as $\mathbf{q} = [x\ y\ \theta]$, where (x, y) is the center position of axle of rear wheels and θ is the orientation of the robot.

$$\dot{\theta} = \frac{s_i \sin \phi_i}{l} \tag{3.1}$$

$$\dot{x} = s_i \cos \theta_{i+1} \cos \phi_i \tag{3.2}$$

$$\dot{y} = s_i \sin \theta_{i+1} \cos \phi_i \times dt_i \tag{3.3}$$

, where l is the length between two axles, ϕ is the steering angle of the front wheels and s is the motion speed.

As seen in previous section while planning, randomly exploring free C-space for finding a feasible trajectory can be straightforward if no constraints are placed on robot motion, i.e., robot can move to any C-point in any straight-line direction. While such assumption is true for holonomic robots and robot arms, but for car-like robots to be from any C-point to some other C-point may require complex manuevering.

Thus, we need an extra step here on whether the robot is able to move to that newly discovered C-point. Algorithm 2 illustrates this new condition in path planning process.

Algorithm 2: Path_PlanNH

Given inputs start pose $\mathbf{q_s}$ and goal pose $\mathbf{q_g}$ of the robot and N be the number of tries to connect trajectory from $\mathbf{q_s}$ to $\mathbf{q_g}$
Trajectory $\Gamma = \emptyset$
for i = 1 : N **do**
  Explore the C-space for collision free C-space
  if The robot can move to that newly found free C-space satisfying non-holonomic constraints **then**
    Start growing the trajectory using kinematic model of the robot in newly found collision-free C-space from start pose to some unexplored poses
    if $\mathbf{q_g}$ is in the connected free C-space from $\mathbf{q_s}$ **then**
      Add that grown trajectory in to set Γ
    end if
  end if
end for
return Γ_{best} from set Γ using the heuristics set by user

3.1.3 Framework for autonomous navigation of car-like robot using way points

Given a car-like robot to drive on roadways, this section focuses on fundamental algorithms required to achieve the task. For simplicity we initially assume that there is no traffic on the road and the Algorithm 3 illustrates an approach to drive the car on the road from start configuration to goal configuration.

As seen from the algorithm the start pose and end pose of the car are given and the output that the algorithm generates is concatenated non-holonomic trajectories for car-like robot stored in variable Γ. It is often common to use GPS with the car for the human driver to navigate on the road using simple instructions, such as, turn left on juncture, etc. The algorithm below uses that information to come up with physical poses (called as *way points*) of the car at that each step. Note that the car-like robot may not be exactly at ending way-point but needs to be near that within some threshold distance.

Using Path_PlanNH algorithm discussed in earlier section is called to plan or generate the non-holonomic trajectory between intermediate way-points. So this in

essence reduces the problem from global planning (i.e. connecting entire trajectory from start to goal possibly without any car stops) into smaller steps which could be seen as local planning.

Algorithm 3: Path planning of car-like robot in absence of traffic

1: Given inputs start pose $\mathbf{q_s}$ and goal pose $\mathbf{q_g}$ of the car
2: The non-holonomic trajectory $\Gamma = \emptyset$
3: List the *global* step by step direction using map api (such as Google Map, Map Quest, etc.)
4: At each above step insert an *intermediate* pose of the car called as way-points. A set of ordered way points W of the car poses is given by:

$$W = \{\mathbf{q_1}, \mathbf{q_2}, ..., \mathbf{q_k}\} \tag{3.4}$$

5: $\Gamma = \Gamma$ + Path_PlanNH($\mathbf{q_s}, \mathbf{q_1}$)
6: **for** $i = 1$ to $k - 1$ **do**
7: $\Gamma = \Gamma$ + Path_PlanNH($\mathbf{q_i}, \mathbf{q_i + 1}$).
8: **end for**
9: $\Gamma = \Gamma$ + Path_PlanNH($\mathbf{q_k}, \mathbf{q_g}$)
10: **return** Γ

As seen from algorithm we need three fundamental algorithms: a) Step by step directions generator, (b) Converting steps to physical poses (way-points) and (c) Path planning between intermediate poses. Step by Step directions generator is already implemented by many industries and commonly available in market, such as, Google Maps [125], Map Quest [126], etc. Converting steps to way-points requires a combination of localisation [127] of car at that step achieved using GPS sensor to find physical poses of car.

However, path planning between intermediate poses remains a largely open problem where, the trajectory should not only be non-holonomic in nature but also need to be guaranteed collision-free for safety purpose. Generating non-holonomic collision-free trajectories is mostly the focus of this thesis.

3.2 Revisiting current state-of-art approaches

Most non-holonomic path generations procedure/planners [27, 28] does not take the advantage of incremental computation in finding collision free paths. This leads to recomputation of the entire non-holonomic path from start to goal if some part of the path is in-feasible. While, making sub-goals for a path can reduce this problem for current state-of-art approaches it still may not find a collision free path not to mention the added computational cost.

Autonomous car driving in real environments require to tackle unforeseen changes in obstacle positions during path planning procedure. Such procedures/planners to be applied for dynamic environments may lead to a lot of recomputations which may further have difficulty finding feasible solutions in real-time while maintaining non-holonomic constraints.

Thus current advancements needed in non-holomonic path generation process so as to be incorporated by most planners for any kind of environment are:

1. Computational simplicity in non-holonominc path generation.
2. Facilitation of incremental collision-free path tests
3. Ability to modify the existing non-holonomic path generated based on updated sensor data from the environment

This motivates us to further study our approach and apply it to a path planner to test its feasibility and applicability.

3.3 Technical characteristics of mechanism augmenting planners for non-holonomic path generation

The following section addresses the advancements needed by non-holonomic path generator for autonomous car driving.

3.3.1 Computationally inexpensive

Car like robot being mobile will have limited battery which constrains the processing power consumption. Recomputing non-holonomic paths while driving on roads may be needed which can lead to high processing if the approach itself requires high processing steps to compute a sample non-holonomic path. Thus, the critical requirement from path planner is to work by limiting computational tasks that involve computational expensive non-holonomic path generation and collision checking. While its difficult to reduce the collision checking cost, the approach mentioned in this research makes it possible to have non-holonomic path generation in real-time constraints.

3.3.2 Instantaneous

While computationally inexpensive is a necessity making the non-holonomic path generator almost instantaneous is ideal for path planning as only collision computing cost needs to be borne by the planner for making car react to avoid obstacles in real-time.

3.3.3 Simplicity in integrating with already existing path planners

Literature [128] offers wide variety of planners with different strengths and weakness pertaining to different domains. Thus augmenting a path planner to facilitate non-holonomic path generation should have minimal changes on the planner when integrating the generator into it.

3.4 Achievements and contribution

Our contribution is the incremental non-holonmic path generation which leads to multiple benefits addressed in this section.

3.4.1 Interweaving incremental collision checking with generator

While two major computational costs of path planners being non-holonomic path generation and collision checking, doing them serially one after the other can tremendously increase the computational cost making it difficult to exhibit real-time performance.

Rather, if an incremental non-holonomic step is computed and immediately check for collision the planner would immediately know if the step was favorable or not. Although, this incremental step still needs serial execution of non-holonomic path generation and then collision checking, the computational burden is much less compared to entire non-holonomic path computed and then checked for collision. So this can be viewed as if the the collision checking process is interweaved with the non-holonomic path generation process.

3.4.2 Strengths of most existing planners can now be applied for non-holonomic path planning

Most successful class of planners are sampling based planners but they are usually used for planning robotic arms or holonomic mobile robots. Their basic tendency to search the continuous C-space [122] is by sampling in C-space randomly and then connecting them together via a collision-free path(s). Usually the strength of planners is to identify the best subset of free C-space while identifying potential C-obstacle space and guiding the robot to the goal safely.

Our strategy is based on that– not only the sampled points given by planners are taken as input by non-holonomic generator, but also the control inputs (steering and acceleration/de-acceleration) are computed by the generator. Thus, the planner does not need to be concerned with control inputs and rather focus on position and orientation of the car while planning. Such isolation of control inputs from configuration space inputs facilitate existing planners to still plan in C-space.

3.4.3 Could be easily scaled for dynamic environments with existing motion planners

The notion of C-space can be extended to CT-space and same algorithm for planning in C-space can be applied to known dynamic environments. So as mentioned in previous subsection same benefits apply here.

Recomputations often needed by motion planners to plan motion if dynamic environments are uncertain. As there is obvious advantage of combining collision checking at incremental step, the new position of obstacles can be used to quickly modify the current already computed non-holonomic path that will be in collision effectively by keeping the previous collision-free configurations for non-holonomic path intact.

Although proving it to work for uncertain dynamic environments is out of scope for current thesis, the ability of non-holonomic generator to be randomly instantaneous makes it favorable to be applied for any kind of dynamic environment.

3.5 Practical aspects

Although, this thesis is aimed at showing results in simulations, the algorithms proposed are aimed at implementing on a real car-like robot. Also, the assumption of pre-processing the path planning is heavily used in this thesis.

Chapter 4

Augmented Path Planner: RRT* Fixed Nodes for Non-Holonomic robot (RRT*FN-NH)

As car-like robots are mainly aimed for outdoor activities, corresponding navigation algorithms should be able to account for the constraints imposed by the non-holonomic type of movement. A very popular and successful family of navigation algorithms is based on the Rapidly-exploring Random Tree (RRT) path planning method. In this Chapter, some variety of modifications are proposed for the basic RRT algorithm that aim to improve the performance with respect to aspects, such as, time, path length, and trajectory smoothness, while observing the non-holonomic kinematic constraints.

The main goal is to achieve an *instantaneous random* non-holonomic path for car-like robot such that the non-holonimic path has a predefined start pose but random end pose. This scheme is especially useful as most sampling path planners randomly sample the configuration space of the robot.

So, further this scheme relies on path planner to guide the instantaneous random non-holonomic path generator to reach the desired goal pose. This is the main

elegant tactic that is proposed by this thesis. Further a more constrained variant of RRT* called RRT*-Fixed Nodes (RRT*-FN) is augmented by this tactic and is called as RRT*FN-NH and can be applied to car-like robots. This chapter is dedicated to address the feasibility of this tactic and approach used by thesis.

4.1 A review of RRT*FN

Basic functions of RRT path planner were discussed in Chapter 1. A variant of RRT called RRT* takes into account the heuristics of the possible paths from start to goal configuration and returns the best path satisfying heuristics set by the user. RRT* Fixed Nodes (RRT*FN) [31], a variant of RRT*, adds an additional constraint of maximum number of nodes in a tree.

RRT* Fixed Nodes (RRT*FN) minimizes memory requirements of RRT* by removing weak nodes[1] in presence of high-performance node. The tree is grown identical to RRT* until a maximum M number of nodes is reached. If the algorithm does not find a feasible path connecting $\mathbf{q_s}$ to $\mathbf{q_e}$ then algorithm is restarted and the old node is removed whenever a new node is added.

4.1.1 Overall structure

The flowchart for overall structure for RRT*FN is shown in Figure 4.1. Similar to RRT it runs for N iterations and finds the probabilistically best path connecting robot configuration $\mathbf{q_s}$ to somewhere near to $\mathbf{q_e}$ using at most M nodes. The core functioning is carried out at connector $\mathbf{d}$, which is described in following sections.

[1]These are the nodes, although, feasible but increases the path cost based on heuristics selected by user.

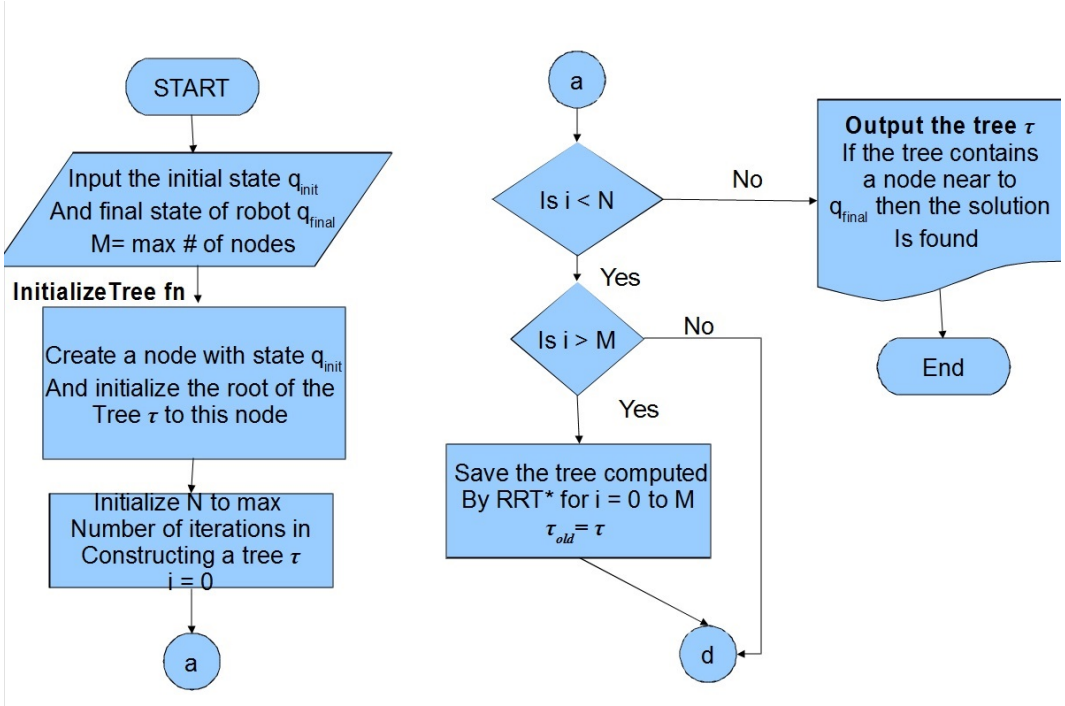


Figure 4.1: Flowchart of overall structure of RRT*FN. The bold text represents the logical function (fn) stated in RRT*FN algorithm

4.1.2 Sampling of new node

The next flowchart for sampling a new node is shown in Figure 4.2. The steps are exactly similar to that of RRT except that a $\mathbf{q_{near}}$ pose is computed that may not be same as $\mathbf{q_{rand}}$.

4.1.3 Connection of new node to tree

The next flowchart is for connecting the new node to the tree and is shown in Figure 4.3. The obstacle free function is similar to RRT as we had seen in Chapter 2. However, we have an additional "choose parent" parent function that selects the nodes based on heuristics and then makes an appropriate insertion to the parent node as shown by the flowchart.

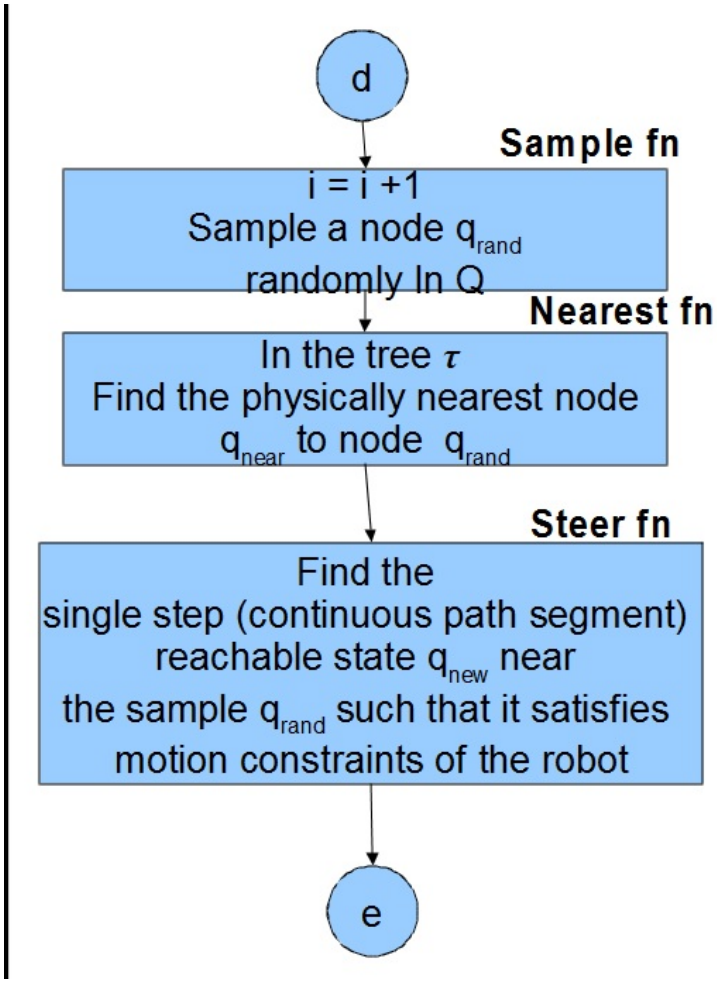


Figure 4.2: Flowchart of sampling a new node that can be connected to RRT*FN

4.1.4 Rewiring of nodes

The rewiring of nodes in tree is required to optimise the neighbourhood of the tree and also maintain the constraint of maximum fixed nodes by removing less optimal nodes in the tree. This step is specifically dedicated for that and its flowchart is shown in Figure 4.4.

4.1.5 Maintaining constraint of maximum fixed nodes

To maintain constant fixed number of nodes, if the tree exceeds the number of nodes then the only way is to delete some old node to add new node using certain criteria.

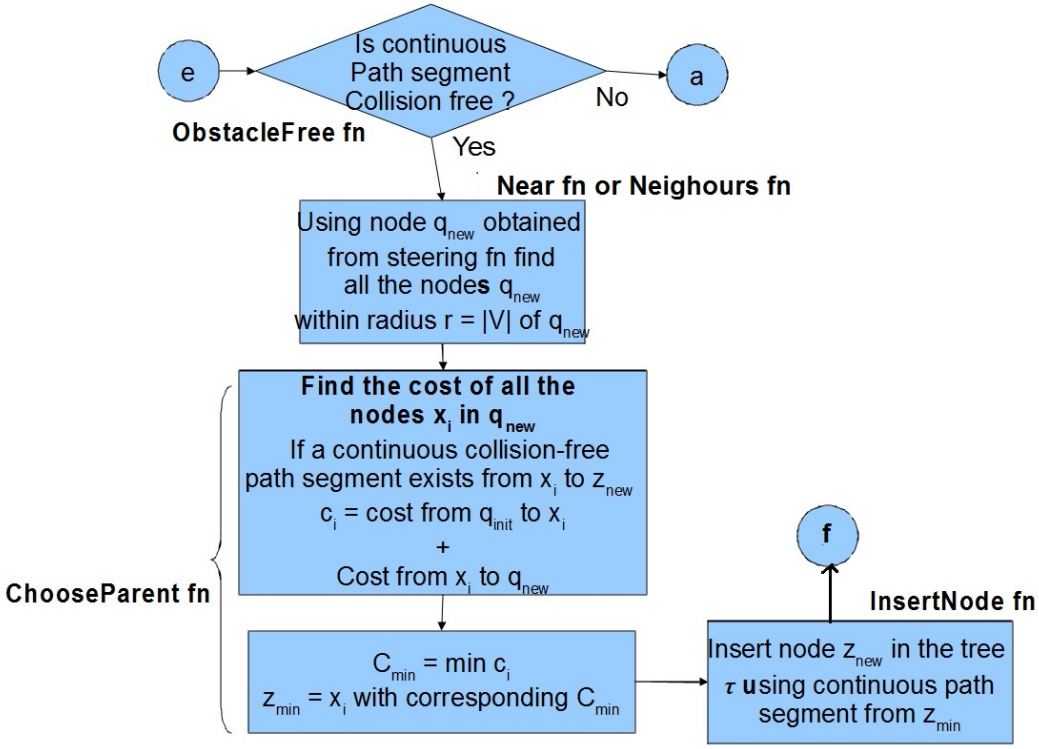


Figure 4.3: Flowchart of connecting a new node to tree

These criteria are shown in flowchart in Figure 4.5.

Basically three tactics are used,

1. does the new node make tree optimal? if not then remove the new node instead of old node
2. does the child of any node can be removed? if yes remove it
3. if no node can be removed then simply use the old tree and ignore the new node insertion process

4.1.6 Summarising into an algorithm

Similar to RRT the Algorithm 4 runs for N iterations and finds the probabilistically best path connecting robot configuration $\mathbf{q_s}$ to somewhere near to $\mathbf{q_e}$, using cost

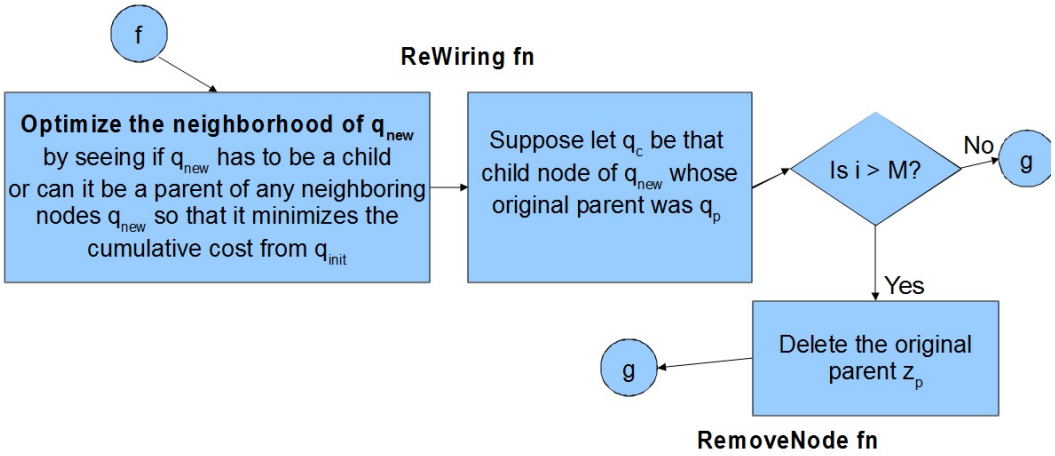


Figure 4.4: Flowchart of rewiring node of tree to optimise its neighbourhood

function $c(\Gamma)$. Similar to RRT It samples a node in $\mathbf{Q}$, finds the nearest neighbour $\mathbf{q_{near}}$, and then performs collision checking of trajectory of steering the robot from from $\mathbf{q_{near}}$ to $\mathbf{q_i}$. If that trajectory is collision free, then the node is considered to be added.

If the number of nodes exceeds max. number of allowed nodes M in a tree τ it removes a low performance node using force removal strategy[2] or simply restoring the old tree with M nodes. Thus, a new node is only added to the tree τ if the node $\mathbf{q_i}$ at iteration i makes the path to $\mathbf{q_s}$ more cost-effective.

4.1.7 Problems in Extending RRT*FN for non-holonomic systems

As seen in this section RRT*FN has proposed significant strategies to tackle path planning of holonomic robot systems. However, all the above strategies can not be directly applied to non-holonomic systems. For example, RRT*FN mentions on connecting a random node in configuration space of the robot to explore the environment. Although, for non-holonomic systems the random configuration of that system can be generated by randomly varying its joint parameters, the fundamental

[2]Search the whole tree τ and remove randomly one node with no children.

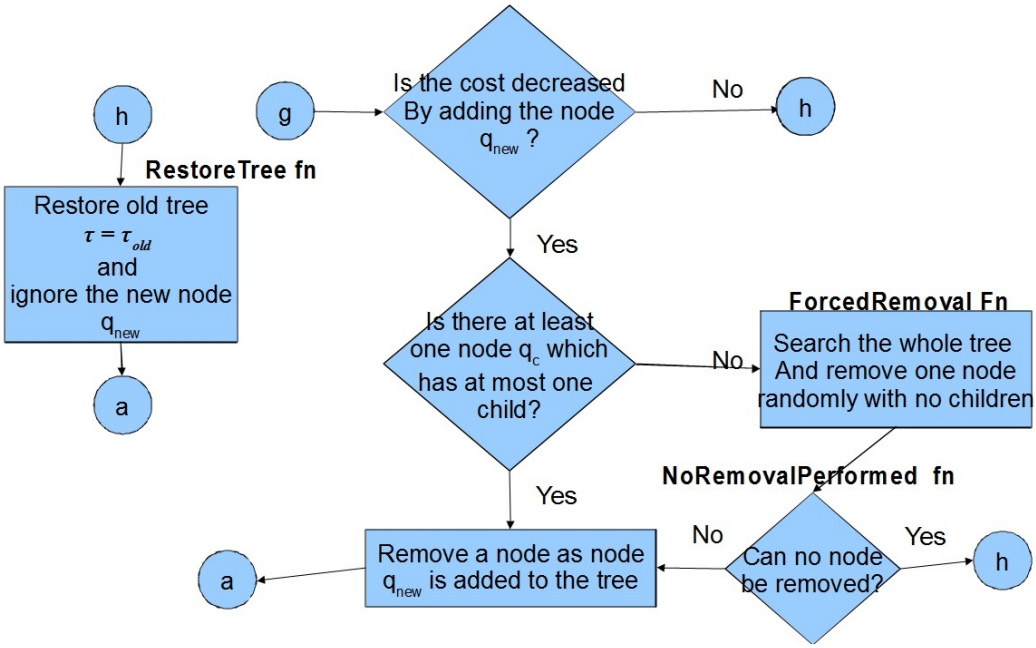


Figure 4.5: Flowchart of various tactics used to maintain fixed number of nodes

problem lies in maintaining non-holonomic constraints between this random node and the nearest node in the tree.

Rather, to maintain such constraints is solving inversely for a connected path that is computationally expensive. Also the incremental planning style of RRT breaks down. Due to this problem the effectiveness of rewiring strategy seems unattractive. In the next sections the author proposes on how to apply most of the strategies from RRT*FN for non-holonomic systems.

4.2 Problem Statement

The motion of the car-like robot is commonly described by a *control* vector $\mathbf{u} = [\phi\ s]$, where ϕ is the steering angle of the front wheels and s is the motion speed. The path generation of such a robot satisfying non-holonomic constraints is introduced in this section.

The configuration (or pose) of car-like robot can be expressed as $\mathbf{q} = [x\ y\ \theta]$, where

Algorithm 4: RRT*FN

Input $\mathbf{q_s}$, $\mathbf{q_e}$, N and M
Initialize tree τ with one node $\mathbf{q_s}$
for $i = 1$ to N **do**
  if the number of nodes added to τ exceeds M **then**
    $\tau_{old} \leftarrow \tau$
  end if
  Sample a node $\mathbf{q_i}$ randomly in $\mathbf{Q}$
  $\mathbf{q_{near}} \leftarrow n$earest node to $\mathbf{q_i}$ in τ
  $[\mathbf{q_i}\ u_{i-1}] \leftarrow S$teer the robot from $\mathbf{q_{near}}$ to $\mathbf{q_i}$
  if Steering the robot from $\mathbf{q_{near}}$ to $\mathbf{q_i}$ is $c$ollision-free **then**
    Insert node $\mathbf{q_i}$ in τ using control vector $\mathbf{u_{i-1}}$
    Using local neighbourhood re-wire the node $\mathbf{q_i}$ to some other parent that minimizes path cost to $\mathbf{q_s}$
    if $\exists\ \tau_{old}$ and new node $\mathbf{q_i}$ is cost-effective **then**
      if $\exists$ a node $\mathbf{q_\emptyset}$ with one or no child **then**
        Force remove that node $\mathbf{q_\emptyset}$
      end if
    end if
    if no node can be removed or new node $\mathbf{q_i}$ is not cost-effective **then**
      $\tau \leftarrow \tau_{old}$ when *no removal performed*
    end if
  end if
end for
$c(\Gamma_{best}) = \min\{c(\Gamma), \forall$ feasible Γ from $\mathbf{q_s}$ to near $\mathbf{q_e}$
$r$eturn Γ_{best}

(x, y) is the center position of axle of rear wheels and θ is the orientation of the robot w.r.t. world frame $\{W\}$ (see Figure 4.6). Let $\mathbf{Q}$ be the configuration space of the robot, which is three dimensional.

The problem can be formulated as:

- **Inputs**
  - $\mathbf{q_s}$: The starting configuration of the car described by three parameters $[x_s\ y_s\ \theta_s]$
  - $\mathbf{u_s}$: The starting control vector applied to generate configuration $\mathbf{q_s}$
  - $\mathbf{q_e}$: The ending configuration of the car described by three parameters $[x_e\ y_e\ \theta_e]$

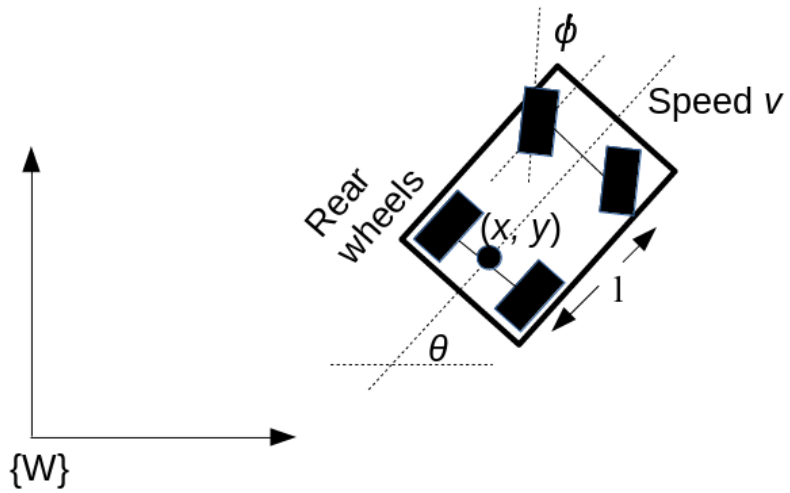


Figure 4.6: Symbols used in describing the control vector **u** and the configuration **q** of the car-like robot.

- **Output**
  - Find a path (refer Figure 4.7) described by configurations $\mathbf{q_i}$ and their corresponding control vectors $\mathbf{u_{i-1}}$ connecting the configurations $\mathbf{q_s}$ to $\mathbf{q_e}$ subject to non-holonomic constraints and trajectory profile constraints, such as, max velocity, acceleration, etc.

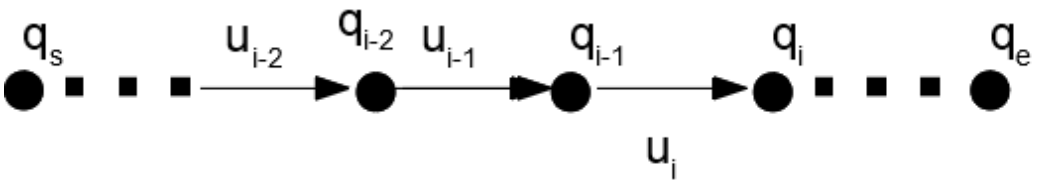


Figure 4.7: Applying control vector $\mathbf{u_{i-1}}$ to $\mathbf{q_{i-1}}$ results in new pose $\mathbf{q_i}$ of the robot

4.3 Non-holonomic path generation using small *dt*

Since non-holonomic equations are differential in nature they can only be formulated to find the next configuration $\mathbf{q_{i+1}}$ using previous robot pose $\mathbf{q_i}$ and the control vector $\mathbf{u_i}$, if :

- the differential time dt is assumed to be constant
- the same non-changing control vector input $\mathbf{u_i}$ is applied within that time interval dt
- can be useful for planning if one-step motion can be computed by assuming dt to be some small value, which is propotional to acceleration of the car

In this section, an instantaneous random non-holonomic path generator is devised after modelling the kinematics of car-like robot using a small dt.

4.3.1 Modelling kinematics of car-like robot

Kinematic model of the car-like robot satisfying non-holonomic constraints is widely known in literature [129]. Applying control vector $\mathbf{u_i} = [\phi_i \; s_i]$ to the robot at current configuration $\mathbf{q_i} = [x_i \; y_i \; \theta_i]$ for a small time interval dt_i, the resulting configuration $\mathbf{q_{i+1}} = [x_{i+1} \; y_{i+1} \; \theta_{i+1}]$ can be expressed using following equations:

$$\theta_{i+1} = \theta_i + \frac{s_i \sin\phi_i}{l} \times dt_i \tag{4.1}$$

$$x_{i+1} = x_i + s_i \cos\theta_{i+1} \cos\phi_i \times dt_i \tag{4.2}$$

$$y_{i+1} = x_i + s_i \sin\theta_{i+1} \cos\phi_i \times dt_i \tag{4.3}$$

, where l is the length between two axles.

If the current time is t_i, then $t_{i+1} = t_i + dt_i$ is the time when robot is at configuration $\mathbf{q_{i+1}}$. The small time interval $dt = t_{i+1} - t_i$ enables to make simple assumption such

as the speed s_i remains nearly constant from time t_i to t_{i+1} and we can compute $\mathbf{q_{i+1}}$ directly using above differential equations.

The small interval dt_i is also chosen such that the incremental distance the robot covers can be approximated by a constant straight line euclidean distance d_λ.

4.3.2 Random non-holomic instantaneous path generator

The possible speed s_{i+1} for computing next configuration $\mathbf{q_{i+2}}$ is limited upon acceleration/de-acceleration a applied to the robot within time interval dt_i and the change in speed $ds_{i+1} = s_{i+1} - s_i$ can be expressed as:

$$-adt_i \leq ds_{i+1} \leq adt_i, \text{where} \quad dt_i \leq \frac{d_\lambda}{ds_{i+1}} \tag{4.4}$$

The Algorithm 5 implements next configuration generation satisfying non-holonomic constraints and constraint expressed in eqn.(4.4). The algorithm requires presetting distance threshold d_λ, and maximum magnitude of acceleration/de-acceleration of the car a_{max}. To build a path Γ for car-like robot with n configurations, Algorithm 5 can be iterated for $i = 1$ to n given the starting configuration $\mathbf{q_0}$ and $\mathbf{u_{-1}} = [0\ 0]$.

Algorithm 5: Next configuration $\mathbf{q_i}$ generation

1: Input starting configuration $\mathbf{q_{i-1}}$ and $\mathbf{u_{i-2}}$
2: $r = [0, 1]$ be a random generator
3: $dt_{i-1} = \sqrt{\frac{d_\lambda}{a}}$, where $a = ra_{max}$
4: $s_{i-1} = s_{i-2} + (-1 + 2r)adt_{i-1}$
5:
$$\phi_{i-1} = \begin{cases} \phi_{i-2} & \text{if } r < 0.5 \\ \phi_{i-2} + (-1 + 2r)30^\circ & \text{Otherwise} \end{cases}$$
6: Compute $\mathbf{q_i} = [x_i\ y_i\ \theta_i]$ using eqn.(4.1– 4.3) with $\mathbf{u_{i-1}} = [\phi_{i-1}\ s_{i-1}]$ and $\mathbf{q_{i-1}} = [x_{i-1}\ y_{i-1}\ \theta_{i-1}]$

The path generation using algorithm 5 needs a path planner to lead the car from starting configuration to a pre-desired goal configuration. Specifically, the planner

can find a sequence of control actions $\mathbf{u}$ to drive the robot from initial configuration $\mathbf{q_s}$ to some final configuration $\mathbf{q_e}$ in presence of constraints imposed by environment and by robot dynamics. The prospects of RRT is cited in [18]. It randomly samples the configuration space of the robot and tries to connect that sampled configuration (node) $\mathbf{q_i}$ to its nearest configuration (node) $\mathbf{q_{near}}$ in the tree τ rooted at node $\mathbf{q_s}$.

For this paper, the relevant variants of RRT to be considered are RRT* [130] and RRT*FN [31]. RRT* introduces heuristics into RRT along-with the concepts of local neighbourhood of newly added node and its re-wiring. Thus, RRT* finds a feasible path Γ_{best} from $\mathbf{q_s}$ to $\mathbf{q_e}$ using a cost function $c(\Gamma)$.

4.4 RRT*FN with non-holonomic constraints

Enabling RRT*FN to function for non-holonomic robots is the focus of this section. As Algorithm 5 generates an incremental step in a random direction, RRT*FN can guide the series of generation of incremental steps towards goal configuration. Also, maintaining the fixed nodes strategy could be adopted here.

4.4.1 Simple modifications to RRT*FN

Algorithm 6 (called as RRT*FN-NH) proposes on making RRT*FN work for robot with non-holonomic constraints. Similar to RRT it samples a node $\mathbf{q}$ in the configuration space $\mathbf{Q}$ of the robot but does not uses that node to connect to nearest node $\mathbf{q_{near}}$ in tree τ. Instead, using Algorithm 5 it generates the next configuration $\mathbf{q_j}$ using $\mathbf{q_{near}}$ and control vectors $\mathbf{u_{i-2}}$ and $\mathbf{u_{i-1}}$. The parent control vector $\mathbf{u_{i-2}}$ is needed in generating the next configuration $\mathbf{q_j}$ (Algorithm 5). Using the output control vector $\mathbf{u_{i-1}}$ of Algorithm 5 the robot is steered to the next incremental configuration. Thus, RRT*FN-NH while satisfying non-holonomic constraints explores

the configuration space and tries to connect to the goal using similar principle of RRT*FN.

Algorithm 6: RRT*FN-NH

1: *I*nitialize tree τ with one node $\mathbf{q_s}$ and $\mathbf{u_{-1}} = [0\ 0]$
2: **for** $j = 1$ to N **do**
3: **if** the number of nodes added to τ exceeds M **then**
4: $\tau_{old} \leftarrow \tau$
5: **end if**
6: *S*ample a node $\mathbf{q}$ randomly in $\mathbf{Q}$
7: $\mathbf{q_{near}} \leftarrow$ *n*earest node to $\mathbf{q}$ in τ
8: $\mathbf{u_{i-2}} \leftarrow$ the control vector applied at parent of $\mathbf{q_{near}}$
9: $[\mathbf{q_i}\ u_{i-1}] \leftarrow$ Using Algorithm 5 *g*enerate $\mathbf{q_i}$ with $\mathbf{q_{i-1}} = \mathbf{q_{near}}$ and $\mathbf{u_{i-2}}$
10: $\mathbf{q_j} = \mathbf{q_i}$ and store u_{i-1} with node $\mathbf{q_j}$
11: **if** robot at $\mathbf{q_i}$ is *c*ollision-free **then**
12: *I*nsert node $\mathbf{q_i}$ in τ using control vector $\mathbf{u_{i-1}}$
13: **if** $\exists\ \tau_{old}$ and new node $\mathbf{q_j}$ is cost-effective **then**
14: **if** $\exists$ a node $\mathbf{q_\emptyset}$ with one or no child **then**
15: *Force remove* that node $\mathbf{q_\emptyset}$
16: **end if**
17: **end if**
18: **if** no node can be removed or new node $\mathbf{q_j}$ is not cost-effective **then**
19: $\tau \leftarrow \tau_{old}$ when *no removal performed*
20: **end if**
21: **end if**
22: **end for**
23: $c(\Gamma_{best}) = \min\{c(\Gamma), \forall$ feasible Γ from $\mathbf{q_s}$ to near $\mathbf{q_e}$
24: *r*eturn Γ_{best}

4.4.2 Highlights on RRT*-FN-NH

While most of the strategies are taken from RRT*-FN. There lies these fundamental differences that had to be proposed for it to function with non-holonomic systems:

1. Instead of sampling a node in configuration space, the node is randomly sampled from a tree and a random control vector is applied to it. So this dual approach to sampling strategy by RRT variants is proposed.

2. While the non-holonomic systems are differential in nature and RRT*-FN can not function with it, the incremental step planning style of RRT is exploited.

Since, the step is incremental, the next node can be directly computed using differential equations while maintaining non-holonomic constraints. So this key insight is explored by this thesis. Not only it make RRT*FN functional for non-holonomic systems but also provides a means for any incremental planner to adopt this strategy.

3. Also the rewiring strategy is removed for RRT*-FN-NH. Instead with above elegant strategies RRT*-FN-NH was made functional and effective. Chapter 5 discusses the use of the above collective strategies as mentioned in Algorithm 6 in various environments for a car-like robot.

The focus of this thesis was to facilitate any planner to be applied for non-holonomic systems with less changes as possible. So as seen with RRT*-FN-NH only a few strategies were not used and new additional strategies were proposed.

There are few additional differences between RRT*FN and RRT*FN-NH. The collision checking of robot configuration is done instead of a trajectory while steering the robot from $\mathbf{q_{near}}$ to $\mathbf{q_j}$. As the next incremental step is small, possibility of finding a feasible configuration is much higher than a trajectory steering the robot. This also provides simple computationally less-intensive collision checking of a robot configuration instead of continuous collision checking of a trajectory.

Chapter 5

Experiments and Results

Testing the following parameters for RRT*FN-NH in different scenarios is the main focus of this Chapter.

1. # **of Iterations** N**:** The number of iterations will show the *feasibility* of the approach for real-time constraints. If number of iterations required are large, as shown in [131] RRT-FN-NH should converge, similar to any sampling based planner. However, note that since in every iteration there is always a motion step satisfying non-holonomic constraints the only reason that a node is not attached to tree is because of collision with environment.

2. # **of Nodes** M **:** The number of nodes needed in the tree to find a solution directly relates to the memory needed by the embedded system on-board the car-like robot. So a test is required to give an indication on cost of reducing the number of nodes to find paths in traditional environments.

3. **Total Length of Path** L**:** The total length of the path for traditional benchmark environments will show the optimality of the approach in finding a solution path while maintaining non-holonomic constraints. This step can be critical as if the non-holonomic path has too many curves, where a simple straight line motion would suffice, then not only its risky driving but also

uncomfortable for passengers in the car-like robot. To control this a special step is introduced in Algorithm 7 that controls the randomness in steering angle change.

4. **Total Time of Path** T: Total time of path tests whether the acceleration constraints set by user is met most of the time or not. Maintaining non-holonomic constraints if the car moves near max. speed set by user most of the time, then RRT*FN-NH has been optimal in finding the paths and slowing down as a sharp turn is encountered.

5. **Computation time**: The time of path computation is not affected by non-holonomic node generation but only by collision checking cost. Also this would not play a role if the initial dense tree is planned offline. However, for online path adaptation the computation time will be dependent on collision-checking cost that could be minimised by the planner but currently it is out of scope of this research.

Experiments were conducted on modified RRT*FN toolbox [31] in Matlab. The tool originally provides appropriate implementation of RRT*FN mostly for robots, like an arm, rectangular moble robot, etc. that was modified by the author to work for non-holonomic constraints.

5.1 Specific details on implemented approach

In this section some of the critical design constraints that were implemented are discussed.

5.1.1 Minimising steering angle change

For every single step taken if the steering angle keeps on changing as mentioned in Algorithm 5, then the car-like robot would rarely move in a straight smooth jerk-free motion. To make a non-holonomic path that would make appropriate turns the following Algorithm 7 is used in implementation instead of Algorithm 5 as mentioned in Chapter 4. Also note that the bounds on the speed of the car is also checked.

Algorithm 7: Next configuration $\mathbf{q_i}$ generation

Input starting configuration $\mathbf{q_{i-1}}$ and $\mathbf{u_{i-2}}$
$r = [0, 1]$ be a random generator
$dt_{i-1} = \sqrt{\frac{d_\lambda}{a}}$, where $a = ra_{max}$
$s_{i-1} = s_{i-2} + (-1 + 2r)adt_{i-1}$
if $s_{i-1} < 0$ **then**
    $s_{i-1} = a \times dt_{i-1}$
end if
if $s_{i-1} < v_{max}$ **then**
    $s_{i-1} = v_{max}$
end if
Generate random number r
if $r < 0.5$ **then**
    $\phi_{i-1} = 0$
else
    Again generate random number r

$$\phi_{i-1} = \begin{cases} \phi_{i-2} & \text{if } r < 0.5 \\ \phi_{i-2} + (-1 + 2r)30^\circ & \text{Otherwise} \end{cases}$$

end if
Compute $\mathbf{q_i} = [x_i\ y_i\ \theta_i]$ using eqn.(4.1– 4.3) with $\mathbf{u_{i-1}} = [\phi_{i-1}\ s_{i-1}]$ and $\mathbf{q_{i-1}} = [x_{i-1}\ y_{i-1}\ \theta_{i-1}]$

5.1.2 Collision checker

While collision checking algorithm poses severe computational constraints in real-time for finding a collision-free path, studying their effect in this section is beyond

the scope of the research. For two different environments different collision checking algorithm were used.

Benchmark Environments

A simple 2D object oriented bounding box (OBB) collision check was implemented to do fast collision checking in case for traditional environments, where obstacles are polygonal in nature (shown in red). The algorithm 8 shows the collision detection between a car pose and obstacles in environment.

Algorithm 8: Collision checker for Benchmark Static Environments

```
1: Input pose q_i of the robot and set of static polygonal obstacles described by
   ordered vertices
2: Fit Axis Aligned Bounding Box (AABB) to all the polygonal obstacles
3: OBB of car is same as the car geometry described with ordered vertices at
   pose q_i
4: for each AABB o describing a static obstacle do
5:    if rectangle o is in collision with oriented rectangle OBB then
6:       return collision exists
7:    end if
8: end for
9: return no possible collision exists
```

Google Maps

For testing non-holonomic path generation on Google maps a simple collision checking whether pixels occupied by the car are on the road pixels or not was developed. Algorithm 9 shows computing free space occupancy of the car. All pixels that are not road pixels are considered as obstacle pixels.

5.2 RRT*FN-NH in traditional environments

Most RRT based algorithms are tested on common benchmark based environments consisting of polygonal obstacles that we refer here as traditional environments.

Algorithm 9: Collision checker for Google Maps

1: Input pose $\mathbf{q_i}$ of the robot and set of pixels P_{road} belonging to the road on the map
2: Find the set of all the pixels P_{car} occupied by the car at pose $\mathbf{q_i}$
3: **if** $P_{car} \subset P_{road}$ **then**
4: return no possible collision exists and car is on the road
5: **else**
6: return collision exists and car is off road
7: **end if**

Although, the results produced here are not compared with some other approach, the results here can be used to get a measure on the performance of RRT*FN-NH under the non-holonomic constraints assumption.

5.2.1 Initial parameters

We tested the RRT*FN with non-holonomic constraints in many different environments. The environment was a bounded rectangle area of size 10 unit $\times$ 10 unit. The car robot tested had dimension 0.58×0.38 unit^2. The minimum and maximum velocity of the car was set to 0.001 and 0.05 unit per seconds respectively. The length between the two axles of the car was 0.38 units. The maximum acceleration of car was 0.04 units/sec^2. The incremental or discritization step d_λ was set to a constant of 0.1 units.

Random seed as a parameter

The random generator needs a random seed in Matlab for generating random numbers. If the random seed is same, then the random numbers generated will be appearing in same sequence. Thus, if the random generator generates random numbers not in favour of path planning steps it might happen that because of that chosen random seed the solution found was not feasible.

So to get results the random seed was chosen specifically by the user. There are

experiments conducted to show the effect of random seed parameter on finding a feasible solution.

5.2.2 Feasibility of RRT*FN-NH in finding solutions

Table 5.1 shows the results of six tests with following output parameters: L as path length and T as total time taken for robot to reach the goal. The input parameters to RRT*FN for non-holonomic constraints are N (the number of iterations in RRT*FN) and M (the max. number of nodes in tree τ).

Test # 1 shows a traditional example of narrow passage (Fig. 5.1(a)) in path planning problem. Test # 2 enforces the car to take multiple turns (Fig. 5.1(b)) without stopping the car as min. velocity of car is greater than 0. Test # 3 checks if the car finds the shortest route (Fig. 5.1(c)) compared to longer route; hence, the number of iterations are set to 50,000. Test # 4 checks if the car can take sharper turn (U-turn) to reach the goal (Fig. 5.1(e)). Test # 5 (Fig. 5.1(e)) and # 6 (Fig. 5.1(f)) are for the same environments but with different set accelerations that result in different total time.

As seen in Figure 5.1 RRT*FN for non-holonomic constraints always find a solution but requires iterations of at least 10,000. Two factors affect the number of iterations, the discritization step and the non-holonmic constraints of the robot. As seen in results L and T is crucial parameters and relevant to environment and the constraints of the car-like robot movement. Also, for Test # 4, the acceleration was reduced to 0.004 units/sec^2 to obtain feasible solution in 10,000 iterations, but total time increased. So we conducted test # 6 with the same acceleration as original and with 10,000 iterations no feasible solution was obtained. So we had to increase the iterations upto 50,000. Thus, if the speed of the car is high most of the time then the chances of it hitting the obstacles are higher and more iterations will be needed.

Table 5.1: Experimental data of RRT*FN with Nonholonomic constraints

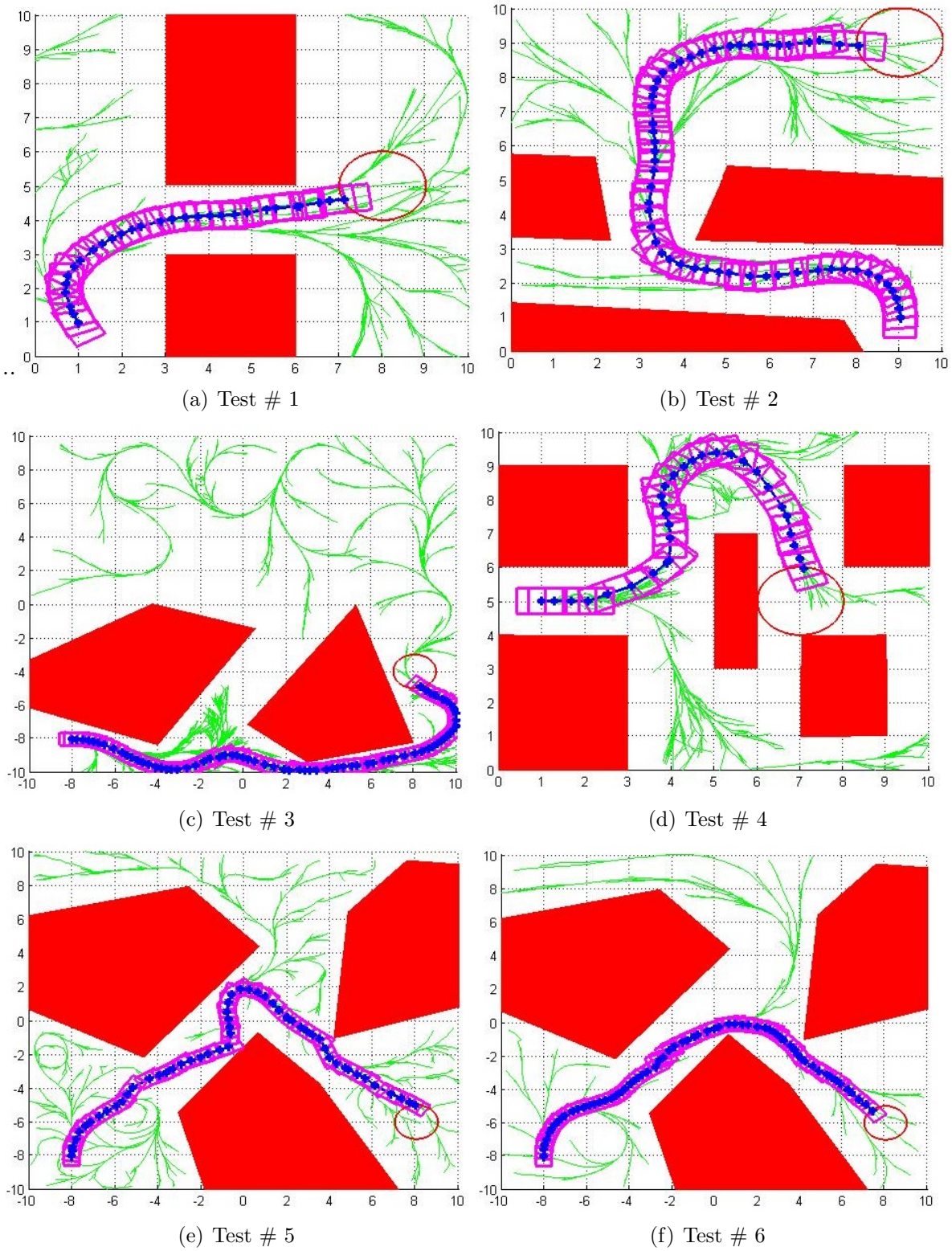
Test #	**N** ($\times 10^3$)	**M** ($\times 10^3$)	**L** (units)	**T** (secs)
1	10	1	9.02	67.27
2	10	1	18.25	124.1
3	40	15	24.08	162.6
4	10	1	13.12	221.29
5	10	1	27.40	395.01
6	50	1	22.08	156.085

5.2.3 Study of narrow passage example

A narrow passage problem poses a challenge to path planner such that the path planner needs to explore (instead of exploit) the high dimensional configuration space. Thus, how fast can it explore the configuration space to find a feasible solution is focus of this section.

As discussed earlier random seed plays an important role in finding feasible solutions, the test will take into account the random seed as a parameter. Also if same problem was given to traditional RRT* path planner using the Algorithm 7 for non-holonomic path generation, the number of nodes required to form a feasible path is given in Table 5.2 for comparison with RRT*-FN-NH. For all the experiments N was set to 5000 iterations.

So RRT* here is same as RRT*FN-NH with the condition $M = N$, i.e., the number of nodes equals number of iterations. So as seen in Table 5.2 M is deliberately chosen much smaller value than N so as to see if the memory conservation can be achieved. Clearly from the table most of the cases RRT*FN-NH is able to find feasible solution with less associated memory cost but increased Total time T of the path. This indicates that car-like robot had to move more slowly to cross the narrow passage when constrained by max. number of nodes allowed. Figure 5.2 illustrates an example that compares RRT* generated path with RRT*FN-NH path.



(a) Test # 1 (b) Test # 2

(c) Test # 3 (d) Test # 4

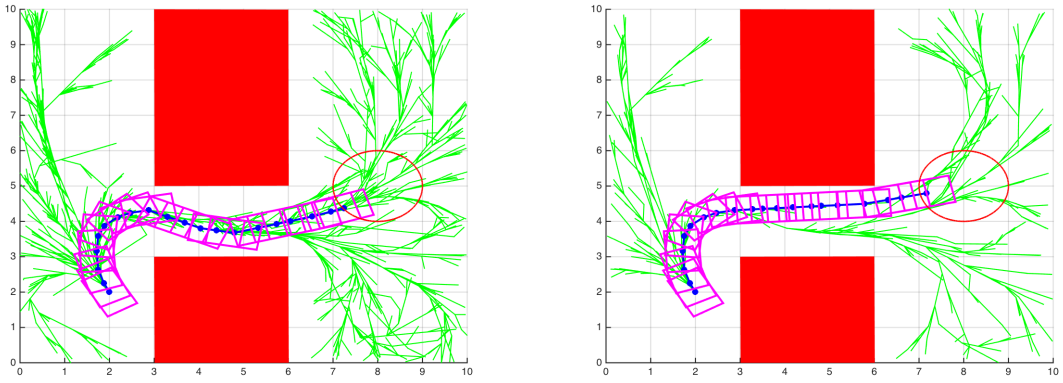
(e) Test # 5 (f) Test # 6

Figure 5.1: A path Γ_{best} returned by RRT*FN NH

Another interesting observation is for random seed # 80 where exactly reverse case is observed, i.e., as $M < N$, as memory cost decreases so does total time but there

Table 5.2: Experimental data of RRT*FN-NH Vs. RRT* for narrow passage environment

Random Seed #	**Path Planner**	**M** ($\times 10^3$)	**L** (units)	**T** (secs)	**Feasible**
10	RRT*	2225	8.3941	128.6839	Yes
	RRT*FN-NH	1225	–	–	No
20	RRT*	2379	7.9951	125.1129	Yes
	RRT*FN-NH	879	7.695	155.3891	Yes
30	RRT*	2472	8.3079	172.9373	Yes
	RRT*FN-NH	972	8.257	181.15	Yes
40	RRT*	2875	9.3683	163.6372	Yes
	RRT*FN-NH	875	9.3495	159.8945	Yes
50	RRT*	2608	7.6588	109.5401	Yes
	RRT*FN-NH	1108	7.7038	151.7836	Yes
60	RRT*	2647	8.9314	169.0311	Yes
	RRT*FN-NH	1147	8.8929	172.71	Yes
70	RRT*	2292	9.0344	122.595	Yes
	RRT*FN-NH	1292	8.2007	168.2634	Yes
80	RRT*	2536	8.5819	169.2465	Yes
	RRT*FN-NH	1036	9.4647	143.2935	Yes
90	RRT*	2942	9.0344	122.595	Yes
	RRT*FN-NH	942	9.0344	122.595	Yes
100	RRT*	2620	8.7925	132.4406	Yes
	RRT*FN-NH	1120	8.0729	140.7413	Yes



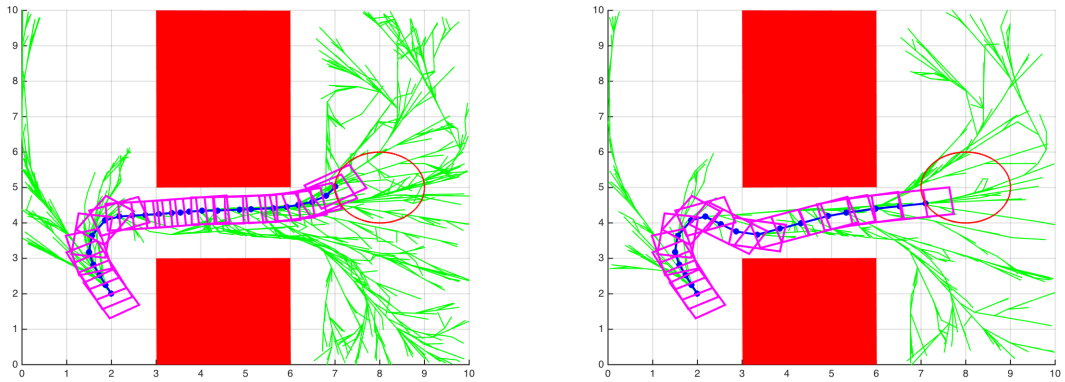
..

(a) RRT* (b) RRT*FN-NH

Figure 5.2: Path Γ_{best} returned for narrow passage example with Random Seed # 100

is increase total length of the path. Figure 5.3 illustrates the non-holonomic path generated by RRT* and RRT*FN-NH. As seen with this example with RRT*FN-

NH the car-like robot moves more towards the lower obstacle of narrow passage and then makes the car go straight to the goal with more speed, thus, decreasing the total time but increasing path length.



..

(a) RRT* (b) RRT*FN-NH

Figure 5.3: Path Γ_{best} returned for narrow passage example with smaller total time and larger total length with Random Seed # 80

However, there exists 1 in 10 case where the RRT*FN-NH fails to find a feasible solution when $M < N$ for random seed # 10. This indicates that memory can not be made arbitrary small and is a function of the complexity of the environment that RRT*FN-NH is trying to solve. However, this section shows that there are many benefits of using RRT*FN-NH and are listed below:

- Ability to solve narrow passage problem for non-holonomic constraint robot using less number of nodes compared to RRT*
- Total path length mostly decreases when RRT*FN-NH is used
- Memory constraints can be achieved but is a function of the complexity of the environment to be solved for.

5.2.4 Study with environment requiring multiple manuevers

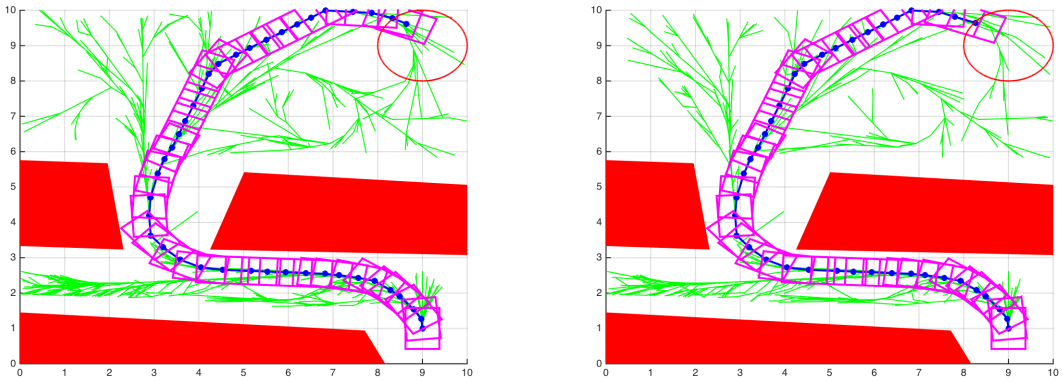
A car-like robot may need to perform multiple manuevers to reach its intermediate destination or final destination. Solving traditional environment 4 requires at least two turning manuevers to avoid central obstacles as shown in Figure 5.1(b).

Similar to the study of narrow passage environment, the random seed parameter is used to compare RRT* with RRT*FN-NH and the results are given in Table 5.3. The number of iterations for all the experiments were set to constant 5000.

Table 5.3: Experimental data of RRT*FN-NH Vs. RRT* for car needing multiple manuevers to reach goal

Random Seed #	**Path Planner**	**M** ($\times 10^3$)	**L** (units)	**T** (secs)	**Feasible**
10	RRT*	1765	19.2683	298.0733	Yes
	RRT*FN-NH	1265	19.2683	298.0733	Yes
20	RRT*	1275	17.7123	275.443	Yes
	RRT*FN-NH	775	17.777	313.4863	Yes
30	RRT*	2213	18.8971	315.9048	Yes
	RRT*FN-NH	713	16.3752	308.333	No
40	RRT*	2197	18.1019	329.2288	Yes
	RRT*FN-NH	1197	17.9108	314.303	Yes
50	RRT*	1688	18.6306	317.5444	Yes
	RRT*FN-NH	1188	18.2871	311.696	Yes
60	RRT*	2594	17.1476	275.834	Yes
	RRT*FN-NH	594	17.1477	252.610	Yes
70	RRT*	2662	18.7617	260.0442	Yes
	RRT*FN-NH	1162	18.7617	260.0442	Yes
80	RRT*	917	18.5985	388.5194	Yes
	RRT*FN-NH	917	18.5985	388.5194	Yes
90	RRT*	2729	17.7479	280.2562	Yes
	RRT*FN-NH	1229	17.8295	300.1967	Yes
100	RRT*	2719	18.0902	276.6442	Yes
	RRT*FN-NH	719	18.2293	261.144	Yes

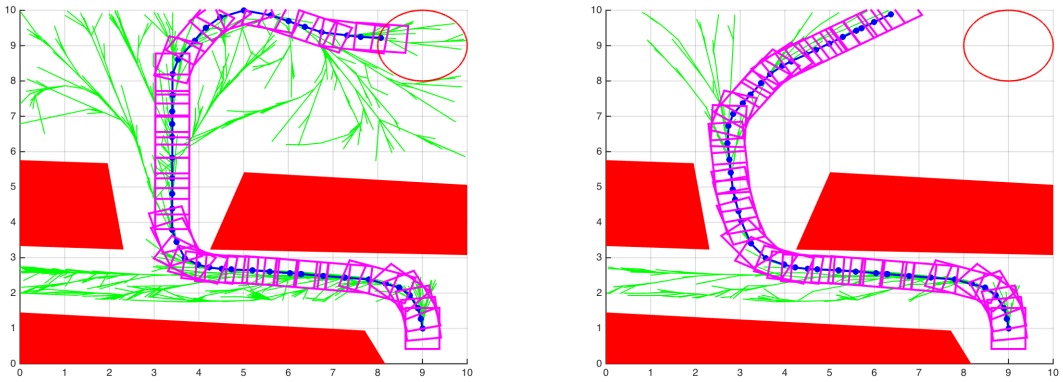
Although, the number of nodes can be decreased the total path length achieved is nearly same as that of RRT* and also total time has increased (see Figure 5.4 for example). Also, there exists a case, where the path found was unable to connect to



(a) RRT* (b) RRT*FN-NH

Figure 5.4: Path Γ_{best} returned for Random Seed # 50, where car needed multiple manuevers to reach goal

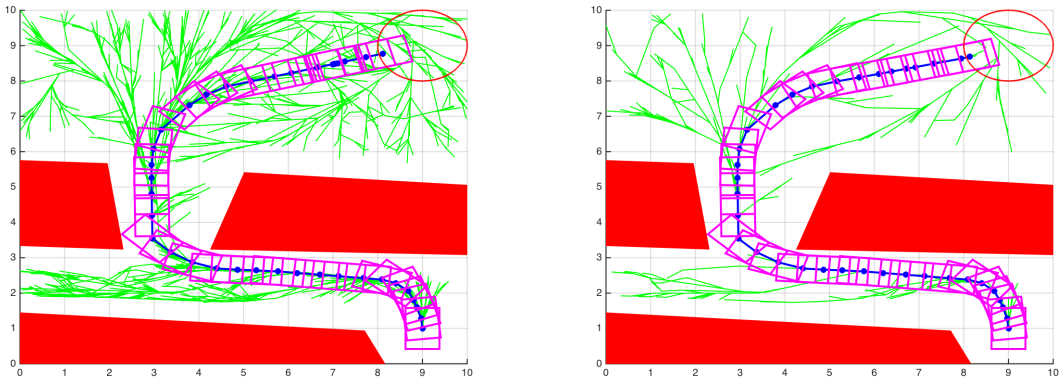
goal (see Figure 5.5). This shows that complex manuevers requires more number of nodes for the car-like robot to avoid obstacles and also maintain non-holonomic constraints. However, if the random generator favors the path planner search process then also the right non-holonomic path can be achieved with less number of nodes (see Figure 5.6).



(a) RRT* (b) RRT*FN-NH

Figure 5.5: Path Γ_{best} returned for Random Seed # 30, where car needed multiple manuevers to reach goal

Thus, this section shows that complex manuevers can be easily handled by combining instantaneous next non-holomic constrained step and making path planner to guide the robot to the goal. This indicates the simplicity and elegant nature of the



(a) RRT* (b) RRT*FN-NH

Figure 5.6: Path Γ_{best} returned for Random Seed # 60, where car needed multiple manuevers to reach goal

approach.

5.3 RRT*FN-NH on Google Maps

Although traditional environments provided some challenging task for path planner to find right manuevers or explore the environment, it did not however represent the real scenario. So in this section, a real road map scenario (without traffic) is considered as an environment for RRT*FN-NH to generate non-holonomic path for car-like robot. Two local environments are chosen that mostly represent the roads found in urban areas.

The map 1 (see Fig 5.7(a)) is chosen such that it mostly represents multiple turns at intersections to reach the goal. Such turns will often require slowing down car to a certain speed when nearing intersection and then take a smooth turn thus offering a necessary constraint commonly found in urban road driving which was missing in benchmark examples. These roads are nearly straight with slight curvature.

The map 2 (see Fig 5.8(a)) represents commonly a major junction found in cities where there is a circular obstacle (roundabouts) at the center where traffic from all

the roads merge into and separate out. Although, we have not taken into account the merging of traffic, maintaining nearly same speed near the junction may be critical to compete with fast moving traffic. Thus contrary to map 1 here the slowing down of vehicle may not be needed but rather nearly continuous speed motion is necessary.

The local maps were taken from Google static map API [125] at a constant zoom level of the Google map. The car had width 3 pixels, length 6 pixels and length between axle as 2 pixels.

5.3.1 Car speed profiles

Three speed profiles were tested: Slow, Medium and Fast. Table 5.7 represents the speed profile parameters. d_{step} is the maximum allowable one step motion that can be taken by the car-like robot. Note that the acceleration was kept constant for all speed profiles as a car would have a constant acceleration but moving with different speed ranges.

Table 5.4: Speed profile of car in pixels, pixels/sec, pixels/sec^2

Profile	**Slow**	**Medium**	**Fast**
d_{step}	0.75	1	1.25
v_{min}	0.1	0.5	0.1
v_{max}	1.5	2	4
a_{max}	0.1	0.1	0.1

Given some arbitrary random seed chosen, different profiles are used to test N (no of iterations) and M (max. number of nodes) required by RRT*FN-NH and also the quality of non-holonomic path using parameters L (Total path length), and T (total time taken by path) are described in following tests.

Slow speed

Table 5.5 presents the results obtained by running RRT*FN-NH on google maps 1 and 2. Given the speed range of minimum 0.1 pixels/sec and maximum 1.5 pixels/sec, the total time taken by the car to reach destination for Map 1 is 1027 secs and Map 2 is 399 secs. Note that Map 1 consist of many lanes and longer distance to travel compared to Map 2 and hence car takes longer time for Map 1.

Figure 5.7 shows the resulting non-holonomic path in Map 1 using slow speed profile. Also local enlarged areas are shown to see the quality of non-holonomic path being generated.

Table 5.5: Results in pixels and sec for slow speed profile

Parameters	**Map 1**	**Map 2**
Random seed	1000	100
N	28×10^3	4×10^3
M	8188	1330
L	628.1257	246.7283
T	1026.7629	398.5783

Medium speed

Table 5.6 mentions the results obtained by running RRT*FN-NH on google maps 1 and 2. Given the speed range of minimum 0.5 pixels/sec and maximum 2 pixels/sec, the total time taken by the car to reach destination for Map 1 is 877 secs ($<$ 1027 secs required for slow profile) and Map 2 is 417 sec ($>$ 399 secs required for slow profile). Thus for map 1 the car was able to move faster with the medium speed profile, however, for Map 2 the turning curve caused the problem for the car to possibly de-accelerate to maintain right curvature and was slow or nearly same as slow speed profile (also, note, the random seed for Map 2 was changed to test feasibility).

Table 5.6: Results in pixels and sec for medium speed profile

Parameters	**Map 1**	**Map 2**
Random seed	1000	1000
N	8×10^3	4×10^3
M	2666	1097
L	643.6551	248.3377
T	877.6788	416.9307

Fast speed

Table 5.6 mentions the results obtained by running RRT*FN-NH on google maps 1 and 2. Given the speed range of minimum 0.1 pixels/sec and maximum 4 pixels/sec, the total time taken by the car to reach destination for Map 1 is 540 secs ($<$ 877 secs required for medium profile) and Map 2 is 376 secs ($<$ 417 secs required for medium profile). Same thing is again observed for Map 1. However, for Map 2 as the min speed was set to 0.1 pixels/sec the path it generated around the curvature was nearly in the same nature as that of path in slow profile. This shows that min allowable speed needs to be selected appropriately to get the right results or expectations.

Table 5.7: Results in pixels and sec for fast speed profile

Parameters	**Map 1**	**Map 2**
Random seed	1000	123
N	32×10^3	4×10^3
M	8898	1078
L	648.6269	254.3213
T	540.9243	375.4331

5.3.2 Insights

Lets look at the quality of non-holonomic path found by RRT*-FN-NH for Map 1 in slow profile. For first intersection present in local area 1 clearly as seen from figure 5.7(b), the car slightly diverges from straight line its following to maintain

the curve. Also note if the two consecutive configurations are nearby implies that speed of the car had to be decreased and if its further implies that speed of car was increased. It seems that it almost maintains the speed while turning, but unfortunately does not correct the steering wheel back to straight position on time which makes the car move in a zig-zag way. This implies additional heuristics are needed to be enforced by the planner to achieve human-like turning movement of car.

Next in figure 5.7(c) represents a nearly straight road. As seen by increasing distance between neighbouring configurations the car picks up speed but soon realises it going to hit the other side of road and de-accelerates to make a small turn to keep going forward along the road. Here the interesting characteristic to note is that acceleration/de-acceleration are automatically represented by the non-holonomic path as given by generator. Note typically the road is assume to be single way. If double way needs to be enforced then that part of road has to be represented as obstacle space for planner. Such simple change can easily navigate the car straight along its own road.

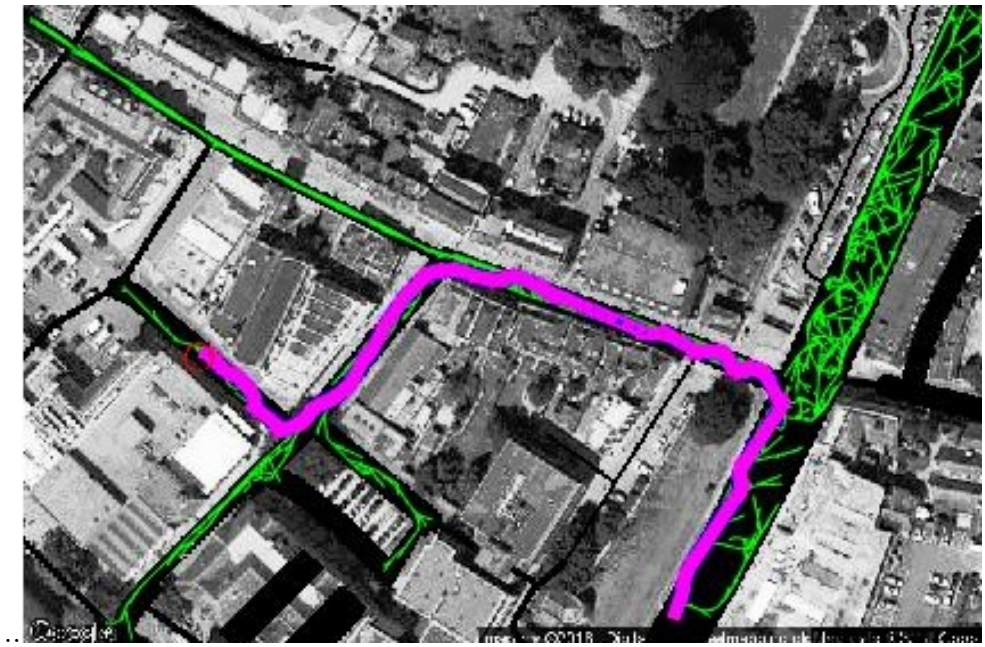
In figure 5.7(d) again the intersection is represented and as seen the car goes slight near the other side of road to make a turn at nearly constant speed. A smooth non-holonomic path was resulted based on the right positioning of car as decided by non-holonomic path planner. Note for local area 1 this was not observed as the start position of car was fixed thus showing the constraint behavior observed by non-holonomic motion.

Next in figure 5.7(e) the non-holomic path take more of a zig-zag to reach the final destination as the length of the road was more based on the computed positioning of the car. This is the result of simultaneously taking two intersections consecutively as speed of the car was attained that may make it difficult to turn. This leads us to have RRT*-FN-NH run with more iterations and more number of nodes.

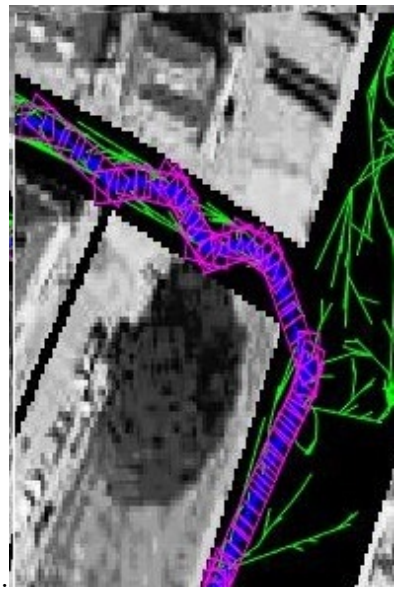
Now lets look at map 2 that has roundabout for slow speed profile. In figure 5.8(b) as seen the road is nearly narrow where only single car can pass by. In-spite of such tight narrow road, although with slight curvature, the car speed is nearly kept constant to navigate through narrow road. Also the non-holonomic path appears smooth. This reflects back to our claim that for two lane roads even if the road gets narrow RRT*FN-NH planner will be able to find a solution in reasonable number of iterations.

Figure 5.8(c) shows a drastic de-acceleration to make sure car does not hit the round-about. Also note the extent of turn causes it to go faster for remaining path that shows the efficacy of path planner for non-holonomic paths. Thus, path planning provides an obvious advantage in by placing the car at right position to favor further car motion. This is the key insight observed for map 1 as well as map 2.

While the path quality for other different speed profiles are shown in figure 5.9 and figure 5.10 respectively for meduim and fast profile, the key thing to note is with such constrained maps for RRT*FN-NH, it can be still easily find feasible paths. Thus, RRT*FN-NH seems to have less difficulty and can be easily applied on google maps to generate non-holonomic paths. Note here speed profile change still produces nearly same path lengths but with better total time performance at the cost of increased number of iterations need to be increased with increased speed profile. However, the memory needed stays nearly the same.



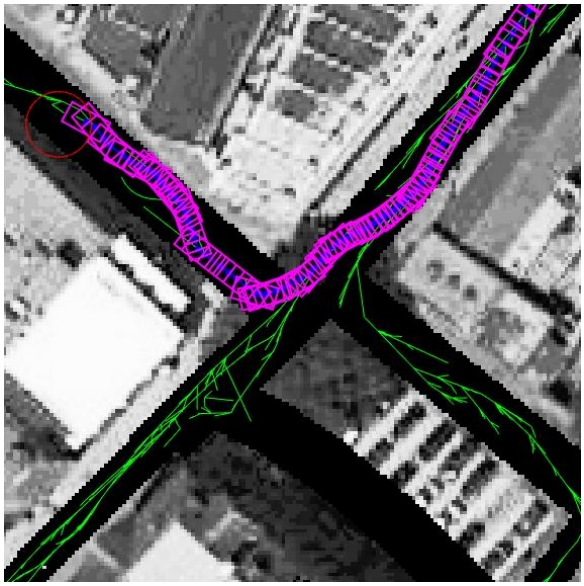
(a) Map 1



(b) Enlarged local area 1

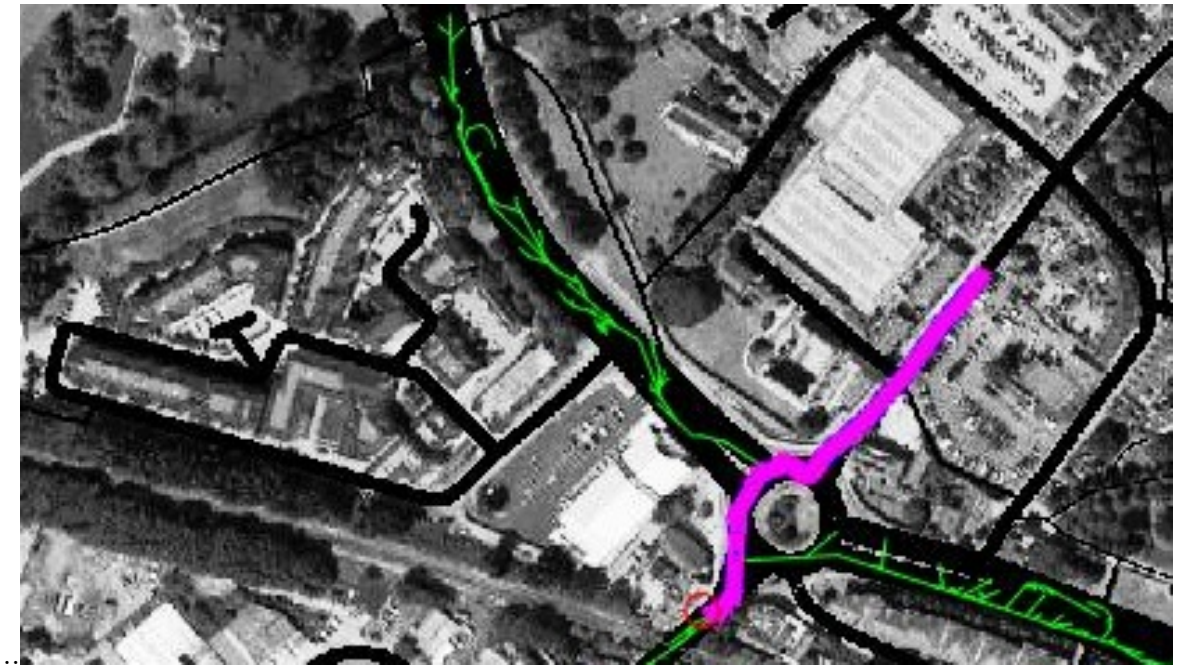
(c) Enlarged local area 2

(d) Enlarged local area 3

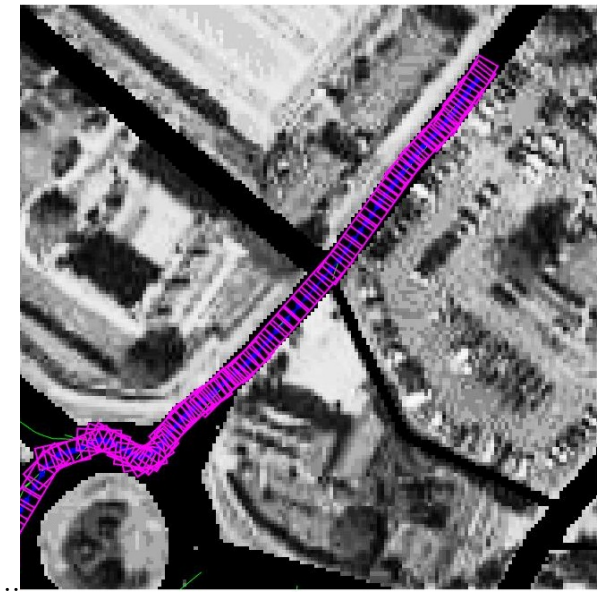


(e) Enlarged local area 4

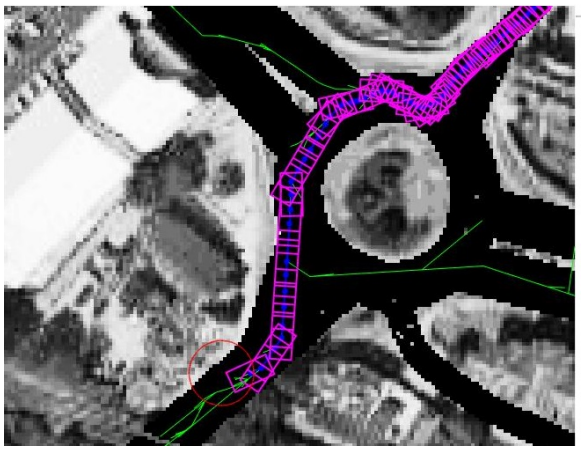
Figure 5.7: Path Γ_{best} returned for Map 1 in slow speed profile of car



(a) Map 2

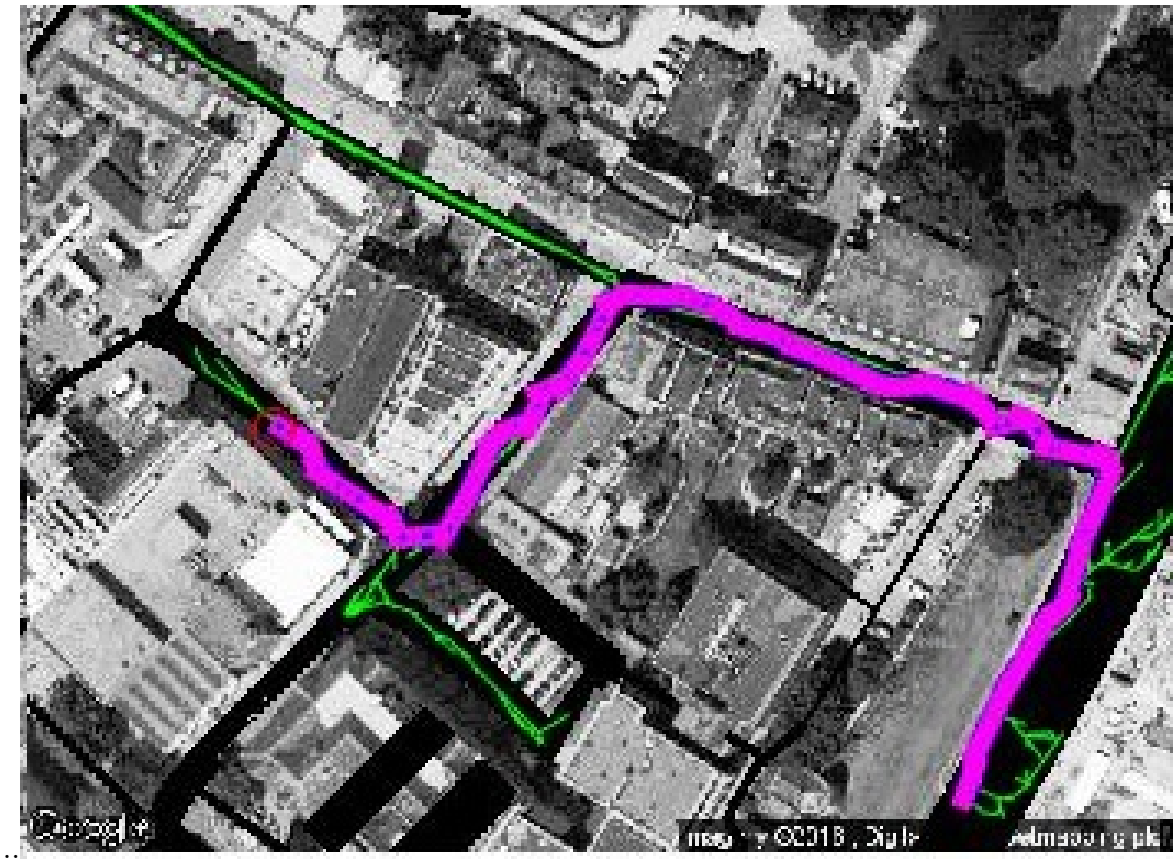


(b) Enlarged local area 1

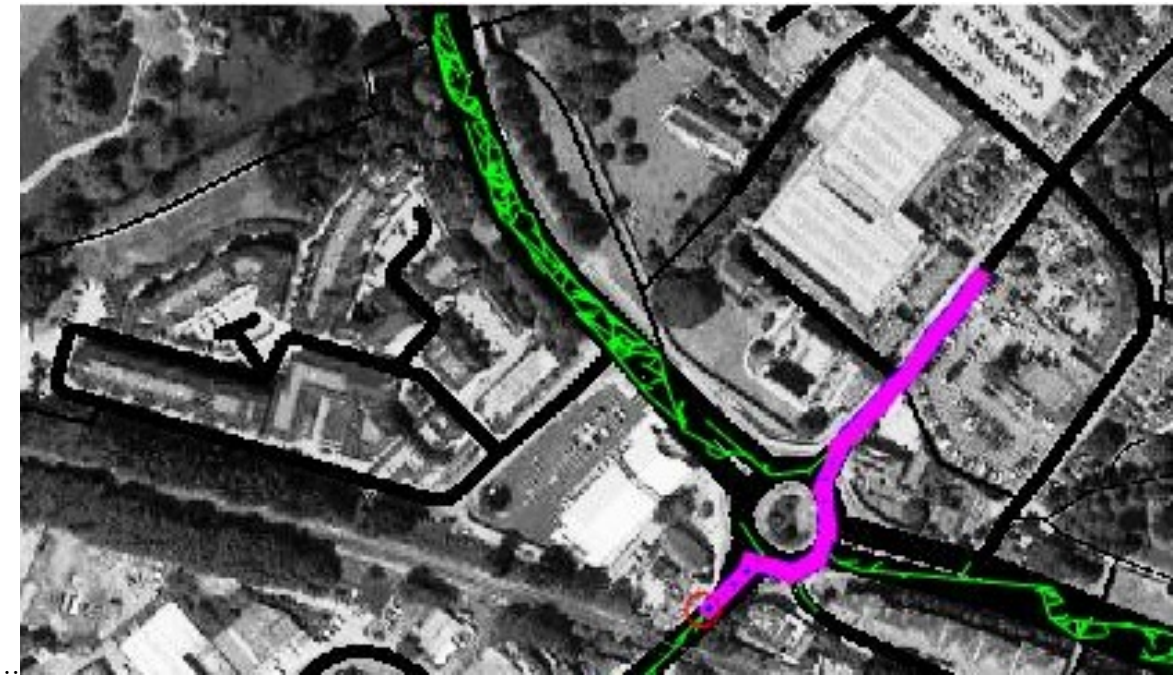


(c) Enlarged local area 2

Figure 5.8: Path Γ_{best} returned for Map 2 in slow speed profile of car



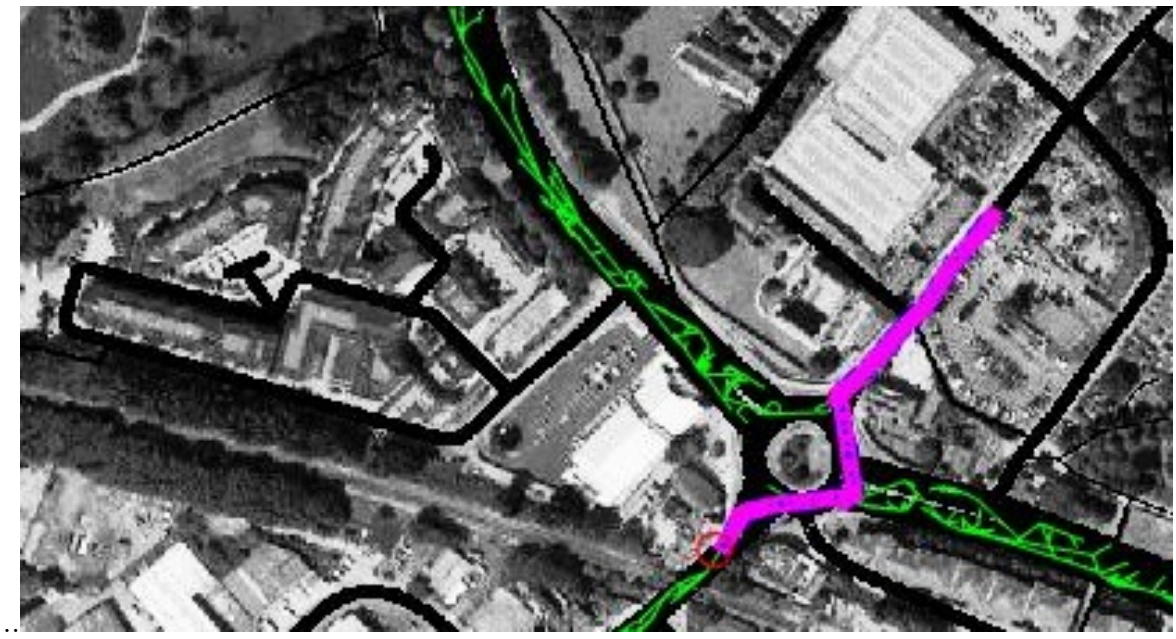
(a) Map 1



(b) Map 2

Figure 5.9: Path Γ_{best} returned for Map 1 and Map 2 in medium speed profile of car

(a) Map 1



(b) Map 2

Figure 5.10: Path Γ_{best} returned for Map 1 and Map 2 in medium speed profile of car

Chapter 6

Conclusion and Future Work

The ability for any mobile robot to navigate in its environment is a fundamental task. There are many literature (refer Chapter 2) already existing that provides ability for a mobile device to avoid dangerous situations such as collisions and unsafe conditions (temperature, radiation, exposure to weather, etc.). However, such mobile devices addressed in literature are not mostly for non-holonomic systems, like a car with steering mechanism and speed control (acceleration/stop pedestal).

There exists rich literature [128] of path planning of robots (not mostly for non-holonomic systems) that can now mostly be applied for non-holonomic systems with the approach presented in the thesis. Specifically, this thesis showed how a constrained planner RRT*FN can be applied for non-holonimic systems (a car-like robot) elegantly using the non-holonomic path generation incrementally step-by-step.

By using incremental small next step and the modified path planning algorithm RRT*FN-NH for robots with non-holonomic constraints, the thesis demonstrated experiments of car-like robot in different challenging static environments with additional constraint of having minimum velocity of car. The tests showed that to obtain feasible solution, the number of iterations required are in the magnitude of 10^3.

6.1 Achievements

This research aimed at addressing the path planning process for autonomous driving cars. The key novel points are mentioned below about the approach and outcome.

1. Utilizing the basic form of kinematic differential equations to compute "incremental next step" of non-holonomic systems (see Section 4.3.1).

2. This resulted in an instantaneous random path generation algorithm satisfying non-holonomic constraints (see Sections 4.3.2, and 5.1.1).

3. The randomness required by sampling based planners was achieved at the time of generating non-holonomic constraints. Thus no need of connecting two disconnected configurations using local planner that normally results in computationally expensive path that may also be infeasible (see Section 4.4.2).

4. Effectively combined the path planner capability of using randomness in exploring the configuration space of the robot while facilitating that feasible random path to grow more towards the goal (see Sections 4.4, 5.2, and 5.3).

5. A variant of RRT*FN was used as a demonstrative planner with the strategies as mentioned in this research for non-holonomic robots (see Chapter 4).

6. Experiments shows that a vast literature of path planning for various robots can now be easily extended for non-holonomic robots (see Sections 5.2, and 5.3).

7. Further to prove practicality of path planning process for car-like robots, the road-like environments were also tested at different speed profiles (see Section 5.3).

8. Non-holonomic path generator being instantaneous can be easily scaled for dynamic environments with unforeseen changes.

6.2 Summary

Chapter 1 introduces the basic tools for autonomous navigation of robot commonly used in literature that can not be applied efficiently for non-holonomic systems. Further this chapter discusses the necessity of a new approach that would facilitate most of the already existing literature to be applied for non-holonomic systems. Further it also tries to scale up for environments that are dynamic in nature. To limit the effects of uncertainity in dynamic environments the major assumption this thesis makes is that all the cars on the road are autonomous and not manually driven.

Chapter 2 refers to literature survey extensively. The references for existing non-holonomic path planners indicate that mathemathical computation of such paths can be time-consuming especially if the entire non-holonomic path is generated for sub-goals and then collision check is made. Not only its difficult to modify the existing non-holonomic paths based on collision information but also such generators can not be effectively applied to standard planners like RRT. Further, Chapter 2 indicates existing literature on safe path planning, factors that affect path planning and planning using fuzzy logic. Such literature already existing can be applied to facilitate non-holonomic path planning by elegant combination of incremental non-holonomic step generation and path planners solution to reach goal.

Chapter 3 discusses the basic framework for autonomous car driving having one of its critical component as non-holonomic path planning. Further it indicates to handle autonomous car driving in presence of obstacles, ideally the non-holonomic path generator should be computationally inexpensive and moreover instantaneous to rapidly change the found non-holonomic path in collision. Further it argues that the generator should easily integrate with incremental sampling based planners based on their methodology. Then it discusses the achievements and contributions

made by the thesis.

Chapter 4 discusses the algorithms used to generate non-holonomic paths and effectively augments RRT*FN planner to find collision-free non-holonomic path from start to end. It provides the configuration of the car and also the control vectors (speed, steering angle) as inputs at a time that needs to actuated by the car to achieve that non-holonomic path for reaching goal. After briefly introducing RRT*FN, Chapter 4 provides the modified version of RRT*FN called RRT*FN-NH to plan collision-free paths for car-like robots. RRT*FN was chosen so that RRT is constrained to work in a limited amount of RAM as found in most embedded systems that have to be deployed with the car.

Chapter 5 finally examines RRT*FN-NH in traditional benchmark static environments and produces a validity that RRT*FN-NH finds most of the time the feasible non-holonomic collision-free path if it exists. Multiple tests are done and quality of paths are discussed for this benchmark simulations with a variety including narrow passage problem and requiring multiple maneuvers. Further, this thesis tries to apply RRT*FN-NH on Google Map roads so that the quality of path can be tested for real-scenarios. Elegantly applied for google maps the tests show similar results as obtained for benchmark environments with a cost of increased iterations as the speed profile of car increases. Such number of iterations usually goes in the multiples of 10k based on complexity of environment.

6.3 Possible future advancements of our work

While the purpose of this thesis was to facilitate the existing algorithms for non-holonomic path planning, it tested for a RRT based algorthm. While RRT and PRM have been dominantly used in the field, most of the algorithms that are based on RRT should be able to use the thesis directly to facilitate non-holonomic path

planning. However, the strengths of different approaches used in path planning like PRM needs to be tested with this approach especially if PRM relies on some local planner to connect between the sub-goals.

The algorithm shown here can scale up for dynamic environments, however tests need to be made to indicate the real-time performance achieved by this method especially being incremental in nature. While the roads (without traffic) based on google maps were tested, the need of central planner to plan all the autonomous car motion simultaneously is needed.

6.4 Autonomous robot navigation and obstacle perception

As a result, mobile robots capable of moving in a dynamical and uncertain environment is an important issue in real-world applications. The problem that how to find an optimal real-time collision-free path with a limited sensing range in the presence of dynamically moving objects is arising naturally. The optimal solution should take motion constraints into consideration (including boundary conditions and kinematic constraint), explicitly handle dynamically moving objects, and be analytical.

Based on the previous mentioned methods (refer Chapter 2), it is evident that this technology is well promising for the future. While the human-machine interface is not yet at a transparent level, the degree of autonomy available after a machine has been program is now approaching that once considered purely science fiction. Things that could be done in the future, related to the previous algorithms, are for example to optimize the current techniques using more state-of-the-art methods, testing the navigation algorithms to have a measure of its performance in more complex and realistic scenarios, or even considering for instance that the knowledge about the future behaviour of a robot is less reliable in the distant future, so it could

be interesting to monotonically decrease the influence of the obstacles with respect to time.

One of the ways in which autonomous robot perception could be improved, especially for safe navigation of urban environments, is by object perception. For the majority of this review, all obstacles have been treated as essentially equal (i.e. rough patches to be avoided). However, as the reader may have suspected at some point, not all obstacles are equal. In fact, some obstacles present quite opposite problems to the optimization routine. A patch of ground for example is a "rough patch" that, although preferably avoided, could in theory be traversed if the cost-to-go function (i.e. "roughness-to-go") were to deem a trajectory through that path to be necessary. However, in other situations, especially in urban environments, "traversing" an obstacles is absolutely not an option. One obvious case of an obstacle that may not under any circumstances be traversed is a pedestrian. Pedestrians must be avoided at all costs, including the cost of potentially never reaching the end destination or (from a programmatic point of view) never being able to calculate possible trajectories leading to the destination. This would be the case in a hypothetical situation where a never-ending stream of pedestrians is crossing a street. One of the recent entries into the US Department of Defense – sponsored annual competitions for autonomous robot navigation was the Stanford car dubbed "Junior" (2013). As reported [36], "Junior" was able to very accurately tell the difference between people, cars, animals, signs, and roads. This was largely thanks to a novel laser and sensor calibration scheme that involved a great deal of machine learning in the original situation in the navigation environment. This was an example of a case for which existing information about the visual environment was used to estimate or interpolate parameters for that environment at later times points, by looking primarily at the aspects of the environment that change. "Junior" autonomous vehicle [43] is able to turn a density "cloud" of obstacle perception (left) into a refined, crisp image (right) with sufficient details to make out the identity of

different obstacles, provided that an initial sensor calibration is performed. Once this calibration has been performed at the beginning of the vehicle's trip or trajectory, it does not need to be repeated until the vehicle is transported by carrier and placed in a new environment. The refined image allows for certain obstacles (such as pedestrians in a crowded urban environment) to be identified and avoided at all costs, in favor of traversing less important obstacles if need be (such as curbs, stairs, etc.) Another means by which machine vision is becoming more sophisticated in the perception of objects, is in a sense by moving in the opposite direction to how Stanford's "Junior" progressed from earlier autonomous vehicles. Whereas "Junior" was able to use more specific, fine-grained features of the environment, [37] is able to detect whether it is nighttime or daytime outside, and base interpretation of features, obstacles density paths, and corresponding trajectories on this information. For example, an accurate assessment of the position of the sun (as well as other light sources, during the night) allows for the detection of shadows with greater accuracy. Evidently, shadows can be traversed provided there are not hidden obstacles. To this author's knowledge, no methods that delve into predicting obstacles hidden in shadows have been developed to date. As one can see, the richness of the visual information is increasingly being taken advantage of by autonomous vehicles, thanks in large part to these vehicles' increasingly powerful artificial intelligences. As a result, the rate of acquisition of this data is becoming increasingly "thirsty", and autonomous vehicles are traveling faster and faster. "Junior" [45], for example, can travel up to 35 miles per hours in a crowded urban environment (slowing or stopping where necessary, of course, to avoid pedestrians and other key obstacles). However, the faster that autonomous vehicles go, the more error is introduced to their sensors thanks simply to some basic principles of optics. For example, bending of light occurs even at moderate (highway) speed. As a results, in [45] the authors present a sensor capable of recording single-photon time of flight information based on correlation with other photons.

6.5 Applications of autonomously-navigating car-like robots

With all the above literature review and discussed devoted to sensors, techniques, problem types, and optimization algorithms for autonomous vehicle perception of obstacles and navigation by optimization of trajectories around obstacles, little has been said thus far about the actual applications of autonomously-navigating robots. What is the interest in, and what are therefore some possible applications of, these increasingly intelligent and self-aware road and off-road travelers? Below is a discussion describing a variety of different existing and emerging applications of such robots. An obvious, although far from universally accepted or even much considered, application for autonomous vehicles is for the transportation of people. Some autonomous vehicles already transport people. However, there is potential for autonomous personal and public vehicles to largely replace the manually-operated equivalents of today and yesterday. Because there is not yet an "internet of things" (IoT), a term used below that refers to the potential future in which all objects are connected to the internet via tiny wireless sensors, directions and features of roads and in particular traffic conditions (for example detours due to construction) are often not updated into mobile road navigation apps on many drivers' cell phones or GPS units. Therefore, at least for the foreseeable future (until there is a veritable IoT or at least higher-integrity, more reliable set of traffic / road conditions comprehensively and instantaneously updated in real time – no 15-minute delay allowable) autonomous vehicles would have to be able to read road signs just as any human driver would. A recent study [46] provides a method for achieving rapid, in-transit machine-vision sign reading. However, the authors performed their training as well as validation under conditions of fair lighting, unlike what may often be the case even with headlights illuminated. One possible applications autonomously-navigating robots is as traffic-monitoring "drone" vehicles. These vehicles would

patrol highways, and collect information about traffic density, and other environmental factors such as temperature, humidity, and surface conditions (e.g. precipitation accumulation). These drone vehicles would be networked to an information hub, either a higher-level computer or a human operator and traffic surveyor. Relaying information to a central information hub would enable high-resolution, real-time information about traffic to be distributed to passengers. This would be accomplished in multiple possible ways, for example by allowing the information to be accessible to mobile phone apps. One recently-proposed means of distributing the information gathered by robotic drone vehicles is by using cloud computing [47]. In this model, cloud computing would also be used to allow communication (and thereby formation of consensus data interpretation and analysis) between different robots. Cloud computing is fast, easily accessible, and cheap. In order to address issues of security related to autonomous vehicles, especially those with a multitude of sensors containing possible sensitive information (but generally without the size, complexity, or infrastructure to effectively protect against virus or rogue cyber intrusions), some sophisticated theoretical as wel as practical design steps have already been taken. As explained in [48], while denial of service (DoS) style attacks (or sophisticated cyber-attacks originating from multiple points simultaneously) on static networks remains a problem, the technological development of wireless sensors has in many cases led to the adoption of a mobile, robotic platform. In parallel, the possibility of DoS by a mobile, malignant node arises. This article is the first to describe this problem explicitly, describing the unique advantages to DoS agents that mobility brings, and to propose a solution for overcoming these new advantages. The article [48] mentions several means by which malicious, mobile nodes could disrupt wireless sensor networks (WSNs) that would be impossible without mobility. A mobile node, if equipped with robotic arms, could move up to a node in the WSN, pick it up, and move it. This disruption in position would throw flags in the WSN security routine, and may even automatically cut out the node from the WSN, leaving it easy pretty for the malicious node. Malignant, mobile nodes could also move to as

many different positions as possible, searching for weaknesses (i.e. spots where their positions would be more likely to be accepted as characteristic of a "safe" node), jamming communications, and moving nodes. In addition, the diversification of attack paths would make traceback impossible, without a prior assumption that the nodes were both mobile and hostile. Although purely theoretical, the article does propose several strategies for combating the threat of a mobile, malignant node or swarm of nodes. The article focuses on the case wherein the WSN is static, and only the malignant nodes are mobile. A straightforward means of detecting a mobile malignant node would be to keep a list of neighbors, leveraging the fact that the WSN is static. However, this would place severe constraints on the topology of the WSN, as a pre-defined set of neighbors would have to be supplied to the base node as a unique key, for each node. The authors suggest using an adjustable threshold maximum time limit between signals from a neighboring node, with the assumption that, beyond this threshold, the neighbor would be considered as potentially mobile (and therefore malignant). The more nodes flag the same outside node as malignant according to this criterion, the more likely is the base node to pass a judgment of "malignant".

Bibliography

[1] C. L. Hwang and C. Y. Shih, "A distributed active-vision network-space approach for the navigation of a car-like wheeled robot," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 3, pp. 846–855, March 2009.

[2] C. L. Hwang and L. J. Chang, "Internet-based smart-space navigation of a car-like wheeled robot using fuzzy-neural adaptive control," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 5, pp. 1271–1284, Oct 2008.

[3] D. K. Grady, M. Moll, C. Hegde, A. C. Sankaranarayanan, R. G. Baraniuk, and L. E. Kavraki, "Multi-objective sensor-based replanning for a car-like robot," in *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Nov 2012, pp. 1–6.

[4] V. Delsart and T. Fraichard, "Navigating dynamic environments using trajectory deformation," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2008, pp. 226–233.

[5] C. Fulgenzi, A. Spalanzani, and C. Laugier, "Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 1610–1616.

[6] C. Pradalier, J. Hermosillo, C. Koike, C. Braillon, P. Bessière, and C. Laugier, "The cycab: a car-like robot navigating autonomously

and safely among pedestrians," *Robotics and Autonomous Systems*, vol. 50, no. 1, pp. 51 – 67, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889004001848

[7] J. J. Park and B. Kuipers, "Feedback motion planning via non-holonomic rrt* for mobile robots," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 4035–4040.

[8] L. Palmieri, T. P. Kucner, M. Magnusson, A. J. Lilienthal, and K. O. Arras, "Kinodynamic motion planning on gaussian mixture fields," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 6176–6181.

[9] L. Palmieri, S. Koenig, and K. O. Arras, "Rrt-based nonholonomic motion planning using any-angle path biasing," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 2775–2781.

[10] J. J. Craig, *Introduction to Robotics: Mechanics and Control*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[11] M. T. Mason, *Mechanics of Robotic Manipulation*. Cambridge, MA: MIT Press, Aug. 2001.

[12] J. D. Cohen, M. C. Lin, D. Manocha, and M. Ponamgi, "I-collide: An interactive and exact collision detection system for large-scale environments," in *Proc. of ACM Interactive 3D Graphics Conf.*, 1995, pp. 189–196.

[13] M. C. Lin and S. Gottschalk, "Collision detection between geometric models: A survey," in *Proc. of IMA Conf. on Mathematics of Surfaces*, 1998, pp. 37–56.

[14] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the Association for Computing Machinery (ACM)*, vol. 22, no. 10, pp. 560–570, 1979.

[15] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research (IJRR)*, vol. 5, no. 1, pp. 90–98, 1986.

[16] J. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[17] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," in *IEEE Trans. on Robotics and Automation*, vol. 12, no. 4, 1996, pp. 566–580.

[18] S. M. LaValle and J. J. K. Jr., "Randomized kinodynamic planning," in *IEEE International Conference on Robotics and Automation*, 1999, pp. 473–479.

[19] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, May 2006.

[20] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "OBPRM: an obstacle-based PRM for 3d workspaces," in *WAFR '98: Proceedings of the third workshop on the algorithmic foundations of robotics on Robotics : the algorithmic perspective*. Natick, MA, USA: A. K. Peters, Ltd., 1998, pp. 155–168.

[21] M. H. Overmars, "The Gaussian sampling strategy for probabilistic roadmap planners," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 1999, pp. 1018–1023.

[22] P. Leven and S. Hutchinson, "Using manipulability to bias sampling during the construction of probabilistic roadmaps," *IEEE Trans. on Robotics and Automation*, vol. 19, no. 6, pp. 1020–1026, Dec. 2003.

[23] Y. Yang and O. Brock, "Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation in dynamic environments." in *Robotics Science and Systems II*. The MIT Press, 2006.

[24] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: the path-velocity decomposition," *Int. J. Rob. Res.*, vol. 5, no. 3, pp. 72–89, 1986.

[25] M. Zucker, J. Kuffner, and M. Branicky, "Multipartite rrts for rapid replanning in dynamic environments," in *IEEE Intl. Conf. on Robotics and Automation*, 2007, pp. 1603–1609.

[26] J. Vannoy and J. Xiao, "Real-time Adaptive Motion Planning (RAMP) of mobile manipulators in dynamic environments with unforeseen changes," in *IEEE Trans. on Robotics*, vol. 24(5), 2008, pp. 1199–1212.

[27] S. Karaman and E. Frazzoli, "Sampling-based optimal motion planning for non-holonomic dynamical systems," in *IEEE International Conference on Robotics and Automation (ICRA),*, May 2013, pp. 5041–5047.

[28] S. Petti and T. Fraichard, "Safe navigation of a car-like robot in a dynamic environment," in *Proceedings of the European Conference on Mobile Robots*, 2005.

[29] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[30] V. V. Dixit, S. Chand, and D. J. Nair, "Autonomous vehicles: Disengagements, accidents and reaction times," *PLOS ONE*, vol. 11, no. 12, pp. 1–14, 12 2016. [Online]. Available: https://doi.org/10.1371/journal.pone.0168054

[31] O. Adiyatov and H. Varol, "Rapidly-exploring random tree based memory efficient motion planning," in *IEEE International Conference on Mechatronics and Automation (ICMA)*, 2013, pp. 354–359.

[32] J. Ward and J. Katupitiya, "Free space mapping and motion planning in configuration space for mobile manipulators," in *IEEE Intl. Conf. on Robotics and Automation*, 2007, pp. 4981–4986.

[33] P. Leven and S. Hutchinson, "A framework for real-time path planning in changing environments," *Intl. J. of Robotics Research*, vol. 21, pp. 999–1030, 2002.

[34] V. Govea, D. Alejandro, F. Large, T. Fraichard, and C. Laugier, "High-speed autonomous navigation with motion prediction for unknown moving obstacles," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Oct. 2004, pp. 82–87.

[35] A. Kushleyev and M. Likhachev, "Time-bounded lattice for efficient planning in dynamic environments," in *IEEE Intl. Conf. on Robotics and Automation*, May 2009, pp. 1662–1668.

[36] J. van den Berg, D. Ferguson, and J. Kuffner, "Anytime path planning and replanning in dynamic environments," in *IEEE Intl. Conf. on Robotics and Automation*, May 2006, pp. 2366–2371.

[37] N. Du Toit and J. Burdick, "Robot motion planning in dynamic, uncertain environments," *Robotics, IEEE Transactions on*, vol. 28, no. 1, pp. 101 –115, Feb. 2012.

[38] B. Jones and I. Walker, "Kinematics for multisection continuum robots," in *IEEE Trans. Robot.*, 2006.

[39] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt*," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011, pp. 1478–1483.

[40] P. Muntean, "Mobile robot navigation on partially known maps using a fast a star algorithm version," *CoRR*, vol. abs/1604.08708, 2016. [Online]. Available: http://arxiv.org/abs/1604.08708

[41] M. Otte and E. Frazzoli, "Rrtx," *Int. J. Rob. Res.*, vol. 35, no. 7, pp. 797–822, Jun. 2016. [Online]. Available: https://doi.org/10.1177/0278364915594679

[42] O. Adiyatov and H. Varol, "A novel rrt*-based algorithm for motion planning in dynamic environments," in *2017 IEEE International Conference on Mechatronics and Automation, ICMA 2017*. Institute of Electrical and Electronics Engineers Inc., 8 2017, pp. 1416–1421.

[43] L. Huazhong, L. Yongsheng, W. Meini, and D. Tangren, "Design and implementation of improved rrt algorithm for collision free motion planning of high-dimensional robot in complex environment," in *Proceedings of 2012 2nd International Conference on Computer Science and Network Technology*, Dec 2012, pp. 1391–1397.

[44] I. Noreen, A. Khan, and Z. Habib, "Optimal path planning using rrt* based approaches: A survey and future directions," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 11, 2016. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2016.071114

[45] R. Pepy, A. Lambert, and H. Mounier, "Path planning using a dynamic vehicle model," in *Information and Communication Technologies, 2006. ICTTA '06. 2nd*, vol. 1, 2006, pp. 781–786.

[46] S. Garrido, D. Blanco, L. Moreno, and F. Martin, "Improving rrt motion trajectories using vfm," in *2009 IEEE International Conference on Mechatronics*, April 2009, pp. 1–6.

[47] S. Garrido, L. Moreno, D. Blanco, and F. Martin, "Smooth path planning for non-holonomic robots using fast marching," in *2009 IEEE International Conference on Mechatronics*, April 2009, pp. 1–6.

[48] S. Balakirsky and D. Dimitrov, "Single-query, bi-directional, lazy roadmap planner applied to car-like robots," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 5015–5020.

[49] ——, "Single-query, bi-directional, lazy roadmap planner applied to car-like robots," in *IEEE International Conference on Robotics and Automation*, May 2010, pp. 5015–5020.

[50] S. Rodriguez, X. Tang, J.-M. Lien, and N. Amato, "An obstacle-based rapidly-exploring random tree," in *Proceedings IEEE International Conference on Robotics and Automation*, May 2006, pp. 895–900.

[51] R. Pepy and A. Lambert, "Safe path planning in an uncertain-configuration space using rrt," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2006, pp. 5376–5381.

[52] J. Kim, J. M. Esposito, and V. Kumar, "An rrt-based algorithm for testing and validating multirobot controllers," in *Robotics: Science and Systems I*, Cambridge, Massachusetts, June 8-11 2005.

[53] A. Tahirovic and G. Magnani, "A roughness-based rrt for mobile robot navigation planning," in *In IFAC World Congress*, vol. 18, 2001, pp. 5944–5949.

[54] J. Ziegler and M. Werling, "Navigating car-like robots in unstructured environments using an obstacle sensitive cost function," in *2008 IEEE Intelligent Vehicles Symposium*, June 2008, pp. 787–791.

[55] J. M. Phillips, N. Bedrossian, and L. E. Kavraki, "Guided expansive spaces trees: a search strategy for motion- and cost-constrained state spaces," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 4, April 2004, pp. 3968–3973 Vol.4.

[56] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Rob. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011. [Online]. Available: http://dx.doi.org/10.1177/0278364911406761

[57] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How, "Motion planning for urban driving using rrt," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2008, pp. 1681–1686.

[58] S. Petti and T. Fraichard, "Safe Navigation of a Car-Like Robot in a Dynamic Environment," in *Proc. of the European Conf. on Mobile Robots*, Ancona (IT), France, Sep. 2005, voir basilic : http://emotion.inrialpes.fr/bibemotion/2005/PF05a/ address: Ancona (IT). [Online]. Available: https://hal.inria.fr/inria-00182047

[59] ——, "Safe navigation of a car-like robot in a dynamic environment," in *Proc. of the European Conf. on Mobile Robots*, Ancona (IT), France, Sep 2005, voir basilic : http://emotion.inrialpes.fr/bibemotion/2005/PF05a/ address: Ancona (IT). [Online]. Available: https://hal.inria.fr/inria-00182047

[60] G. Oriolo, M. Vendittelli, L. Freda, and G. Troso, "The srt method: Randomized strategies for exploration," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 5. IEEE, 2004, pp. 4688–4694.

[61] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier, "Probabilistic navigation in dynamic environment using rapidly-exploring random trees and gaussian processes," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2008, pp. 1056–1062.

[62] Y. Kuwata, G. Fiore, J. Teo, E. Frazzoli, and J. How, "Motion planning for urban driving using rrt," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2008, pp. 1681–1686.

[63] A. Yershova, L. Jaillet, T. Simeon, and S. LaValle, "Dynamic-domain rrts: Efficient exploration by controlling the sampling domain," in *IEEE International Conference on Robotics and Automation*, April 2005, pp. 3856–3861.

[64] L. Jaillet, A. Yershova, S. La Valle, and T. Simeon, "Adaptive tuning of the sampling domain for dynamic-domain rrts," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Aug 2005, pp. 2851–2856.

[65] Rodriguez, X. Tang, J.-M. Lien, and N. M. Amato, "An obstacle-based rapidly-exploring random tree," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, May 2006, pp. 895–900.

[66] B. Burns and O. Brock, "Single-query motion planning with utility-guided random trees," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 3307–3312.

[67] A. Yershova, L. Jaillet, T. Simeon, and S. M. LaValle, "Dynamic-domain rrts: Efficient exploration by controlling the sampling domain," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, April 2005, pp. 3856–3861.

[68] L. Jaillet, A. Yershova, S. M. L. Valle, and T. Simeon, "Adaptive tuning of the sampling domain for dynamic-domain rrts," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Aug 2005, pp. 2851–2856.

[69] L. Jaillet, J. Cortes, and T. Simeon, "Transition-based rrt for path planning in continuous cost spaces," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2008, pp. 2145–2150.

[70] D. Devaurs, T. Siméon, and J. Cortés, "Parallelizing rrt on large-scale distributed-memory architectures," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 571–579, April 2013.

[71] C. H. Esteban, C. Hernandez, and E. F. Schmitt, "Multi-stereo 3d object reconstruction," in *3D Data Processing Visualization and Transmission, 2002*, 2002, pp. 159–166.

[72] M. Han and T. Kanade, "Creating 3d models with uncalibrated cameras," in *proceeding of IEEE Computer Society Workshop on the Application of Computer Vision (WACV2000)*, Dec. 2000.

[73] N. Bellotto and H. Hu, "Multisensor-based human detection and tracking for mobile service robots," *IEEE Trans. on Systems, Man, and Cybernetics – Part B*, vol. 39, no. 1, pp. 167–181, 2009.

[74] D. Gavrila, "The visual analysis of human movement: A survey," *Computer Vision and Image Understanding*, vol. 73, pp. 82–98, 1999.

[75] D. Bradley, R. Unnikrishnan, and J. A. Bagnell, "Vegetation detection for driving in complex environments," in *IEEE Intl. Conf. on Robotics and Automation*, April 2007.

[76] A. Murarka, M. Sridharan, and B. Kuipers, "Detecting obstacles and drop-offs using stereo and motion cues for safe local motion," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2008, pp. 702–708.

[77] H. Schneiderman and T. Kanade, "Object detection using the statistics of parts," *Intl. Journal of Computer Vision*, vol. 56, pp. 151–177, 2004.

[78] T. Pham and A. Smeulders, "Object recognition with uncertain geometry and uncertain part detection," *Computer Vision and Image Understanding*, vol. 99, p. 258, 2005.

[79] P. Withagen, K. Schutte, and F. Groen, "Object detection and tracking using a likelihood based approach," in *Proc. IEEE Int. Conf. on Image Processing*, 2003.

[80] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 3, no. 1, pp. 1–130, 2009.

[81] Y. Kodratoff and S. Moscatelli, "Machine learning for object recognition and scene analysis," *International Journal of Pattern Recognition and AI*, vol. 8, pp. 259–304, 1994.

[82] C. C. Chang and K.-T. Song, "Environment prediction for a mobile robot in a dynamic environment," *IEEE Trans. on Robotics and Automation*, vol. 13, no. 6, pp. 862–872, Dec. 1997.

[83] Y. S. Nam, B. H. Lee, and M. S. Kim, "View-time based moving obstacle avoidance using stochastic prediction of obstacle motion," in *IEEE Intl. Conf. on Robotics and Automation*, 1996, pp. 1081–1086.

[84] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," in *Intl. J. of Robotics Research*, vol. 17, 1998, pp. 760–772.

[85] F. Large, S. Sckhavat, Z. Shiller, and C. Laugier, "Using non-linear velocity obstacles to plan motions in a dynamic environment." in *IEEE Intl. Conf. on Control, Automation, Robotics and Vision (ICARCV)*, 2002, pp. 734–739.

[86] A. Elnagar and K. Gupta, "Motion prediction of moving objects based on autoregressive model," *IEEE Trans. on Systems, Man, and Cybernetics, Part A*, vol. 28, no. 6, pp. 803–810, 1998.

[87] J. Vannoy and J. Xiao, "Real-time motion planning of multiple mobile manipulators with a common task objective in shared work environments," in *IEEE Intl. Conf. on Robotics and Automation*, April 2007, pp. 20–26.

[88] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun, "Learning Motion Patterns of People for Compliant Robot Motion," *Intl. J. of Robotics Research*, vol. 24, no. 1, pp. 31–48, 2005.

[89] Z. Chen, D. C. K. Ngai, and N. H. C. Yung, "Behavior prediction based on obstacle motion patterns in dynamically changing environments," in *Proceedings of the 2008 IEEE/WIC/ACM Intl. Conf. on Intelligent Agent Technology*, 2008, pp. 132–135.

[90] A. Ess, B. Leibe, K. Schindler, and L. V. Gool, "Moving obstacle detection in highly dynamic scenes," in *IEEE Intl. Conf. on Robotics and Automation*, May 2009, pp. 56–63.

[91] G. Gallagher, S. S. Srinivasa, J. A. Bagnell, and D. Ferguson, "Gatmo: a generalized approach to tracking movable objects," in *IEEE Intl. Conf. on Robotics and Automation*, May 2009, pp. 2043–2048.

[92] A. Elnagar and A. Hussein, "An adaptive motion prediction model for trajectory planner systems," in *Intl. Conf. on Robotics and Automation*, Sep. 2003, pp. 2442–2447.

[93] V. Govea, D. Alejandro, F. Large, T. Fraichard, and C. Laugier, "Moving obstacles' motion prediction for autonomous navigation," in *Int. Conf. on Control, Automation, Robotics and Vision*, Dec. 2004.

[94] A. F. Foka and P. E. Trahanias, "Predictive autonomous robot navigation," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS*, 2002, pp. 490–495.

[95] W. Dixon, I. Walker, and D. Dawson, "Fault detection for wheeled mobile robots with parametric uncertainty," in *Advanced Intelligent Mechatronics, 2001. Proceedings. 2001 IEEE/ASME International Conference on*, vol. 2, 2001, pp. 1245 –1250 vol.2.

[96] R. W. Brockett, *Asymptotic Stability and Feedback Stabilization.* Boston: Birkhauser, 1983, pp. 181–191.

[97] A. Widyotriatmo, A. Pamosoaji, and K.-S. Hong, "Robust configuration control of a mobile robot with uncertainties," in *Control Conference (ASCC), 2011 8th Asian*, May 2011, pp. 1036 –1041.

[98] N. Roy, W. Burgard, D. Fox, and S. Thrun, "Coastal navigation-mobile robot navigation with uncertainty in dynamic environments," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 1, 1999, pp. 35–40.

[99] P. Missiuro and N. Roy, "Adapting probabilistic roadmaps to handle uncertain maps," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, May 2006, pp. 1261 –1267.

[100] T. Fraichard and R. Mermond, "Path planning with uncertainty for car-like robots," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 1, May 1998, pp. 27 –32 vol.1.

[101] S. LaValle and R. Sharma, "Robot motion planning in a changing, partially predictable environment," in *Intelligent Control, 1994., Proceedings of the 1994 IEEE International Symposium on*, Aug. 1994, pp. 261 –266.

[102] L. Page and A. Sanderson, "Robot motion planning for sensor-based control with uncertainties," in *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, vol. 2, May 1995, pp. 1333 –1340 vol.2.

[103] H. Kurniawati, D. Yanzhu, D. Hsu, and S. L. Wee, "Motion planning under uncertainty for robotic tasks with long time horizons," *The International Journal of Robotics Research*, vol. 30, no. 3, pp. 308–323, 2011.

[104] K. Rebai, O. Azouaoui, M. Benmami, and A. Larabi, "Car-like robot navigation at high speed," in *2007 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec 2007, pp. 2053–2057.

[105] Q. Yuan, J. Y. Lee, and C. Han, "Sensor-based navigation algorithm for car-like robot to generate completed gvg," in *2011 11th International Conference on Control, Automation and Systems*, Oct 2011, pp. 1442–1447.

[106] J. Yang, A. Daoui, Z. Qu, J. Wang, and R. A. Hull, "An optimal and real-time solution to parameterized mobile robot trajectories in the presence of moving obstacles," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, April 2005, pp. 4412–4417.

[107] R. Gall, F. Tröster, and G. Mogan, "On the development of an experimental car-like mobile robot," in *2010 12th International Conference on Optimization of Electrical and Electronic Equipment*, May 2010, pp. 734–739.

[108] S. Rezaei, J. Guivant, and E. M. Nebot, "Car-like robot path following in large unstructured environments," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 3, Oct 2003, pp. 2468–2473 vol.3.

[109] C. Pradalier, J. Hermosillo, C. Koike, C. Braillon, P. Bessiere, and C. Laugier, "An autonomous car-like robot navigating safely among pedestrians," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 2, April 2004, pp. 1945–1950 Vol.2.

[110] H. Lategahn, A. Geiger, and B. Kitt, "Visual slam for autonomous ground vehicles," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 1732–1737.

[111] F. A. A. Cheein, R. Carelli, C. D. la Cruz, and T. F. Bastos-Filho, "Slam-based turning strategy in restricted environments for car-like mobile robots,"

in *2010 IEEE International Conference on Industrial Technology*, March 2010, pp. 602–607.

[112] V. Petridis and N. Zikos, "L-slam: Reduced dimensionality fastslam algorithms," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, July 2010, pp. 1–7.

[113] L. Palmieri and K. O. Arras, "Distance metric learning for rrt-based motion planning with constant-time inference," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 637–643.

[114] T. McMahon, S. Thomas, and N. M. Amato, "Reachable volume rrt," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 2977–2984.

[115] 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," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 3067–3074.

[116] J. A. Starek, J. V. Gomez, E. Schmerling, L. Janson, L. Moreno, and M. Pavone, "An asymptotically-optimal sampling-based algorithm for bi-directional motion planning," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 2072–2078.

[117] O. Arslan and P. Tsiotras, "Machine learning guided exploration for sampling-based motion planning algorithms," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 2646–2652.

[118] K. Shiarlis, J. Messias, and S. Whiteson, "Rapidly exploring learning trees," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1541–1548.

[119] L. He, J. Pan, and D. Manocha, "Efficient multi-agent global navigation using interpolating bridges," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 4391–4398.

[120] S. Shin, J. Ahn, and J. Park, "Desired orientation rrt (do-rrt) for autonomous vehicle in narrow cluttered spaces," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 4736–4741.

[121] F. Islam, V. Narayanan, and M. Likhachev, "A*-connect: Bounded suboptimal bidirectional heuristic search," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 2752–2758.

[122] T. Lozano-Pérez, "Spatial planning: A configuration space approach," in *IEEE Trans. on Computers*, vol. C-32, no. 2, Feb. 1983, pp. 108–120.

[123] S. M. Udupa, "Collision detection and avoidance in computer controlled manipulators." Ph.D. dissertation, Pasadena, CA, USA, 1977.

[124] "Nonholonomic system definition," https://en.wikipedia.org/wiki/Nonholonomic_system, accessed: 2017-03-10.

[125] "Google maps api," https://www.google.co.uk/maps/@53.800651,-4.064941,6z:, accessed: 2017-04-10.

[126] "Mapquest api," https://developer.mapquest.com/documentation/static-map-api/v5/, accessed: 2017-04-15.

[127] "Localisation definition," https://en.wikipedia.org/wiki/Mobile_robot_navigation, accessed: 2014-02-05.

[128] S. M. LaValle, *Planning Algorithms*. New York, NY, USA: Cambridge University Press, 2006.

[129] L. O. Noureddine Ouadah and F. Boudjema, "Car-like mobile robot oriented positioning by fuzzy controllers," in *International Journal of Advanced Robotics System*, vol. 5, no. 3, 2008, pp. 249–256.

[130] S. Karaman, M. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt*," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2011, pp. 1478–1483.

[131] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Rob. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011. [Online]. Available: http://dx.doi.org/10.1177/0278364911406761