Reactive Trajectory Planning in Structured Dynamic Urban Scenarios with Static and Dynamic Obstacles

Pradeep Korivi

From the Faculty IV - Electrical Engineering and Computer Science

Technische Universität Berlin

Department of Telecommunication Systems -Communication and Operating Systems

Master of Science

Guides : Prof. Dr.-Ing. Reinhardt Karnapke
Prof. Dr. Habil. Odej Kao

Eidesstattliche Erklärung
Ich bestätige, dass diese Masterarbeit meine eigene Arbeit ist und ich alle verwendeten Quellen und Materialien dokumentiert habe. Diese Arbeit wurde zuvor keinem anderen Prüfungsausschuss vorgelegt und ist nicht veröffentlicht worden.

Statutory Declaration
I confirm that this Master's thesis is my own work and I have documented all sources and material used. This thesis was not previously presented to another examination board and has not been published.

.....
Pradeep Korivi
Berlin, xxxx yyyyyyyyyy 2018

Acknowledgements

Firstly I would like to thank my thesis guide Zahra Boroujeni for her advice and help in various phases during my candidature. I am indebted to the Dahlem Center for Machine Learning and Robotics where I have met many fellow students who walked along and helped me understand various concepts of autonomous vehicles and provided valuable feedback during the discussions. I am thankful to Prof. Dr. Raúl Rojas and Prof. Dr. Daniel Göhring who have hosted me in their lab and supported me with the required resources during my thesis.

I am thankful to Prof. Dr.-Ing. Reinhard Karnapke, Prof. Dr. Habil. Odej Kao, Dr.-Ing. Daniel Graff for supervising my thesis at TU-Berlin and providing valuable advice on refining the thesis scope and help in streamlining my work.

Abstract

The goal of any motion planner is to achieve a driving trajectory that is collision free, smooth and responsive (reactive), at the same time being far-sighted to maintain consistency, deliberative and allow smooth transitions. Motion planning for robots is a widely researched topic in conventional robotics in unstructured environments. Similarly, trajectory planning for autonomous vehicles is also an extensively studied topic with computationally expensive planning techniques. This thesis proposes a new planning technique suitable for model cars running in structured environments and on low computational devices to generate behaviour similar to fully autonomous vehicles. This thesis offers a hierarchical sampling based algorithm designed especially for small robotic vehicles operating in structured dynamic urban environments. The first layer in motion planner creates a reference path from an obstacle-free road network definition file. The next layer creates multiple forward projecting samples (acceleration and lateral motion), and a pre-cost is assigned to them based on their closeness to the expected behaviour of ego vehicle. Trajectories are generated and evaluated in the lowest first order of the samples till the top of the stack has the lowest cost evaluated trajectory. This reduces the need to evaluate hundreds of trajectories when the trajectory with expected behaviour is found.

The next step in motion planning is to evaluate that the created trajectories are collision-free with respect to static and dynamic obstacles in the scene. This thesis employs a two-step collision detection for static obstacles by checking maximum of four points on the trajectory to decide whether the trajectory is collision free. This is possible by exploiting the properties of the polynomials used in trajectory representation and path modelling. For collision detection with dynamic obstacles, the approach to test for static obstacles is extended to check for collision in time when there is a collision in space for the ego vehicle and the obstacle. It is sufficient to check for the time gap between the ego and the obstacle only at the borders of the collision area to determine whether the trajectory is collision free.

The proposed planner has been verified by testing it in various scenarios that are encountered in urban driving. The planner was able to generate feasible and reactive trajectories that are collision free in different scenarios in dynamic environments and able to run on low computational platforms such as modelcar.

Abstrakt

Das Ziel eines jeden Bewegungsplaners ist es, eine kollisionsfreie, glatte und reaktionsfähige (reaktive) Fahrtrajektorie zu erreichen, die zugleich weitsichtig ist, um Konsistenz zu bewahren, zu beraten und sanfte Übergänge zu ermöglichen. Die Bewegungsplanung für Roboter ist ein weit erforschtes Thema in der konventionellen Robotik in unstrukturierten Umgebungen. In ähnlicher Weise ist auch die Trajektorienplanung für autonome Fahrzeuge ein umfassend untersuchtes Thema mit rechenintensiven Planungstechniken. Diese Arbeit schlägt eine neue Planungstechnik vor, die für Modellautos geeignet ist, die in strukturierten Umgebungen und mit wenig Rechengeräten arbeiten, um ein Verhalten zu erzeugen, das dem von völlig autonomen Fahrzeugen ähnelt. Diese Arbeit bietet einen hierarchischen, stichprobenbasierten Algorithmus, der speziell für kleine Roboterfahrzeuge entwickelt wurde, die in strukturierten, dynamischen städtischen Umgebungen arbeiten. Die erste Ebene im Bewegungsplaner erstellt einen Referenzpfad aus einer Straßendokumentationsdatei ohne Hindernisse. Die nächste Schicht erzeugt mehrere vorwärtsprojizierende Proben (Beschleunigung und laterale Bewegung), und ihnen werden Vorkosten zugeordnet, basierend auf ihrer Nähe zum erwarteten Verhalten des Ego-Fahrzeugs. Flugbahnen werden in der niedrigsten ersten Ordnung der Proben erzeugt und ausgewertet, bis die Oberseite des Stapels die niedrigste Kosten-bewertete Flugbahn aufweist. Dies verringert die Notwendigkeit, Hunderte von Trajektorien zu bewerten, wenn die Trajektorie mit dem erwarteten Verhalten gefunden wird.

Der nächste Schritt in der Bewegungsplanung besteht darin, zu bewerten, dass die erzeugten Trajektorien kollisionsfrei in Bezug auf statische und dynamische Hindernisse in der Szene sind. Diese Arbeit verwendet eine zweistufige Kollisionserkennung für statische Hindernisse durch die Überprüfung von maximal vier Punkten auf der Trajektorie, um zu entscheiden, ob die Trajektorie kollisionsfrei ist. Dies ist möglich, indem die Eigenschaften der Polynome, die bei der Trajektorienrepräsentation und der Pfadmodellierung verwendet werden, ausgenutzt werden. Zur Kollisionserkennung mit dynamischen Hindernissen wird der Ansatz zur Prüfung auf statische Hindernisse erweitert, um rechtzeitig auf eine Kollision zu prüfen, wenn es zu einer Kollision im Raum für das Ego-Fahrzeug und das Hindernis kommt. Es ist ausreichend, den Zeitabstand zwischen dem Ich und dem Hindernis nur an den Grenzen des Kollisionsbereichs zu prüfen, um festzustellen, ob die Trajektorie kollisionsfrei ist.

Der vorgeschlagene Planer wurde verifiziert, indem er in verschiedenen Szenarien getestet wurde, die im Stadtverkehr auftreten. Der Planer war in der Lage, machbare und reaktive Trajektorien zu erzeugen, die in verschiedenen Szenarien in dynamischen Umgebungen kollisionsfrei sind und auf niedrigen rechnergestützten Plattformen wie Modellauto laufen können.

Contents

Chapter 1

Introduction

Over the past years, autonomy has played a vital role in the development of automobiles. The vehicles have moved from being a pure mechanical wonder to a software and hardware combination. The developments are fuelled by advancement in low-cost sensor technology and government regulations to improve the safety of road transportation. At the same time, there is an increasing demand for fully autonomous vehicles with its potential in improving road safety and business services.

The past decade has seen a significant improvement in the development of technology for the autonomous vehicles bringing the science fiction of robots driving people a step closer. This technology has the potential for substantial societal impacts by reducing the fatalities in car accidents to changing the ecosystem of transportation. World Health Organization reports that every year there are more than 1.25 million deaths and between 20-50 million people suffer non-fatal injuries worldwide because of road accidents [53]. The accidents are because of many reasons such as road conditions, vehicle condition, driver alertness etc. The advancements in self-driving technology have a potential to reduce the risks by keeping track of road without distractions reducing human error, and also react quickly in emergency situations.

Increase in the urban population in many cities around the world is leading to increased traffic congestion, this has effects on health and productivity of the people [8]. Increase in congestion is a result of growth in car ownership, to combat congestion cities like Singapore are banning new car ownership [52]. Autonomous cars have the potential to shape the future of transportation by reducing car ownership, thus contributing to the reduction in congestion of cities. The report [56] from Morgen Stanley analyses the further impacts of the autonomous cars on industry and society.

1.1 Motivation

Autonomous vehicles are at the forefront of the research in automotive industry, to bring the technology closer to everyday use there needs to be further research. The new research required is mainly in the areas of urban driving involving a large number of traffic participants and complex road conditions. Though robots can act deterministically, humans and environment are not deterministic, to make autonomous vehicles safe the research should be focused at understanding these interactions.

To work on full-scale autonomous vehicles are expensive, time taking, and safety critical to test in emergency scenarios for understanding these interactions. Though

simulators provide easy ways to evaluate the situations, issues caused by physical aspect of the perception and vehicle control are missing in simulators. Research into autonomous vehicles is also an expensive endeavour with high costs on platforms, operational constraints and involves large teams. Modelcars (1:10 scaled versions of cars) can break this barrier to bring the technology closer to a large number of researchers in the world thus, being the enabler for further innovation. These Modelcars share the aspects of fully autonomous vehicles concerning expected behaviour and conventional mobile robots regarding sensor and computational resources sharing similar challenges of autonomous cars.

The challenges in autonomous vehicles can be subdivided into sensing/perception, navigation and control. The challenge under consideration in this thesis is the navigation/planning problem of driving the ego vehicle (vehicle in consideration for planning) to the destination with available resources. The problem of planning is unique as the module is responsible for understanding the behaviour of the surrounding objects and adjust the ego vehicle's behaviour accordingly following specific rules and react appropriately. Though motion planning for autonomous vehicles is a widely researched topic in structured environments, these methods cannot be applied to the modelcars due to constraints in computational power, sensor measurements and vehicle control. The methods proposed in classic mobile robots research either consider dynamic obstacles as pseudo-dynamic (momentarily static) or moving in a straight line not considering the road properties. There is a need for a planner to address these issues.

1.2 Problem Statement

The underlying problem of trajectory planning can be considered as a general robot path planning problem where the robot has to find a path avoiding obstacles, following constraints and moving on it. It can be formulated as an optimal control problem to determine set of states $x(t)$ or controls $u(t)$ for a dynamic system (ego vehicle) over a time (planning horizon) to minimise a performance index in a dynamic environment $r(t)$. The performance index could be one or multiple parameters such as distance to the goal, following various constraints such as no collision, maintaining the speed limit, comfort etc.

1.3 Thesis Statement

This thesis advocates that effective trajectory planning for modelcars in a structured environment similar to fully autonomous vehicles can be achieved at a low computational cost by combining prediction and approximation techniques in trajectory creation and evaluation.

1.4 Thesis Contributions

The main contribution of this thesis is the introduction of a trajectory generation framework and algorithms for modelcar platforms. The hierarchical framework employed divides the problem into four layers of planning named route planner, behavioural layer, trajectory planner and control node. The focus is the development of trajectory creation algorithm using samples of acceleration profiles and lateral shifts, an algorithm to pre-assign costs to samples such that planner evaluates the paths based on pre-costs and converges faster to the solution. The final contribution is collision detection algorithm that works in two steps for detecting collision with static obstacles and three steps for dynamic obstacles.

1.5 Thesis Structure

The thesis is broadly organised as follows. The initial chapters present relevant research in this field, followed by a chapter outlining the target hardware and software platform setup, and approach to motion planning. Implementation overview, evaluation and conclusion follow the approach.

- Firstly, Chapter 2 introduces the research direction in autonomous vehicles, followed by different planning approaches in on-road driving. Next, an overview of different trajectory representation and evaluation techniques is presented.
- Chapter 3 provides a basic overview of the software and hardware architecture of the target platform.
- Chapter 4 provides a detailed illustration of the motion planning algorithms developed for this thesis followed by evaluation techniques to validate the trajectories by checking for collision.
- Chapter 5 provides a basic overview of the implementation of the proposed planning algorithm by describing the structure of different modules implemented and message flow across them.
- Chapter 6 details on the different experiments performed to validate the planner followed by general criteria-based evaluation of the proposed planner.
- Finally. Chapter 7 concludes regarding the work done in this thesis, discusses regarding possible improvements and further research options.

Chapter 2

Related Work

Motion planning is a widely researched topic with roots in control, artificial intelligence, computational geometry and Robotics. The literature presented in [37], [39] review significant portion of standard planning techniques. Due to the complexity of the autonomous cars and the environments, general techniques from robotics cannot be applied. Planning for autonomous cars, in general, is subdivided into two categories namely planning in unstructured environments such as parking areas etc. and structured environments such as Road Networks where the traffic rules have to be followed. This chapter mainly focuses on the planning techniques in structured urban environments. The following subsections discuss regarding various approaches involved in planning, path representation techniques and finally regarding trajectory evaluation.

2.1 Planning Approaches

Autonomous vehicles have been in research communities from 80's with projects such as PROMETHEUS [2]; the research was accelerated by competitions such as DARPA Grand Challenge in 2006 and DARPA Urban Challenge (DUC) in 2007 [7]. In DUC, six autonomous vehicles from different universities have completed the challenge and have used various techniques to accomplish the task. The methods developed during these events formed the basis for modern-day autonomous cars which perform with greater safety and comfort compared to the DUC participants.

Approaches followed by different participants of DUC are described in [7]. All participants in the competition followed a similar procedure with differences in internal techniques to solve subproblems. Planning module of Boss, autonomous vehicle from Carnegie Melon University which won the DUC is described in general here. Its planning framework is mainly subdivided into three submodules

- Mission control: It is the higher level module which creates a global path, assigns lane to the vehicle to reach the checkpoints and detects blockades.

- Behavioral module: It is responsible for decisions such as precedence at intersections, lane change decision, speed planning, look ahead point assignment etc.

- Motion planning module: It is responsible for generating local trajectories and converting them to steering and acceleration values. It initially creates a forward-looking path, and velocity planning is performed on top of that.

After DUC, research efforts have been increased to develop state of the art solutions in perception, planning and control of autonomous vehicles. Motion planning plays a crucial role in the system with the aim of building trajectories that are collision free and also adhere to kinematic and dynamic constraints of vehicle, road boundaries and traffic rules [28]. There have been numerous approaches to solving the problem of path planning, and some of them are discussed in this section. Potential fields is one of such algorithm, it models state space with attractive forces towards goal and repulsive forces near obstacles [33] [50] [61]. In this approach, a path is found by travelling along the steepest gradient of the potential field. However, there is a risk of paths getting trapped in local minima. The grid-based approach is another method used in Robotics, here environment is perceived as a set of grids and path is found travelling across these grids using algorithms like A*. The downside of these approaches is that the complexity increases exponentially with increase in grid resolution and grid size, there have been different variants of this approach as discussed in [41] [15] [32]. A comprehensive study of various approaches in motion planning for autonomous vehicles is presented in [28] [49].

Planning approaches that are widely used in autonomous driving are classified as shown in Figure 2.1. The following subsections discuss research in each of the described categories.

2.1.1 Random Sampling Approaches

Rapidly-exploring Random Tree(RRT) technique was initially introduced by Steven M. LaValle in his work presented in [38]. RRT builds a tree incrementally by sampling new states. In each iteration, a new sample x is sampled and connected to the nearest neighbour x_{neartest} in the tree if it is collision free. The tree starts at the start location and is built till the path to goal location is found. Probabilistic Roadmaps algorithm[29](PRM) is another approach where uniform sampling is performed across the state space and are connected if a collision-free path exists. Then a graph search algorithm is employed to find a path from source to destination. Both of these approaches are probabilistic complete, i.e., as the number of samples increases the probability of finding a solution approaches one given problem is solvable.

Different variants of RRT's are widely accepted in robot motion planning because of its ability to explore higher dimensional problems at ease[60]. To improve the performance of RRT, bidirectional RRT with two trees(Bi-RRT) has been proposed. Though it enhances the performance, handling discontinuities between two trees are difficult[34]. Environmentally guided RRT(EG-RRT) is another variant that incorporates information from scene analysis to guide samples and converge faster[26].

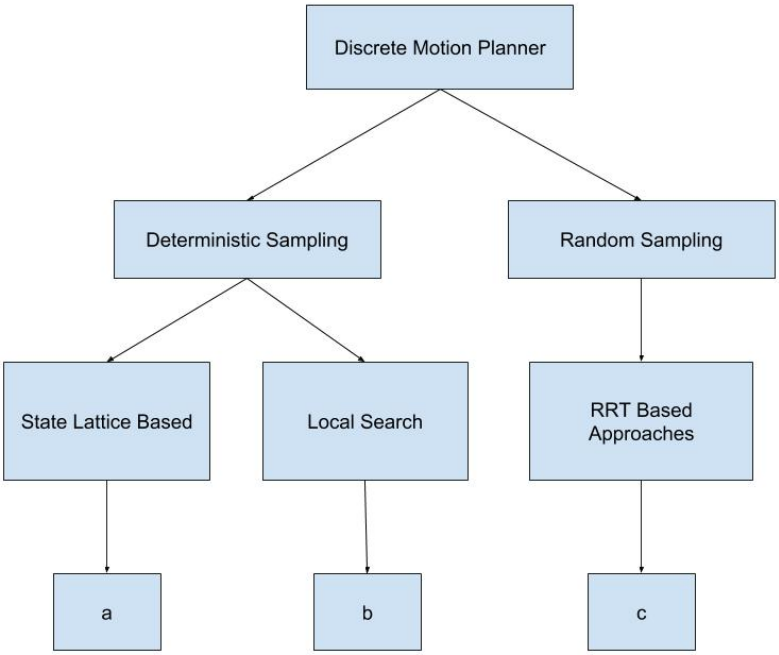


Figure 2.1: General Overview of On-road planners. a - [45] [58] [62] [23] [5] , b - [32] [40] [7], c -[19] [35] [16]

MIT has applied a closed loop prediction model into RRT(CL-RRT) in its autonomous vehicle at DUC [36], a snippet of this planning approach is shown in Figure 2.2. In CL-RRT motion models are used to generate trajectories to sampled states and are evaluated for feasibility and performance. A variant named RRT* [19] has been proposed which guarantee asymptotic optimality. In this approach, each state stores cost from the start and when a new sample is added, surrounding neighbours are tested if a better path can be found and the tree is rewired thus leading to an efficient path. There have been many techniques to reduce the number of sampled states in-order to save computational time, Informed-RRT [18] is one such approach which samples space according to the actual best path and states that are far away from the goal are not sampled.

In real-world situations' environment is not entirely predictable and the computed path needs to be dynamically re-planned over the time. This replanning needs either a new plan to be generated quickly or rapid correction of old path. The approach Anytime-RRT [27] creates an initial path using RRT and optimises it on the go to create an optimal path, re-planning is triggered when the path is no longer valid. The method RRT^x [48] updates the search space whenever obstacle changes are observed and repair the surrounding tree.

Though RRTs are best-performing algorithms in planning, they exhibit certain deficiencies concerning motion planning for autonomous vehicles, especially in on-road

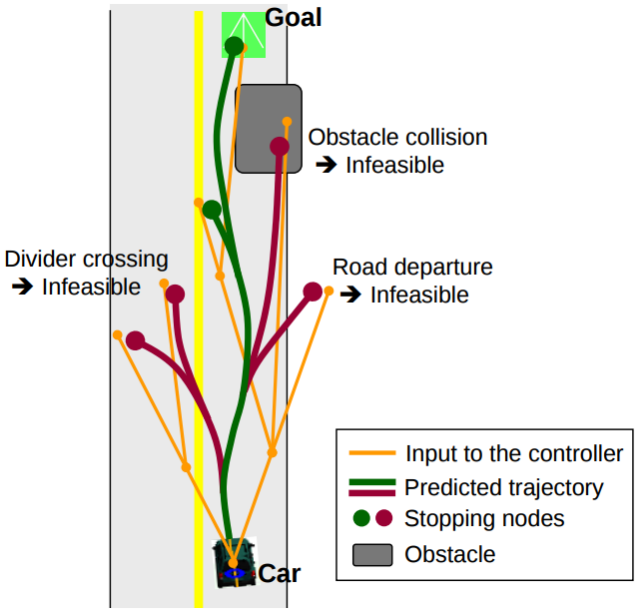
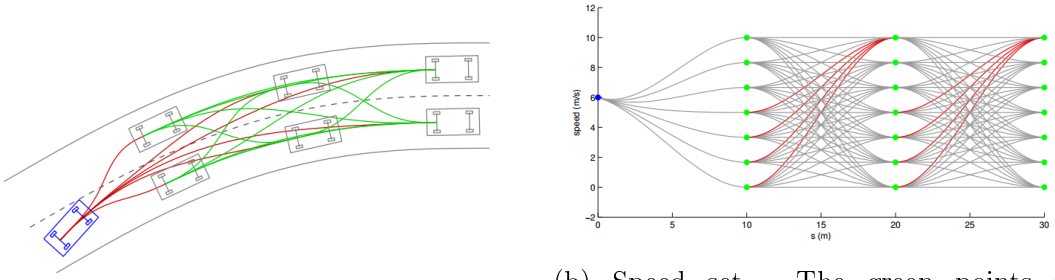


Figure 2.2: MIT Darpa Urban Challenge - RRT based planning. In this approach, motion plans are propagated using vehicles dynamic model, and these paths are evaluated for feasibility

driving conditions. RRT-based planners generate jerky and unnatural trajectories that contain many unnecessary turns[13]; these are suitable for open areas such as parking areas. Road parallel paths are preferred in structured environments.

2.1.2 Lattice Planners

The lattice planners use a discrete representation of the planning area with multiple states, each state is multidimensional with dimensions such as position, acceleration, velocity, time, curvature, heading etc. These states are connected, then the problem reduces to finding a path from the initial state to the final state in the lattice. This approach is well suited for non-holonomic robots and highly constrained areas such as road networks[14]. Lattice planners are resolution complete, i.e., they can be automatically adjusted to change in resolution to explore state space consistently. The approach proposed in [14] uses a multi-resolution state lattice with high resolution near start and goal locations and lower resolution in the middle to reduce computational complexity. Continuity of path and curvature are constraints in path planning, these are addressed in the research [54] by defining a 4D configuration including 2D position, heading and steering. In [45] the author added curvature to each state along with 2D position heading and curvature, paths between the vehicle and sampled sates are connected using third order spirals. A range of times and velocities are assigned to each vertex to enable spatiotemporal search. It uses constant acceleration profiles



(a) Path set. blue vehicle - current vehicle pose, and grey vehicles - sampled endpoints, red paths-quartic curvature polynomials and green paths - cubic ones

(b) Speed set. The green points are sampled speeds at sampled longitudinal states, the red curves are beyond the acceleration limit, and the grey curves are valid ones

Figure 2.3: Lattice Planner - Path velocity representation in [62]

making it difficult for the vehicle to follow. Thus, due to a multitude of states involved, the number of trajectories created are in order of few hundred thousand. The number of trajectories to be evaluated increase exponentially if the state resolution is increased or new dimension is added. A graphical processing unit(GPU) is needed to run the evaluations in parallel to provide real-time response.

To improve the performance of [45], the research published in [62] utilizes quartic polynomials to ensure continuous curvature, connections are made from sampled endpoints and current state, Figure 2.3 shows further information about path and velocity representation. In this approach, speed profiles are generated inversely, and checks are included to ensure comfort, efficiency. To reduce the computational complexity the method proposed in [24] uses a two-level planning approach which initially generates optimal collision-free reference path and then performs the search across this reference path to find an optimal trajectory. The reference path leads to a focused search and more human-like driving style. The research published in [58] further improves the trajectory smoothness ensuring a high level of trajectory diversity.

In summary, lattice planners create trajectories that are smooth, optimal and complying with dynamic and kinematic constraints of the vehicle. These approaches are well suited for structured and dynamic environments. They are also computationally expensive, and complexity increases exponentially with the addition of new state or increase in resolution.

2.1.3 Local Search

Local search is the most popular technique in autonomous driving, in this method instead of searching the complete graph a local state space is searched for a feasible solution as shown in 2.4. The planners proposed in [7] [47] [32] [6] [40] [42] perform

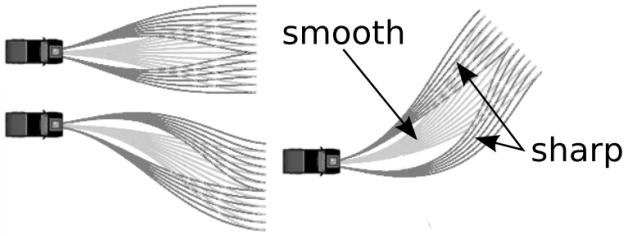
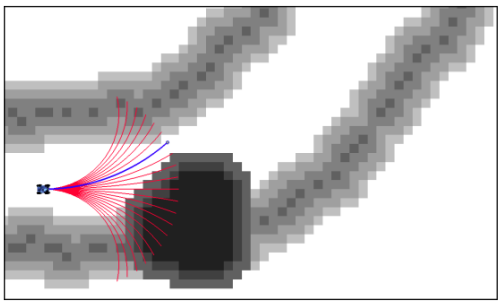


Figure 2.4: CMU Darpa Urban Challenge - Local Search Forward projected trajectories

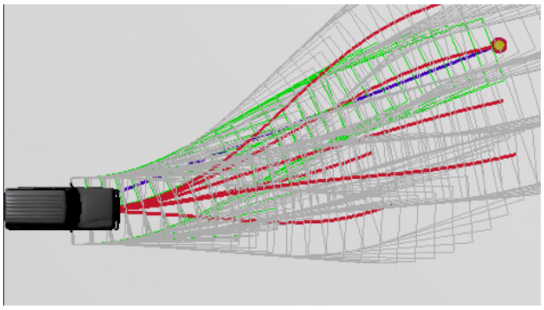
a variation of local search. In this approach a set of trajectories are forward projected based on vehicle dynamics, controls, to reach sampled states. The generated candidate trajectories are evaluated with a set of cost functions for collision checking and comfort to finalise the final trajectory. Paths with lateral shifts can be split into two categories, i.e., lateral shifts in action space(controls as a function of time and controls as a function of time and state) and state space(position, orientation, linear and angular velocities, curvature) of vehicle[25]. Partial motion planning is another technique used to search locally reducing computational power significantly [3]. This technique has gained popularity due to ease in implementation and ability to perform rich sampling in short horizon for solving motion planning problem. In summary, local search algorithms can create short-term trajectories at a low computational cost with limitations in the manoeuvres robot can perform, these are especially suitable for low-speed and on-road driving.

2.2 Trajectory Representation

Trajectory or path representation is another crucial factor in quantifying the trajectories; there are different methods such as arcs, second to fifth order polynomials, Cubic to Quintic Bezier curves, Dubian Paths, Reeds-Shepp curves, Akima splines, splines etc. Each of these curves has different advantages, disadvantages and suitable in different environments as discussed in [28]. Splines and polynomials are used for creating road parallel trajectories, fifth order polynomials produce jerk minimising trajectories[59], and other formats of trajectory representation mentioned earlier are employed by different planners discussed in previous sections. Reeds-shepp curves, a version of Dubian curves allow forward and backward driving [51] thus making them suitable for complicated manoeuvres in parking lots, obstacle course etc.



(a) Static Environments - Map is represented by a set of grid cells; the cells are assigned cost based on the presence of obstacle and closeness to obstacles. Cost of the path is computed based on the cost of the cells it traverses through, darker the cell higher the cost.



(b) Dynamic Environments - The vehicle shape is forward projected for each path, and the collision checking is performed at each step. If at any point the shape of the ego vehicle collides with an obstacle, the path is marked invalid.

Figure 2.5: Collision Checking for Static Obstacles [32]

2.3 Trajectory Evaluation

Trajectory created by different methods mentioned in previous sections must be validated against various constraints to check feasibility, comfort, optimality and collision. The approach proposed in [62] combines costs from different cost functions to check each trajectory for path length, curvature, the rate of change of curvature, lateral offset to the closest centre line, transformed distance to static obstacles, duration of the plan, speed, acceleration, jerk, centripetal acceleration, distance to dynamic obstacles to rank the trajectories. All the planners' test for some or all of the parameters mentioned using cost functions; some weigh each cost differently based on optimising factor, [5] penalise breaking to prefer long-range trajectories, [45] penalises shorter horizons to ensure minimum horizon. Primary criteria for trajectory evaluation is collision checking to ensure safe motion and comfort.

Simulation-based techniques are widely used in collision checking where the vehicle is simulated in time to check for collision with other obstacles.

In [32], for collision checking in static environments, a map is represented as grid cells and based on obstacles and traversable areas the cells are assigned cost, and the trajectories that pass through these cells are added with corresponding costs as shown in Figure 2.5a. Finally, the trajectory with the lowest cost is selected. In dynamic environments' vehicle shape is forwarded and checked for collision with static obstacles as represented in Figure 2.5b.

For collision checking with dynamic obstacles, [32] uses a hierarchical approach. In this method initially given vehicle trajectory and obstacle trajectory bounding boxes are constructed and checked for collision, if they intersect then in incremental

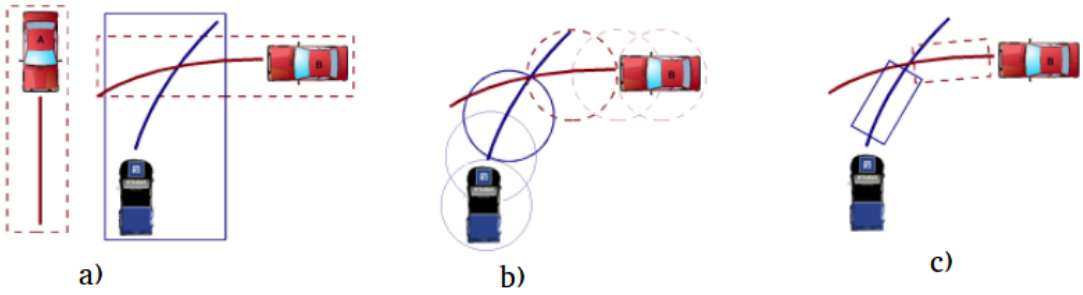


Figure 2.6: Collison checking for Dynamic Obstacles [32] a) Initially bounding boxes along length of trajectory are drawn to determine if the trajectories collide, if they collide b) circle approximating the shape of the ego is forward projected and checked for collision, if this is also colliding then c) shape of the ego is forward projected and checked for collision.

time steps a pessimistic approximation of circles is used to check for collision, if they also collide then actual model of the car is used to check for collision as represented in Figure 2.6.

In [45], for collision checking with respect to static obstacles, each (x,y) point in world is assigned a cost based on proximity to static obstacles, a path that intersects with obstacles has lethal cost, paths that have proximity to these obstacles have high costs and paths that are away from obstacles are assigned zero cost. A Potential function is created based on the position of obstacles to that computes the cost of the path. Similarly, for dynamic obstacles, a potential function is created by adding t dimension to create cost function in (x,y,t). Static and dynamic obstacles are dilated accordingly to remove errors in perception and ensure safety.

In [19] collision checking is performed by forward projecting the vehicle shape represented as rectangle and collision checking with lines joining the initial and final state of obstacles and border lines. The planner proposed in [9] presents a reactive approach, in this distance to the dynamic obstacles ahead, is used as a parameter to stop the vehicle.

The approaches proposed for collision checking rely on predicting future trajectory by assumptions that obstacles will continue with constant velocity, acceleration, current orientation, in the same lane etc., these assumptions are not accurate and may lead to inefficiencies due to ignoring traffic context, interactions between vehicles etc. [28]. There are also uncertainties in data predicted by the perception module, and they must be considered while collision checking.

In summary, collision checking is an important and also a complex task which involves different types of checks, assumptions, and approximations to create trajectories that are safe.

Chapter 3

Target Platform

Figure 3.1: Model car external view

The AutoNOMOS Model Car presented in Figure 3.1 is a 1:10 scaled car, developed as a research platform for educational purposes at Dahlem Center for Machine Learning and Robotics of Freie Universitat Berlin [17]. An overview of the hardware and software components is presented in the next two sections.

3.1 Hardware Components

The target platform for validating the planner is a 1:10 scale model car (AutoNOMOS Mini V3) as shown in Figure 3.1, it is developed as an education and research platform. The car is a modified RC platform, different hardware components used are described in Figure 3.2. The main components are a Brushless DC-Servomotor (FAULHABER 2232) to drive and measure speed, a servo motor (HS-645MG) to control the Ackerman steering, IMU (MPU6050) to measure the orientation of the car. The three modules mentioned are controlled with an Arduino Nano which communicates to the main CPU (Odroid XU4 - Embedded Computer). Other important components are a depth camera (realsense sr300) for forward vision and perceiving the shape of obstacles ahead, 2D-Lidar (RPLidar), WiFi Dongle, Fisheye Camera

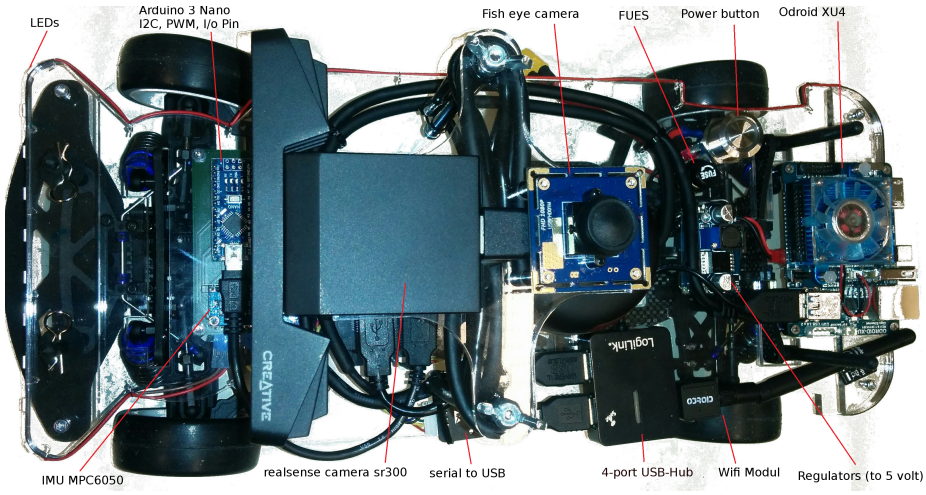


Figure 3.2: Internal components of Model Car [1]

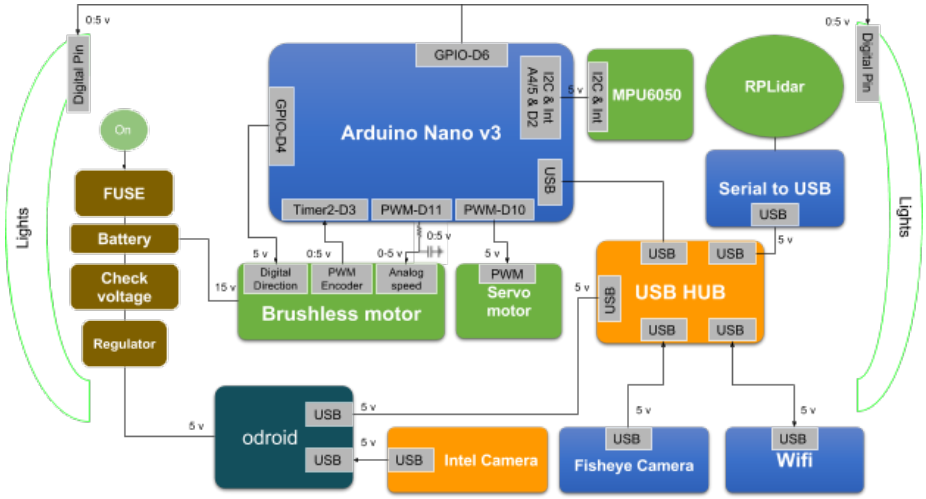


Figure 3.3: Module Connections [1]

for localizing car based on markers on the roof. The Figure 3.3 presents connections across different modules present in the car.

3.2 Software Components

The CPU onboard runs Robot Operating System (ROS) on top of Ubuntu. ROS is a widely known framework for robotics applications due to its modularity and distributed nature. Modularity allows the application designer to choose which modules to be developed and which modules to be used directly. There are thousands of packages available open-source developed by the community which help to realize a project in a short time. The distributed nature of ROS allows to distribute modules across

different platforms based on hardware requirements and communicate easily across them. Though ROS is not real-time capable, it is fast and well suited for robots running in laboratory environments. Many of the issues present in ROS are being addressed in ROS2.0 aiming at industrial scale robots.

The model car implements core components for motor control, reading data from sensors, Odometry, Camera interfaces etc. There are community developed packages for visual GPS to track markers on the roof and localize car, line detection, traffic sign detection etc.

In summary, the model car features all the components to drive the car autonomously, but there are limitations on the accuracy of measurements, control and computational resources on board.

Chapter 4

Planning Algorithm

4.1 Introduction

The goal of the path planner is to navigate the robot from the start state to destination by blending in the traffic. Path planning module is dependent on various modules to receive the data regarding the perceived environment and invoke a set of modules to move the ego vehicle safely. It has to drive the ego vehicle forward considering the traffic rules, obstacles, kinematic and dynamic constraints of the robot and not compromising on the safety and comfort of the passengers inside. This chapter aims to describe path planning techniques developed in this thesis to drive the ego vehicle safely to the destination.

This section is organised to provide an overview of different modules needed for path planning and methods used in each module to achieve the goal. Path planning starts from the initial understanding of where the ego vehicle is and where to go, subsection 4.2 describes the localisation of ego vehicle to provide this data, subsection 4.4 provides the information on how a global path to the destination is calculated. An autonomous vehicle should have an understanding of surroundings concerning where other vehicles are, where pedestrians are, traffic signal information etc. Prediction module described in 4.3 details further on how the ego vehicle perceives the environment. The next subsection 4.5 describes in-depth details on the short-term planning algorithm aka trajectory planner. The final module in the discussion is control unit, described in subsection 4.6. It is responsible for translating the path in space-time into steering and acceleration values to drive the ego vehicle.

Figure 4.1 represents the general architecture of the planning module. It details on the flow of information, dependencies, relative execution frequency. The components in dark blue are the primary focus of this thesis, parts in light blue are adopted from other works and modules in sky blue are preexisting or simulated.

4.2 Localization

Localization module is responsible for providing the current state of the vehicle concerning the position, orientation, speed(linear and angular) and acceleration. The localisation module implemented on the modelcar has two sub-components, Vehicle Odometry and a Visual GPS. Odometry is calculated with dead reckoning [10] with speed information from motor and Yaw information from Inertial Measurement Unit

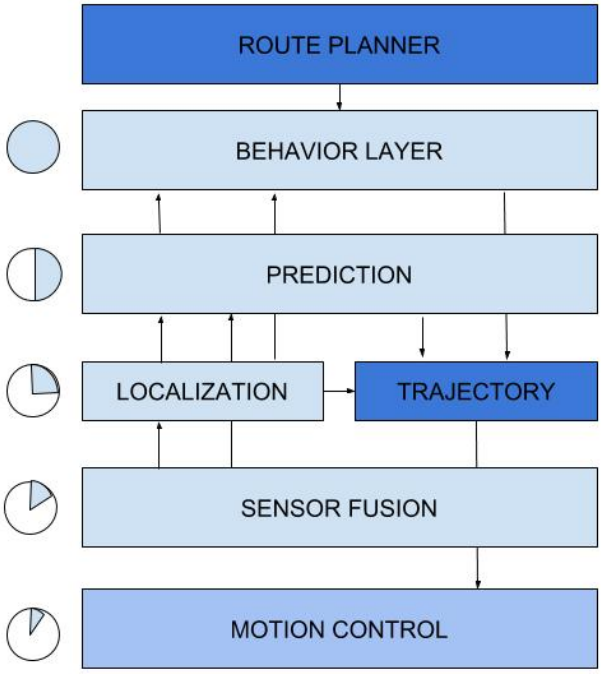


Figure 4.1: Path Planning Module

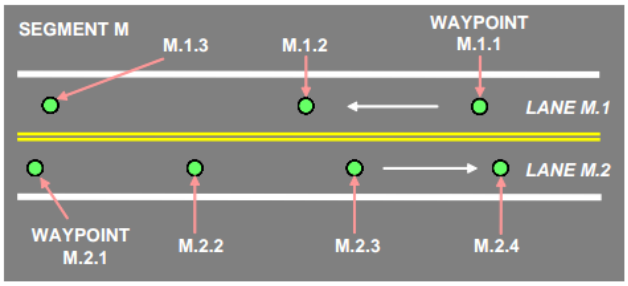
(IMU). The localisation module combines odometry with the data received from a visual GPS node (tracks markers on the roof) to correctly estimate the state of the ego vehicle.

4.3 Prediction

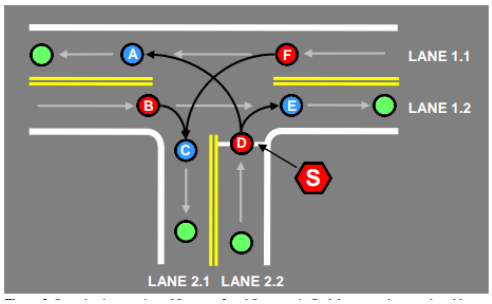
Prediction and Sensor fusion modules receive the data from various sensors such as Cameras, LIDAR, etc. and fuse them together to create an environment model, classifying objects into different categories and predict the state of obstacles in the surroundings. Due to time constraints, this thesis simulates a prediction module to provide motion planner with obstacle information in different traffic scenarios.

4.4 Route Planner

Route Planner is responsible for finding a global route between the current vehicle state and the goal state based on the static characteristics of the environment/map such as lane information, speed limits etc. Route planner obtains this information generally from the maps or other formats to represent the road network. In this thesis a simple model called " Road Navigation Definition File (RNDF)" [12] [11] is used to represent the route network. Next subsections details further about RNDF and global reference route calculation.



(a) Segment representation in RNDF - Segement M has two lanes M1, M2 and each lane has way points 1-N



(b) Exit representation in RNDF - Connections between two segments in a T-Junction

Figure 4.2: Road representation in RNDF [12]

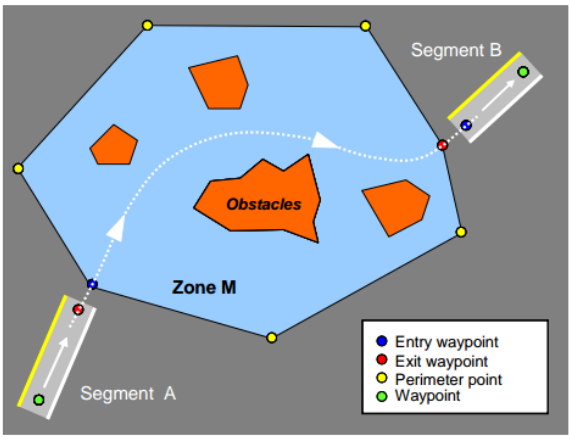


Figure 4.3: Connection between Segments and Zones (free paths) in RNDF [12]

4.4.1 RNDF

This section details about the RNDF file [12] which defines the road network(set of roads/ areas connected) over which the vehicle can traverse. DARPA developed this representation of road for its Autonomous Vehicles Urban Grand Challenge. RNDF representation first divides the traversable areas into two parts, segments and free zones and provides connections across these areas. Free zones represent areas such as parking lots and road segments represent driving lanes. Each segment has multiple lanes; each lane has waypoints along the driving direction. More significant information about waypoints such as whether it is a stop sign, speed limit, start/end, intersection etc. can be added. Each segment/zone is connected to one another using exits, they represent the connections between one segment waypoints at start/end to another. Figures 4.2a 4.2b 4.3 [12] represent different portions of the route representation and Figure 4.4 details regarding one of the route network of map used in Lab experiments.

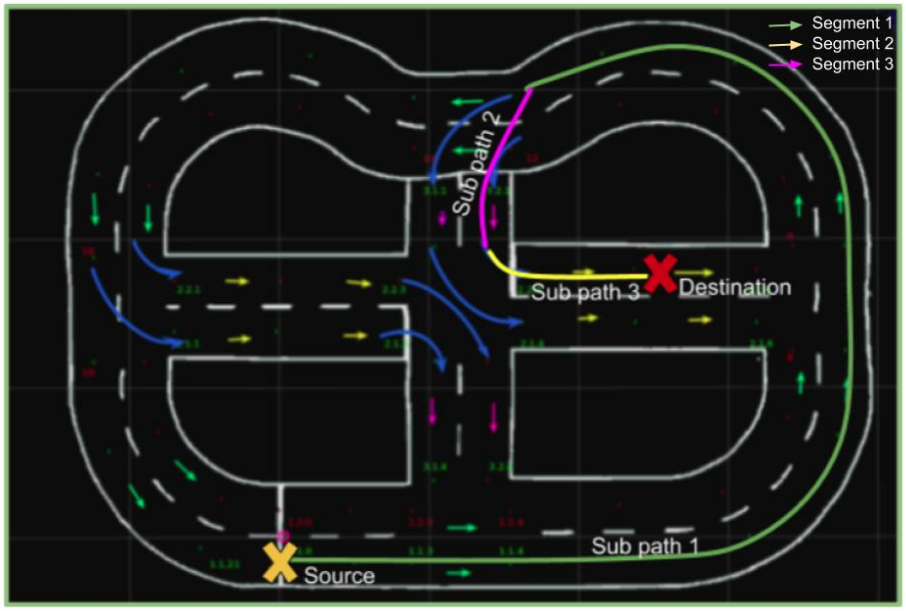


Figure 4.4: Division of shortest path in road network into Sub paths across different segments

4.4.2 Path Representation and calculation

The data in RNDF is represented in the form of a graph with connections across way-points as edges and way-points as vertices. The global path from source to destination is the shortest path between the waypoint closest to the ego vehicle's current position and the waypoint closest to the destination. The shortest path found is divided into sub-paths based on the segment to which the waypoints belong. Once the ego vehicle is at the end of one sub-path it receives a notification from the trajectory planner that a goal is reached, then the route planner transmits the next subpath to the trajectory planner, this process is repeated till destination is reached. Figure 4.4 details further about the division of shortest path across different segments. This method also reduces the memory needed in modelling the road in trajectory planning stage.

4.4.3 Behavioural Layer

Behavioural layer plays an essential role in path planning; it is responsible for understanding the scenario and making decisions according to various traffic rules, constraints and choices that make driving efficient. For example, it decides on target lane and speed based on information such as emptiness of road, other cars in the lane, next needed exit and turn, speed limit etc. The behavioural layer is a vast research topic in itself and not in the scope of this thesis. Currently, a simple simulated approach is implemented where the user can provide inputs to decide behaviour using mouse clicks for lane change and speed information from the map.

4.5 Motion Planner

Motion Planning/Trajectory planning creates short-term space-time trajectories that drive the robot safely adhering to global path discussed in 4.4. This section is organised into multiple subsections which describe each of the subproblems in creating local trajectories. The subsection 4.5.1 provides an overview of the timing Horizon and constraints in dynamic environments for different modules in planning. The next subsection 4.5.2 details regarding path modelling and how this will improve the efficiency of planning along with its constraints. It also discusses on approximation method used to convert coordinates from Cartesian to Frenet frame and vice versa. The core of this section is the creation of trajectories which is discussed in detail in subsection 4.5.3. The next subsections 4.5.4 and 4.5.5 explain further on how the created trajectories are evaluated for collision with static and dynamic obstacles. Next subsection 4.5.6 details on the selection of final trajectory from the set of evaluated trajectories for trajectory following.

4.5.1 Temporal Horizon

Time is an essential aspect of planning in dynamic environments, and there are several timing variables associated with planning. This section is mainly adopted from the doctoral thesis "Autonomous vehicle navigation in dynamic urban environments for increased traffic safety" [44]. These timing parameters define how far into the future different sub-modules of planning will be valid.

The first timing variable in motion planning is timing horizon T_m. It is the measure of how far into the future trajectory of the ego vehicle is planned. Second is the prediction horizon T_p; it is the measure of how far into the future the motion of dynamic obstacles around the ego vehicle can be predicted. The fundamental requirement of planning to be valid is that $T_m \leq T_p$ such that planning is done only so far into the future as the environment is predictable.

Third is T_d, which indicates the computation time of the motion plan. Assuming planning is done in cycles, the plan created in the previous cycle is executed in the current cycle, thus $T_d \leq T_m$. If this condition fails, then the planner will run out of path for the next cycle. In general $T_d \ll T_m$. The fourth timing variable T_s is the perception update cycle time, i.e., perception module updates the state of surrounding dynamic obstacles every T_s seconds. In general world, the predicted trajectories for duration T_p will not hold true, as the behaviour of these vehicles is not controlled by the ego vehicle. Thus the constraint $T_s \leq T_p$ should be valid. This creates an uncertainty in modelling of the environment. Thus the execution duration of current plan T_e beyond T_s is not sensible; this is because obstacle trajectories may have changed in T_s and executing the old trajectory may lead to collisions invalidating the trajectory created for T_m.

The next timing constraint in consideration is T_e, control execution time of the

current plan. T_e should not exceed the perception update time T_s. This restriction also imposes additional constraint on T_d (motion plan computation time), $T_d \leq T_e$.

In summary, timing constraints described above identify the relation between different modules such as motion planning, motion prediction and execution. It is also important to predict farther into future than T_s or T_e for completeness of motion planner concerning goal objective. In general, a farsighted uncertain motion plan potentially directing the vehicle towards the goal is better, but this plan needs to be re-evaluated and re-executed in short intervals for correctness.

In general behaviour of obstacles and participants can be predicted for up-to 5s probabilistically. Thus the temporal T_m & prediction T_p horizon are chosen to be 5s. The planner has an execution time T_d far less than recommended 100ms on a low power computational hardware which allows a high update rate allowing lower values for T_e. As obstacle detection is simulated, a pessimistic value of 250ms for T_s is chosen, and the planner is capable of handling higher update rate also.

4.5.2 Path Modelling

The planned global path is in the Cartesian coordinate system. One of the problems with the Cartesian coordinate system is that the due to variation in curvature of the road, local planning becomes complex. To address this issue planning in curvilinear system or Frenet Frame or lane adoptive (SL) coordinate system has been adopted by researchers, [62] [63] [58] [40] [9] [31] are some of the research works in which Frenet frame is adopted. In this method centre of the lane/road or preplanned global path is used as reference longitudinal coordinate (S) and perpendicular distance with respect to the lane centre is considered as lateral coordinate (L/D) as represented in Figure 4.5 [58]. Thus once converted, (S, L) coordinate system essentially is a straight road.

Conversion from Frenet frame to Cartesian and vice versa is a widely researched topic and many techniques exist which offer different levels of complexity and accuracy. In this thesis, an approximation method is used to convert between these two coordinate systems similar to [9]. This method is computationally inexpensive and provides a required level of accuracy for the model car. To convert an XY coordinate to SL coordinate, (x,y) is projected onto the current path represented by straight lines joining waypoints as in figure 4.6, cumulative distance till this point gives the S coordinate, and the perpendicular distance between the projected point and the current point provides the L coordinate. A similar process is used to convert S,L coordinate to x,y coordinate. S is used to find a point on a segment represented by waypoints, a point at a perpendicular distance L gives the x,y coordinate. We assume that the path between two waypoints is linear which reduces the computational complexity in approximation. This approximation, however, approaches zero error when the spacing between two waypoints approaches zero. Adding dense waypoints in the curves significantly reduces the approximation error. There are different methods

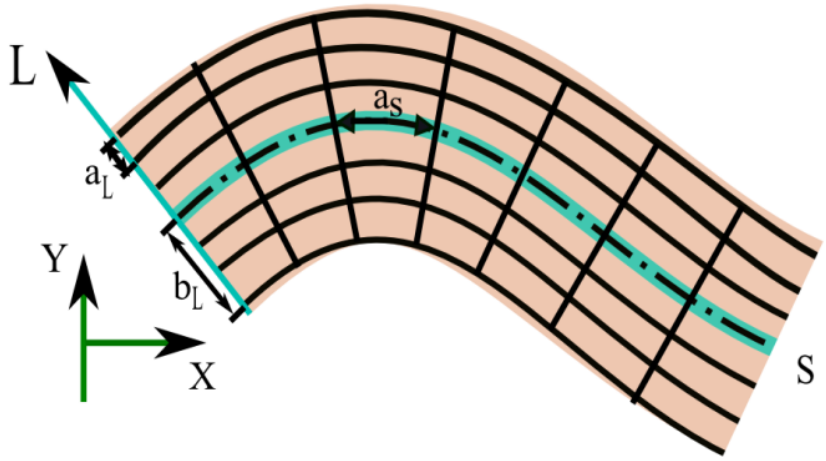


Figure 4.5: SL coordinate system laid over XY coordinate system - [58]

discussed in [57], [30] which provide better accuracy in calculating the paths.

As observed in Figure 4.5, in SL coordinate system the size of the unit distance is not constant, it stretches in the convex side of the road and gets compressed in the concave side of the reference line. This is especially an issue in curves with a lower radius of curvature, as it affects velocity planning thus leading to discomfort in some cases. There is extensive research on the topic of velocity and path smoothening which counter these effects.

In summary, the curvilinear coordinate system makes planning easier but needs extra computation in the conversion from one format to other. It also introduces some errors and inefficiencies in planning if the complete planning is done in SL coordinate system and these need to be addressed.

4.5.3 Trajectory Creation

The core of this thesis is the trajectory planner that drives the robot from source to destination. By understanding the behaviour of human drivers in structured environments (road networks), it is necessary that the trajectory planner creates trajectories that avoid collisions, align with the road network, that are smooth, continuous and comfortable. Chapter 2 discusses various planning techniques used by different planners. This subsection is divided into two sub-subsections for longitudinal and lateral planning of trajectory.

4.5.3.1 Longitudinal Planning

The approach proposed in this thesis is inspired by human driving, i.e., the driver tries to maintain an optimal speed, next shift laterally based on obstacles ahead and brake if a collision is predicted with current driving state or perform an evasive man-

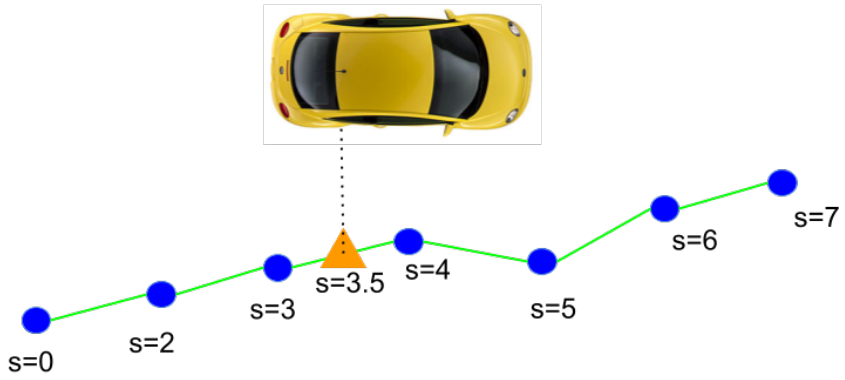


Figure 4.6: Showing projection of the current position of the car shown by green triangle to s coordinate system. Each circle represents a node, and the lines between them are links. [9]

oeuvre. To reach a speed vehicle need to accelerate/decelerate, this can be achieved with various levels of values based on current state as, each acceleration/deceleration level chosen will result in different final states. The initial step is to sample set of acceleration profiles, Figure 4.7 shows different acceleration profiles the ego vehicle can follow in time horizon. Currently, constant acceleration profiles are used due to limitations in the ego state measurement and control, once state approximation and control are improved the planner can be switched to trapezoidal acceleration profiles as shown in sub-figure4.7. A1 represents an acceleration equivalent of around $2m^{-2}$ and A8 of up to $-8m^{-2}$ (in actual cars) which is on the higher end of decelerations, most of the cars are not capable of achieving such large decelerations because of various factors such as road condition, tires etc. Generally deceleration values are up to $-4.5m^{-2}$ [22] [46] [4]. Applying each of this acceleration profile to current ego vehicle state for planning horizon T_m leads to different final states of ego vehicle. The final sates will have different final velocity as shown in Figure 4.9, different distances traversed as in Figure 4.8.

Change of acceleration is defined as jerk and to create smooth trajectories it is important that the trajectories generated by the motion planner must have least jerk. There are various techniques to create these jerk free trajectories as discussed in chapter 2. The selection of smoothness also depends on the capabilities of the ego vehicle and controller to track these fine trajectories.

4.5.3.2 Lateral Planning

Acceleration profiles discussed in 4.5.3.1 solve the problem of longitudinal planning but to avoid obstacles the ego vehicle should also plan lateral (sideways) shifts in its

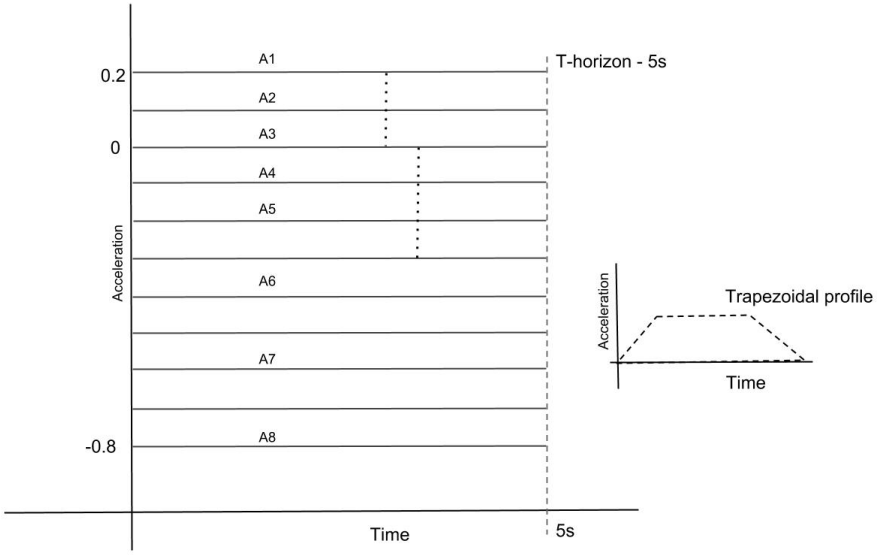


Figure 4.7: Different acceleration profiles a car can follow from current state. If target velocity is reached target acceleration is switched to zero as indicated by dotted lines.

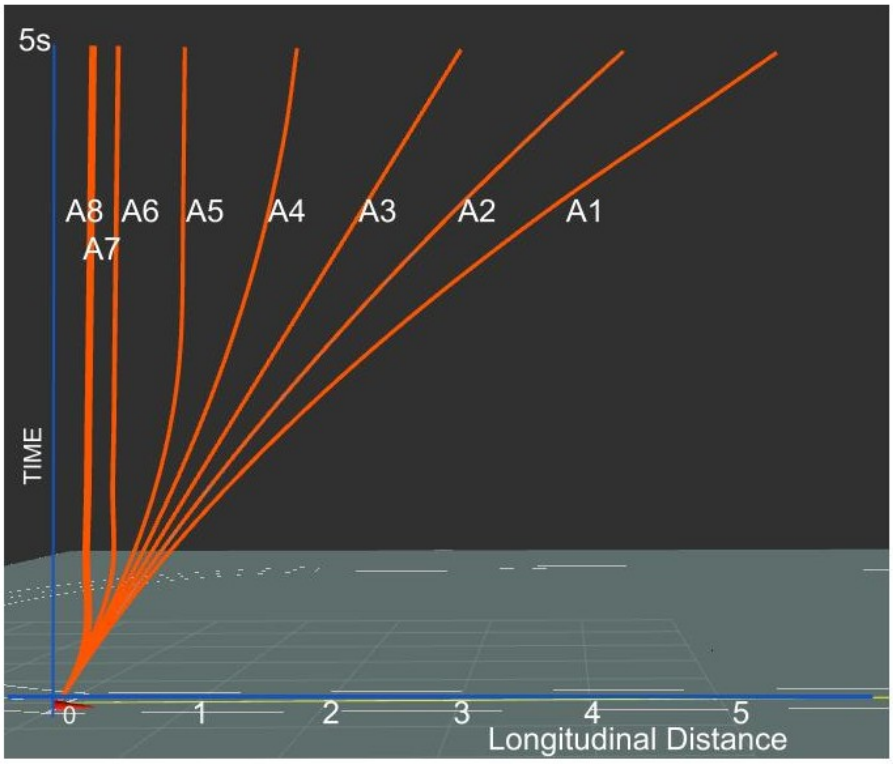


Figure 4.8: Distance traversed vs time with accelerations A1-A8 from 4.7 in time horizon(5s), Initial position = (0,0) velocity = $0.6ms^{-1}$, target velocity = 1.5 & 0 ms^{-1}

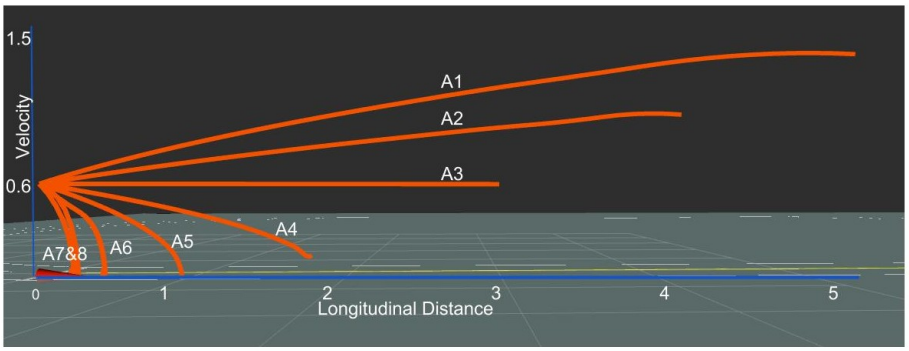


Figure 4.9: Distance traversed vs Velocity achieved with accelerations A1-A8 from 4.7 in time horizon(5s), Initial position = (0,0) velocity = $0.6ms^{-1}$, target velocity = 1.5 & 0 ms^{-1}

trajectory. Similar to different accelerations, lateral shifts are sampled and combined along with acceleration samples to create final states. Lateral shifts can be mapped either as a function of time or distance traversed by ego vehicle. The research of Werling et al. [59] suggests that at lower speeds it is advantageous to map lateral shift as a function of distance and at a higher speed as a function of time. As this thesis is intended towards urban environments with limited speeds, lateral shift is mapped as a function of distance traversed. Lateral shift planning in this thesis is adopted from [40], which uses cubic splines and models lateral shift as a parameter of longitudinal distance as shown in equation 4.1.

$$l(s) = c_0 + c_1 s + c_2 s^2 + c_3 s^3 \tag{4.1}$$

The first and second derivative of the equation 4.1 are equations for lateral velocity 4.2 and acceleration 4.3.

$$\frac{dl}{ds} = c_1 + 2c_2 s + 3c_3 s^2 \tag{4.2}$$

$$\frac{d^2 l}{d^2 s} = 2c_2 + 6c_3 s. \tag{4.3}$$

From the boundary conditions (0- initial state, f - final state), we have

$$l(s_0) = l_0, l(s_f) = l_f \tag{4.4}$$

The angle between the road frame and the vehicle is defined as $\theta(s)$, it can be derived from the first derivative of the lateral shift with respect to s.

$$\theta(s) = arctan(\frac{dl}{ds}) \tag{4.5}$$

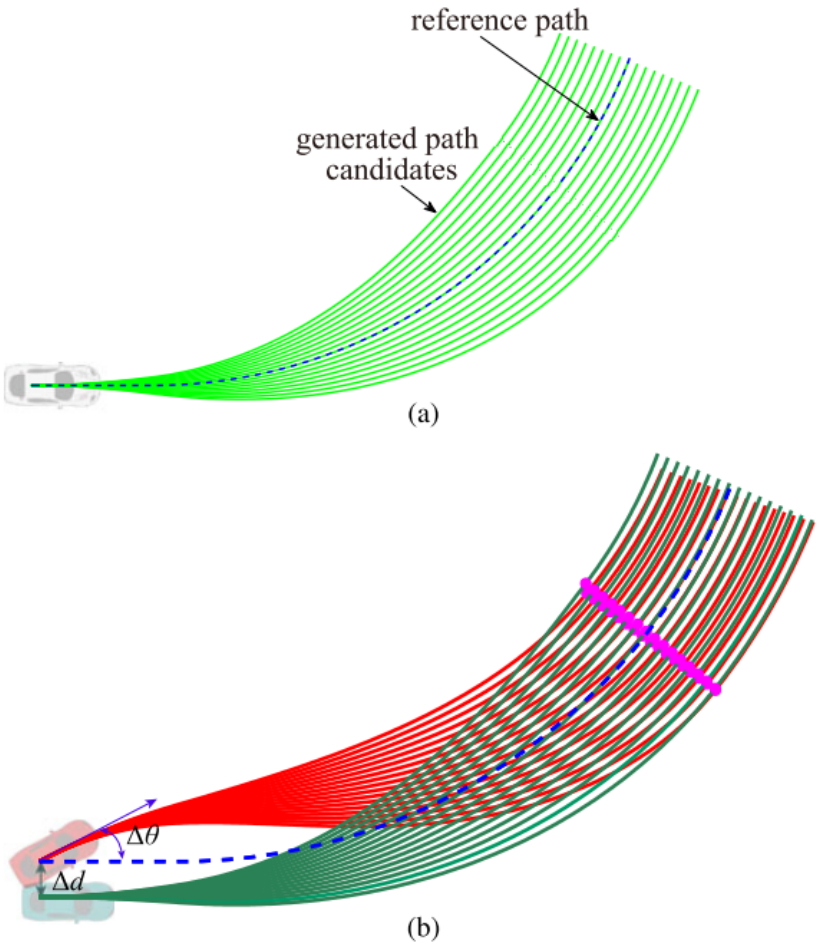


Figure 4.10: Path candidates generation results. (a) Road parallel trajectories $l_0 = 0$ and $\theta_0 = 0$ (b) ego shifted with respect to center and driving at an angle $l_0 = \Delta_d$ and $\theta_0 = \Delta\theta$. [40]

To ensure the generated path follows current curvature and orientation of car and the final orientation is parallel to the road segment, following conditions should be satisfied.

$$\theta(s_0) = \theta_0, \theta(s_f) = 0 \tag{4.6}$$

The figure 4.10 indicates how the initial orientation will affect the shape of the trajectory.

The constants c_0, c_1, c_2, c_3 in equation 4.1 can be obtained by solving equations 4.2 to 4.6.

The Figure 4.11 represents final search space by ego vehicle in XYT space, the density of this space can be increased by increasing the acceleration profiles and lateral samples.

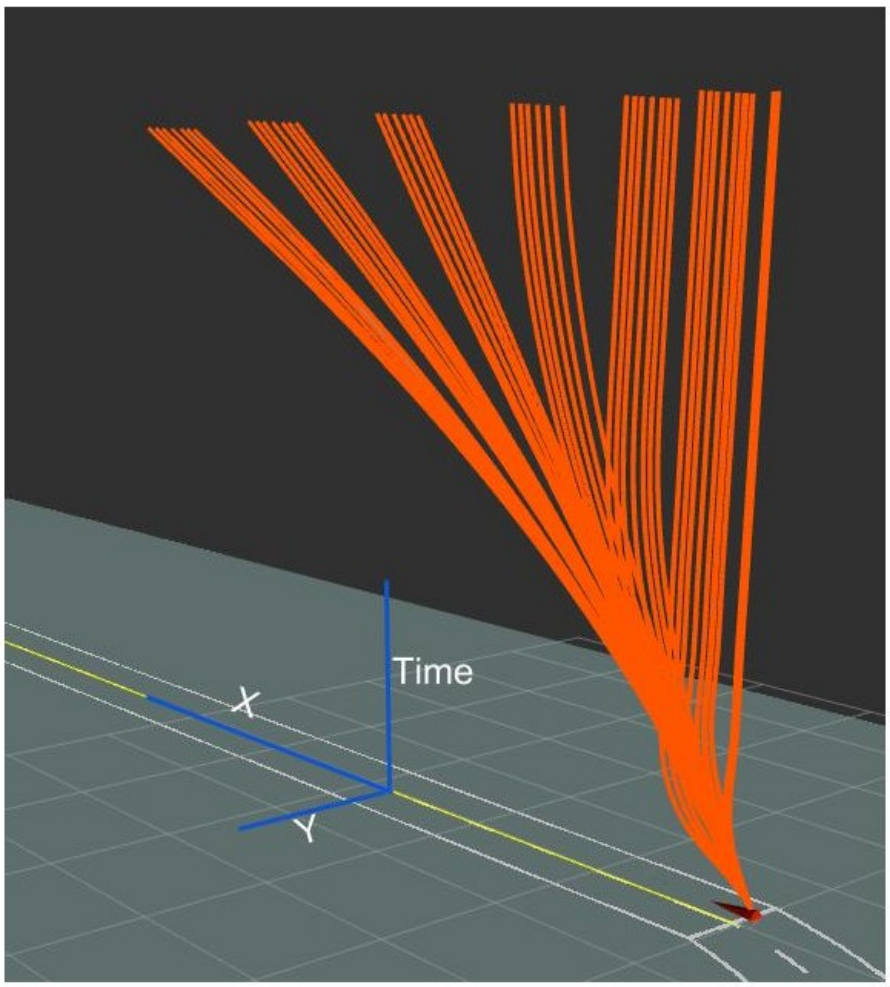


Figure 4.11: Total search space in xyt

In summary, combining samples in acceleration and lateral shifts, multiple trajectories with different final states are created over the time horizon. In the next subsections how these trajectories are tested for collision with respect to static and dynamic obstacles is discussed.

4.5.4 Checking for Static Obstacles

The primary objective of the motion planner is to derive a path that is collision free. The generated trajectories must be evaluated for collision or driving close to obstacles. There are various techniques for collision detection as discussed in the background study. This thesis developed a simple two-step collision checking technique for static obstacles. The obstacles are represented as rectangles (dilated enough to fit complete obstacle), and orientation of obstacles is divided into three categories, they are obstacles moving along, opposite or across the road.

Initially for collision checking, obstacle coordinates are transformed into Frenet frame and represented by a length and width parallel to the road. In the first step, the trajectory in consideration is checked if it has a collision in longitudinal dimension (S coordinate) for the length of the obstacle with formula 4.7. As shown in Figure 4.12 trajectories T0, T1 intersect in S-dimension for obstacle O1 and not for obstacle O2. The next step is to find this intersection region, I1 to I2 in Figure 4.12 represent the S-dimension intersection, the values are dilated for safety. In final it is checked whether from I1 to I2 there is a collision in lateral dimension(d) for trajectory and obstacle. For all points between I1 and I2, the lateral distance between trajectory and obstacle must always be larger than safety value as described in 4.8. As the function representing lateral motion is a continuous function, it is sufficient to check for collision at the start and end of the intersection interval and at critical points (where derivative of the function is zero or does not exist) if they fall in the intersection region. If the distance at any of these points is less than the minimum required or difference vector has different sign then there is a collision. Appendix A.1 details further on sufficiency to check only at critical points and borders.

$$\begin{aligned} intersection = [&max(min(obst_s.begin(), obst_s.end()), path.front()), \\ &min(max(obst_s.begin(), obst_s.end()), path.back())] \end{aligned} \tag{4.7}$$

If $(intersection[1] < intersection[0])$ then there is no intersection in the two paths.

$$|d_{\text{ego}} - d_{\text{obst}}| > car\_width/2 + obstacle\_width/2 + safety\_margin \tag{4.8}$$

It is representative from figure 4.12 that the trajectory T1 has collision and trajectory T0 has no collision. Different costs can be added based on how close the ego vehicle is with respect to the static obstacle.

4.5.5 Checking for Dynamic Obstacles

In dynamic environments such as cities, it is vital to plan collision-free trajectories by predicting the state of other obstacles. Predicting the future behaviour of obstacles is a challenging part of urban driving, various approaches used are described in subsection 2.3. This thesis models dynamic obstacles as squares continuing with their current speed in their detected lane/lateral position for planning duration similar to [5] or moving in opposite direction or moving across the lane based on the angle between the obstacle and road. This assumption can be justified by the fact that the trajectories are re-evaluated at high frequencies and any changes in obstacles lateral distance, orientation or speed will be evaluated in next cycle thus keeping the vehicle safe from the collision.

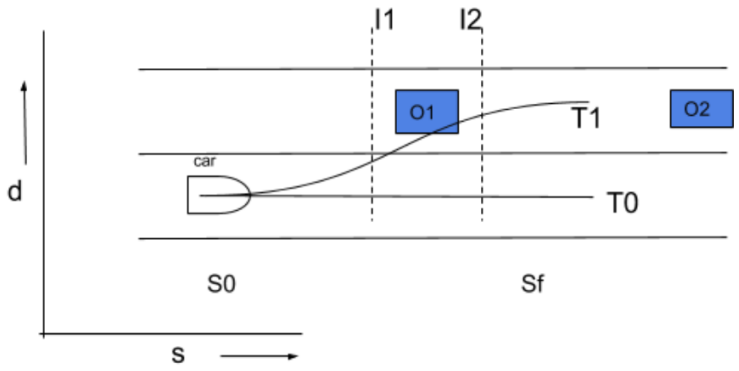


Figure 4.12: Collision check for static obstacles, I1-I2 represent collision region in S-dimension, in this region T1 intersects with trajectory and T0 is collision free.

The collision check for dynamic obstacles has one extra check-in time dimension as compared to checking for static obstacles, it is inspired by forbidden regions calculation as discussed in [21]. In step one the intersection in S coordinate for obstacle and ego vehicle is found, here the length of the obstacle is dilated over the distance travelled by obstacle as represented by dotted line ahead of the obstacle in figure 4.13. If there is a collision in S dimension, then the collision between ego vehicle and obstacle in lateral dimension (D) in the intersection region I1 to I2 is tested in a similar way to static obstacle collision check. If there is a collision in D dimension, then the corresponding S dimension where there is collision is found, represented with J1-J2 in figure 4.13 (generally this will be shorter than I1 - I2). For the range J1-J2, it is checked if they collide in time also, i.e., if they reach the same location in same time, a buffer time is added to be safe. As per instructions for safe driving it is required for the car to maintain a minimum time gap of 2 seconds with the vehicle ahead. There are more formal methods [55] on safety distances for self-driving cars. This thesis implements a simple 2-second rule to safety, and this is a tunable parameter which can be used to increase driving aggression. Figure **??** indicates a similar situation for the collision in time dimension for scenario presented in Figure 4.13 with the difference that the obstacle is in a different lane.

To check collision in time dimension lets initially consider the ego to be a point moving on road, method to add the length of the car to checking is indicated in the final portion of the section. After finding the intersection region as mentioned above, time difference between the ego vehicle and the obstacle at the S dimension intersection borders is tested as shown in figure 4.14. If the gap is larger than 2 seconds at all the times and doesn't change sign then there is no collision, Figure 4.14 a) b) present situation where the time to the collision between ego vehicle and obstacle reduces but there is no collision. If the time difference between ego vehicle

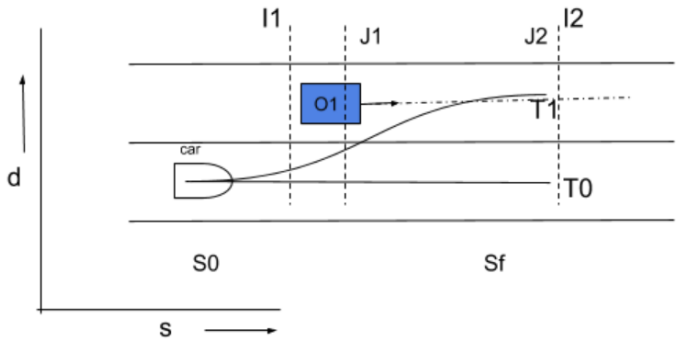


Figure 4.13: Collision check for Dynamic obstacles, I1-I2 represent collision in S-dimension, J1-J2 indicate collision in S and D dimension.

and the obstacle has an opposite sign at the start and end of the intersection region, then it detected as a collision. As difference indicates time to collision, and as time and distance are continuous. A change in sign in time difference demonstrates that in between at some point there was a collision (time difference = 0). Figure 4.14 c) and d) present a situation where the projected trajectory of the ego vehicle will collide with a dynamic obstacle. Figures 4.14 e) represent a case where the planned trajectory of ego vehicle collides with obstacle moving in opposite direction and Figure 4.14 f) represents where the ego vehicles projected trajectory will not collide the vehicle in reverse lane.

The method discussed till now represent the ego and the obstacle as points moving in space-time, but both of them have lengths. Thus, the time at the beginning and end of intersection region should consider the length of these objects, i.e., time for ego or obstacle to cross this point should be considered. As presented in Figure 4.15, time gap at the start of intersection is the minimum of the four-time gaps formed between front of the ego and front of the obstacle, front of the ego and back of obstacle, back of the ego and front of the obstacle, back of the ego and back of the obstacle to cross the beginning of the intersection region. Similarly, minimum of these four-time gaps are considered to find the time gap at the end of the intersection, it is represented by green arrowed lines in Figure 4.15.

Dynamic obstacle moving laterally across the road are represented as static obstacle occupying the length of the road. Thus, if there is a pedestrian crossing the road, the trajectory is considered to be in the collision if there is a collision in S dimension and there will anyways be a collision in D dimension due to dilation of the pedestrian width to the width of the road and no time dimension is checked. This could, however, be improved by checking the direction of walking and also predicting the time to cross the road to check for collision.

In the Figure 4.14 motion of the ego vehicle is approximated with linear velocity,

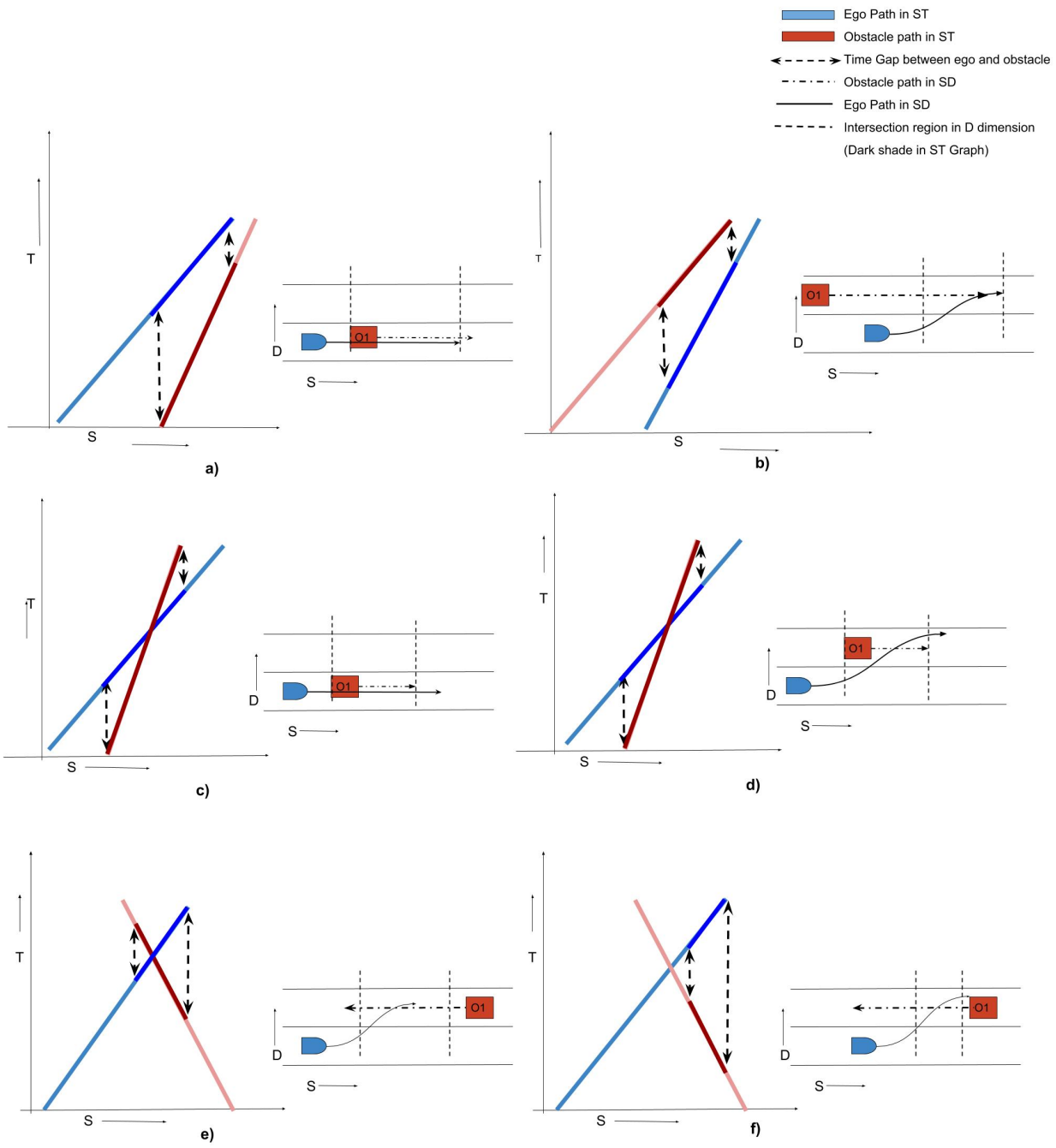


Figure 4.14: a),b) describe situations where ego vehicle's projected trajectory is close to obstacle ahead and behind respectively, but not colliding.
c),d) describe situations where ego vehicle's projected trajectory collides with obstacles ahead and obstacle in next lane respectively.
e) describes situation in which ego vehicle's projected trajectory collides with obstacle moving in opposite direction and f) describes situation where it will not collide.

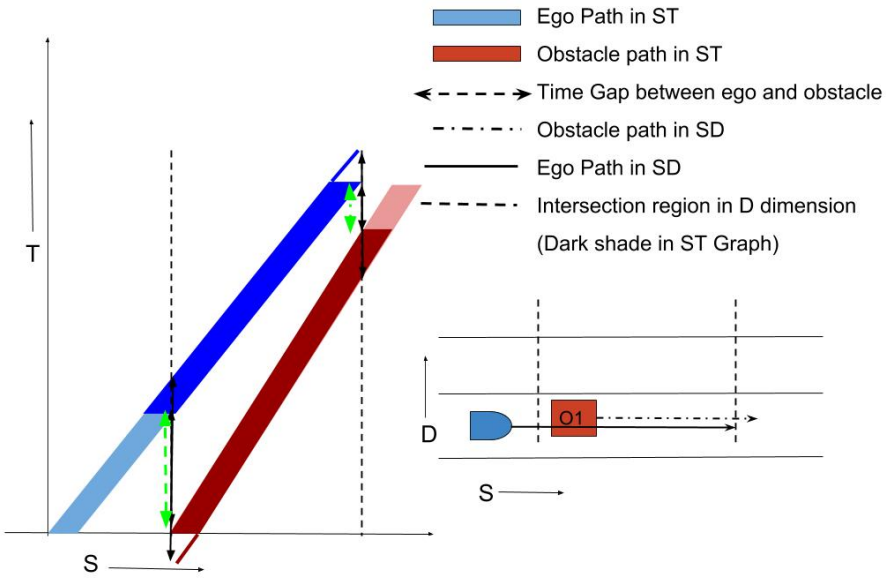


Figure 4.15: Collision checking by considering the length of the car and the obstacle. Time gap at the intersection is the gap between the front of ego and back of ego to cross that point, represented by green arrowed lines.

but in reality, this is not the case, ego vehicle follows a parabolic path with respect to time because of acceleration and deceleration. Figure 4.16 presents a scenario where the ego vehicle with $0.1ms^{-1}$ velocity is accelerated at $0.2ms^{-2}$ represented with a blue line and an obstacle moving at $0.6ms^{-1}$ represented by a maroon line. In this situation at the start and end of the intersection region time difference between the ego and obstacle is positive (green lines) but there is a collision in the middle of the path. The algorithm mentioned above could not detect this collision. There are two ways to mitigate this effect. One is to have a substantial enough buffer time which covers the non-linearities caused by the parabolic path. In the Figure 4.14 orange line between the linear approximation path (Yellow line) and ego path (blue line) represents the maximum difference caused by non-linearity. It is around $0.8s$, and this thesis employs a two-second rule to detect a collision. Thus, if the time difference between the ego and the obstacle intersection region is greater than two seconds at borders, then there is no collision between the two, at some point in the middle they may come close but will not collide. Appendix A.2 provides proof about this hypothesis being correct. Another way to mitigate this issue is to check at multiple points in intersection region to determine collision. The second method can be employed when aggressive driving is needed, and distance to obstacles is kept to a minimum.

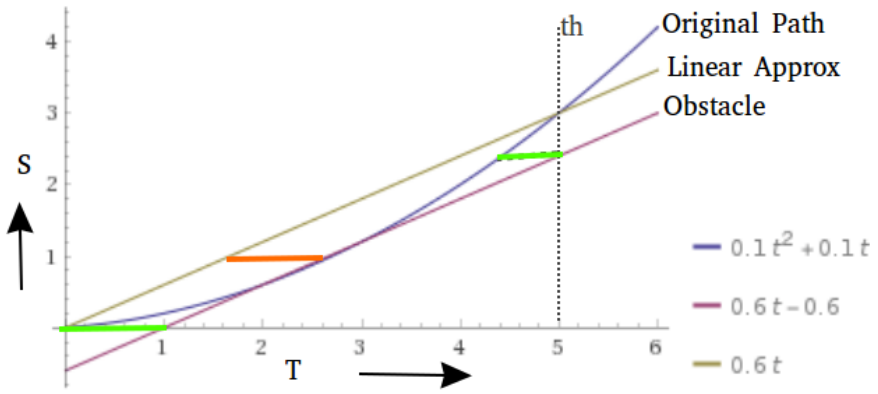


Figure 4.16: Parabolic Path and collision checking

4.5.6 Cost Functions and Trajectory Selection

Selection of final trajectory is based on the different costs associated with it, and costs can be static and dynamic. Static costs are known before trajectory creation and dynamic costs are known after the trajectory is created and evaluated. There is wide research on different costs involved in trajectory selection as discussed in section 2.3

This thesis implements a simple cost function as represented in 4.9 for calculating pre-cost of samples.

$$initial_cost = |V_a - V_t| + |a_t| + |(d_t - d_e) * k1| + |(d_p - d_e) * k2| \tag{4.9}$$

V_a - velocity achieved by trajectory. V_t - Target Velocity. a_t - Target Acceleration. d_t - Target lateral distance. d_e - Trajectory lateral distance. d_p - Previous target lateral for trajectory. k_1, k_2 - Factors to adjust weights, currently used at 0.8 and 0.2

The velocity and acceleration terms in cost function 4.9 promote higher accelerations if the target and current velocity difference are higher and lower accelerations if the difference is low. The next lateral terms promote the trajectories that are closer to the target lateral distance and also close to the previous selection thus limiting the shifts in path selection and maintaining continuity. The initial cost of the previous path is kept to zero to promote ego to follow old path if there is no collision.

The final cost for trajectory selection is the sum of the initial cost, collision detection cost for trajectory and horizon cost as mentioned in equation 4.10. Horizon cost is added to samples if the created trajectory crosses the destination thus promoting the trajectories that stop at the goal. Explicit smoothness costs are not considered as the target validation platform is a model car with no humans inside and limitations of the model car to track a fine trajectory.

$$final_cost = initial_cost + collision_cost + horizon_cost \tag{4.10}$$

Initially, all the sampled trajectories are assigned the costs based on the cost function 4.9 and sorted, then the trajectory with the lowest cost is evaluated and the final cost is calculated as per equation 4.10. The list is sorted again and evaluated till the top of the list has the lowest cost, and the trajectory is evaluated. Thus the trajectories with initial assumption are validated, and best one is chosen in the end.

4.5.7 Velocity Planning

Velocity planning is an important aspect of the planer, in general, behavioural layer defines the target velocity based on speed regulation, other traffic participants, required behaviour, road condition etc. In this thesis, a simple approach of velocity limiting is used based on road curvature to limit lateral accelerations. Max velocity V_{max} is calculated based on the equation 4.11.

$$V_{\max} = min(\sqrt{Acc_{\text{MaxLat}}/|k(s)|}, V_{\text{limit}}, V_{\max}) \tag{4.11}$$

V_{limit} - Velocity limit mentioned in road network, Acc_{MaxLat} - Maximum Lateral acceleration(based on comfort and vehicle dynamics),$k(s)$ - Road curvature and $V_{\max}$ defines limit set by user.

4.6 Trajectory Follower

The controller for following the trajectory is adopted from Made In Germany (MIG) autonomous vehicle. The work [20] details further on how the trajectory is decomposed into steering and speed commands.

Chapter 5
Implementation

This chapter details on the implementation of the concepts discussed in the previous chapter for route planner and trajectory planner modules. All the modules are implemented as independent nodes in Robot Operating System (ROS) and communicate with each other using ROS messages. Figure 5.1 describes the communication across different modules.

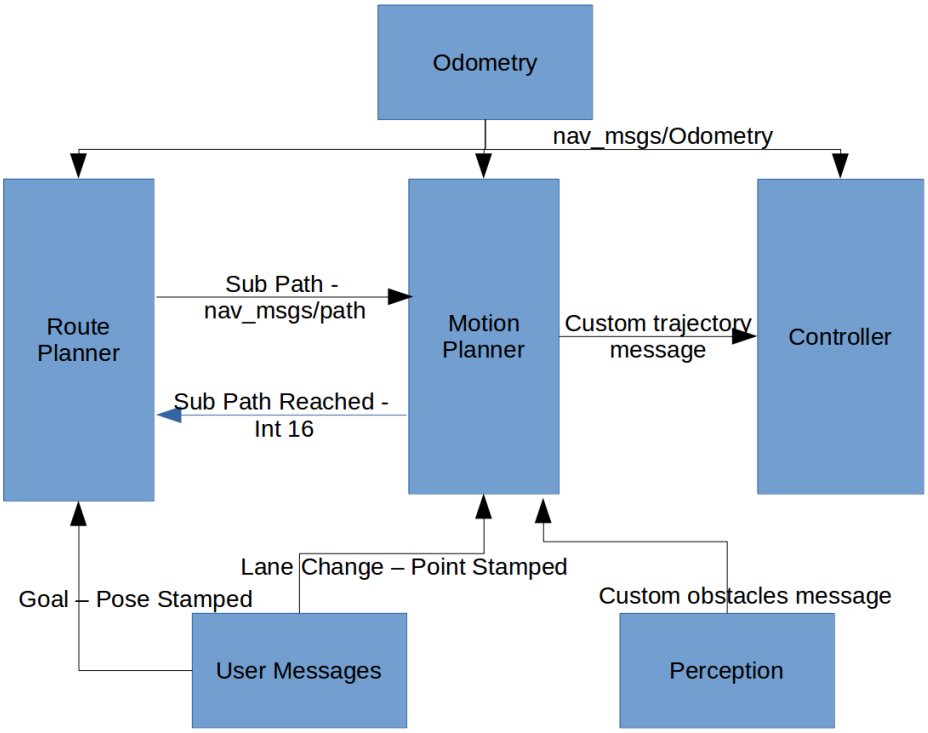


Figure 5.1: ROS Messages Flow across Modules

5.1 Route Planner

Route planner parses the RNDF file and stores waypoints, their relation to lanes and further to segments. The below code segment shows the declaration of the waypoint, here coordi indicates the coordinates of waypoints, idn is the unique id for each waypoint and parent is the lane to which waypoint belongs. Similarly, lane stores set of waypoints, the parent is the segment to which the lane belongs.

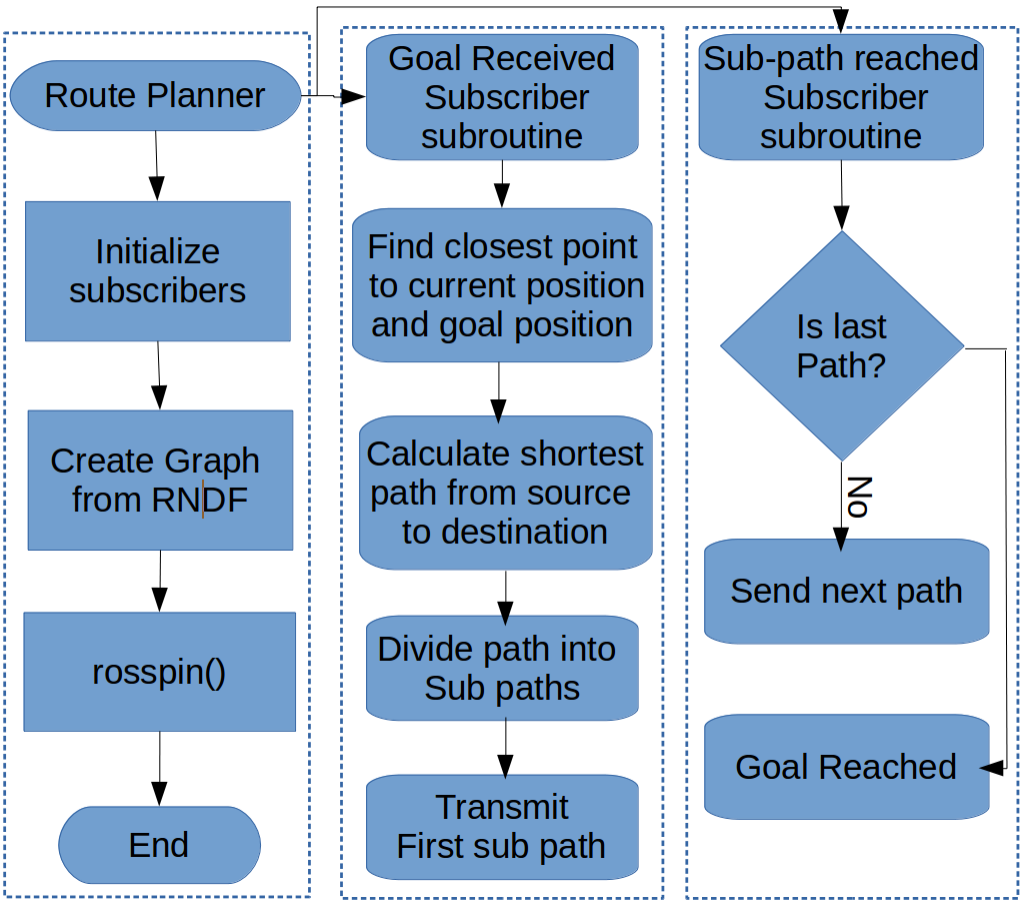


Figure 5.2: Route Planner Functions

```
class c_waypoint(object):
  def __init__(self,name,coordi,parent, idn):
    self.name = name
    self.coordi = coordi
    self.parent = parent
    self.idn = idn
  def __str__(self):
    return '{}'.format(self.name)
```

5.2 Motion Planner

Motion Planner is subdivided into three subclasses namely motion planner, vehicle path and vehicle state. Vehicle path class stores the path to be traversed, performs all the conversions from Cartesian frame to Frenet frame and vice versa. The module

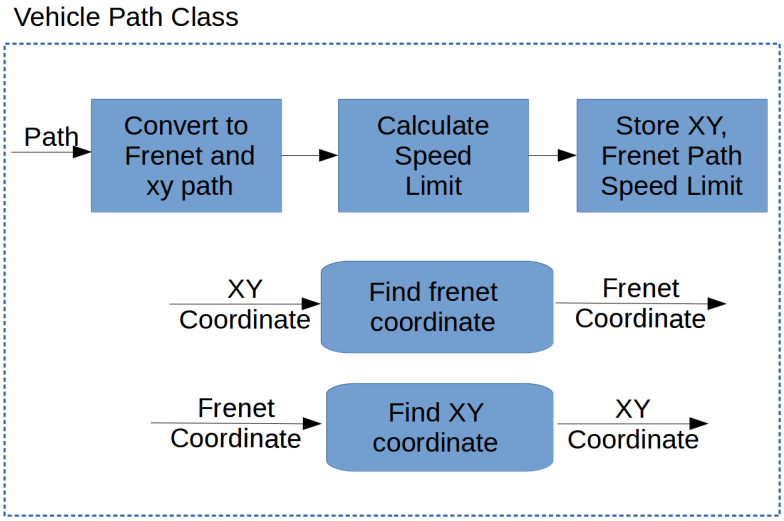


Figure 5.3: Vehicle Path Class and it's sub functions

also calculates speed limit based on the curvature of the road. Figure 5.3 further details on the structure of the vehicle path class.

Vehicle state class maintains the information regarding the current position and, orientation from odometry messages, and list of obstacles including information about their position, orientation, speed, size. In every planning cycle, initially, a snapshot of vehicle state is used to create trajectories and evaluate them to remove effects of changes in states in one planning cycle. Figure 5.4 details further on structure of the Vehicle state class.

Motion planner class is the core of the planner, it initially creates samples, assigns pre-costs to these samples, creates trajectories based on the samples and pre costs, evaluates the trajectories to find the best trajectory which is collision free and closer to the destination. Once the best trajectory is found it is converted to the controller-specific format and transmitted. Figure 5.5 describes the interactions between different functions in motion planning.

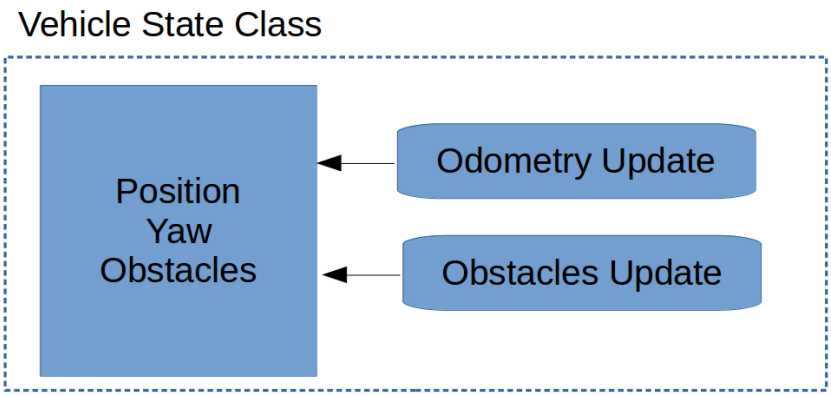


Figure 5.4: Vehicle State Class and it's sub functions

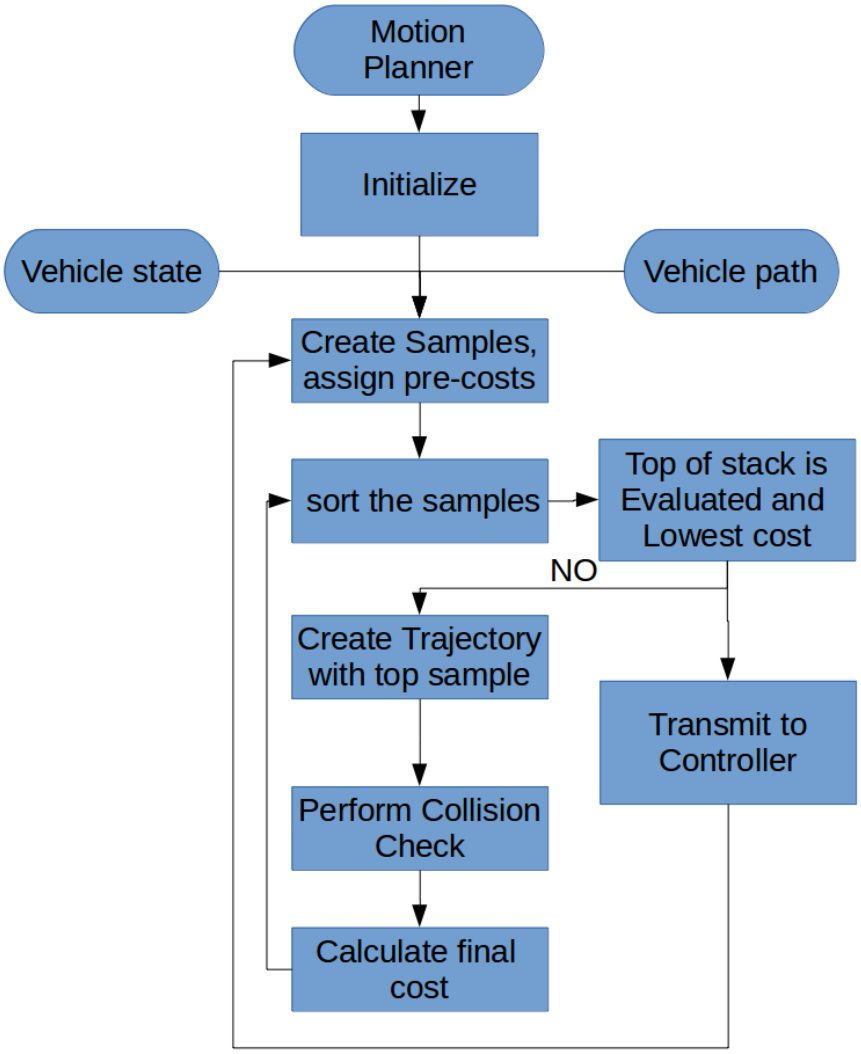


Figure 5.5: Motion Planner Functions

Chapter 6
Evaluation

In previous chapters detailed working of the planning algorithm has been discussed, this chapter discusses the evaluation criteria and results in detail. Various concepts discussed previously will be examined here through a series of experiments reflecting real-life driving scenarios. This chapter is organised as follows: Section 6.1 discusses the systematic evaluation of the planner by exposing it to various situations equivalent to on-road driving conditions. The next section 6.2 discusses a criteria-based assessment for the planner by examining factors such as feasibility, optimality, completeness, run-time etc.

6.1 Experiments

Due to time and resource constraint, most of the experiments to evaluate the planner are performed on the simulator. The test cases involve finding a collision-free path with obstruction in driving lane, avoiding slow moving traffic, merging into ongoing traffic, lane changes etc. The following subsections detail further about each experiment.

6.1.1 Lane blocked

The scenario presented in 6.1 describes a situation when the driving lane is blocked by a static obstacle, and there is a vehicle driving in next lane. The ego vehicle slows down till the left lane is free, overtakes the slow-moving obstacle and shifts back to original lane and drives at increased speed.

6.1.2 Overtaking and Merging into Traffic

Overtaking slow moving obstacles and merging into traffic are two scenarios which are always encountered by the vehicles during on-road driving. The planner proposed in this thesis is capable of performing these two behaviours to enable autonomous driving.

In the scenario presented in Figure 6.2, there are two slow-moving obstacles ahead of the ego vehicle. Initially, ego-vehicle picks up the speed but does not change lane as staying in the same lane is prioritised over accelerating higher. Once ego vehicle is close to the obstacle, it has to brake or shift lane, as shifting lane is prioritized over

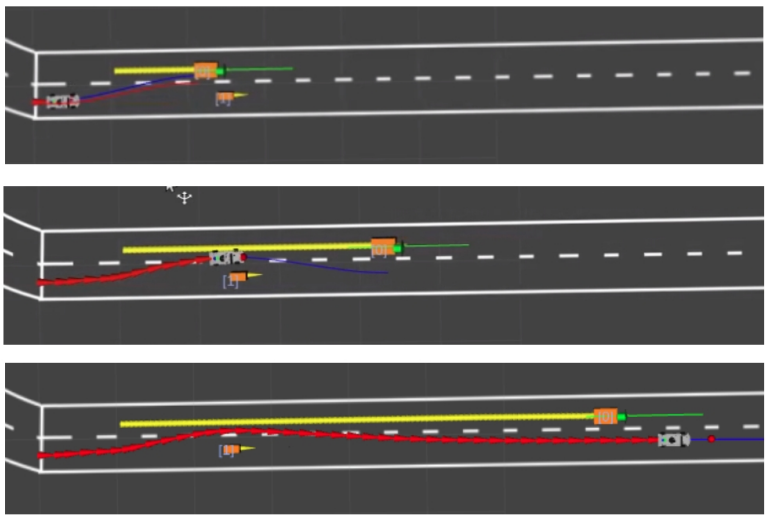


Figure 6.1: Driving Lane is blocked by a static obstacle with an obstacle moving in next lane. The ego vehicle avoids the static obstacle by driving into next lane and comes back to intended lane once it is free

braking ego vehicle shifts into the left lane and starts accelerating. The velocity of obstacles is $0.3ms^{-1}$.

Another general behaviour encountered by vehicles during on-road driving is merging between two obstacles. In the situation presented before, once the ego overtakes first slow-moving obstacle, it can either merge into the gap between the two obstacles as presented in Figure 6.3 or follow the left lane until the second obstacle is cleared as presented in Figure 6.4. The cost functions are tuned such that driving in intended lane (provided by behavioural layer) is preferred over accelerating, acceleration or driving in non-intended lane is preferred over decelerating. These cost functions create behaviours of merging or non-merging. The Figure 6.5 presents velocity change over time for the behaviour of overtaking and merging into the left intended lane. Though the planner produces smooth velocity trajectories as presented in Figure 4.9, due to limitations in controller and platform the final behaviour is not smooth.

6.1.3 Merging into next lane with opposite traffic

In the scenario presented in Figure 6.6 driving lane is blocked by a series of obstacles and the left lane is occupied by a moving obstacle. Ego vehicle starts slowly in the driving lane and waits till the obstacle is passed in the left lane and starts driving forward.

In the scenario presented in Figure 6.7, ego vehicle has enough slack to avoid the series of static obstacles and get back into the right lane avoiding a collision with the

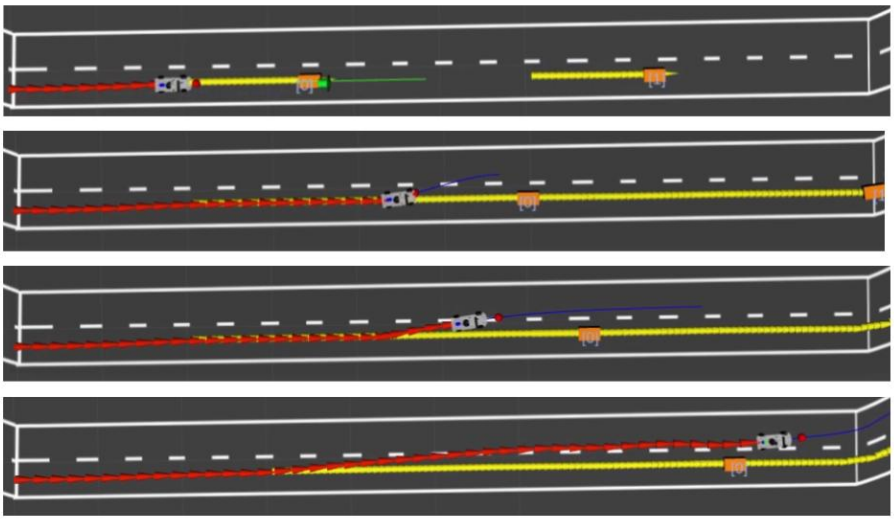


Figure 6.2: Ego vehicle initially picks up the speed, when the obstacle is in collision range ego vehicle starts shifting into left lane and overtakes the slow moving obstacle

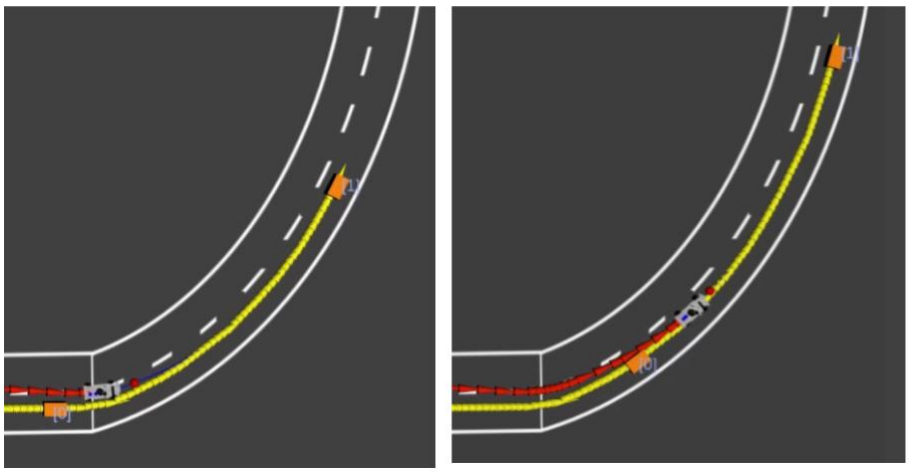


Figure 6.3: After overtaking first obstacle as mentioned in 6.2, ego vehicle merges into right lane between two vehicles.

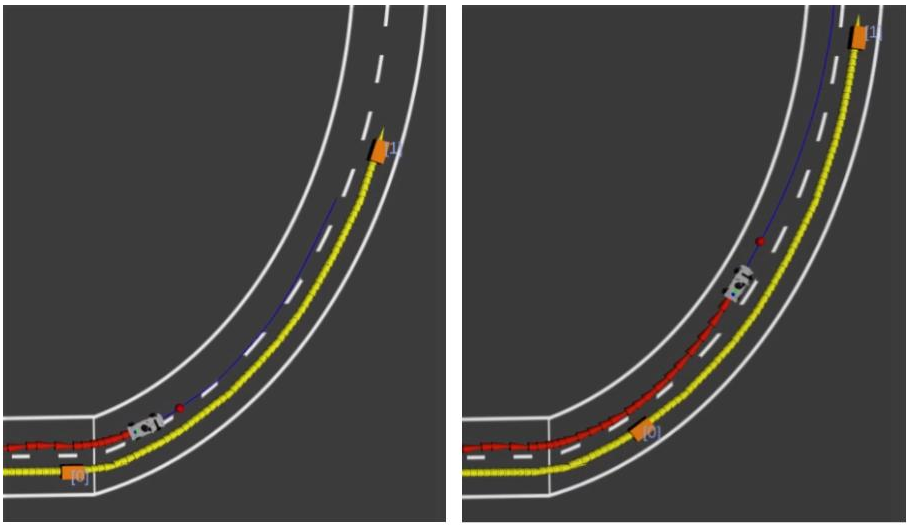


Figure 6.4: After overtaking first obstacle as mentioned in 6.2, ego vehicle does not merge into right lane and continues in left lane.

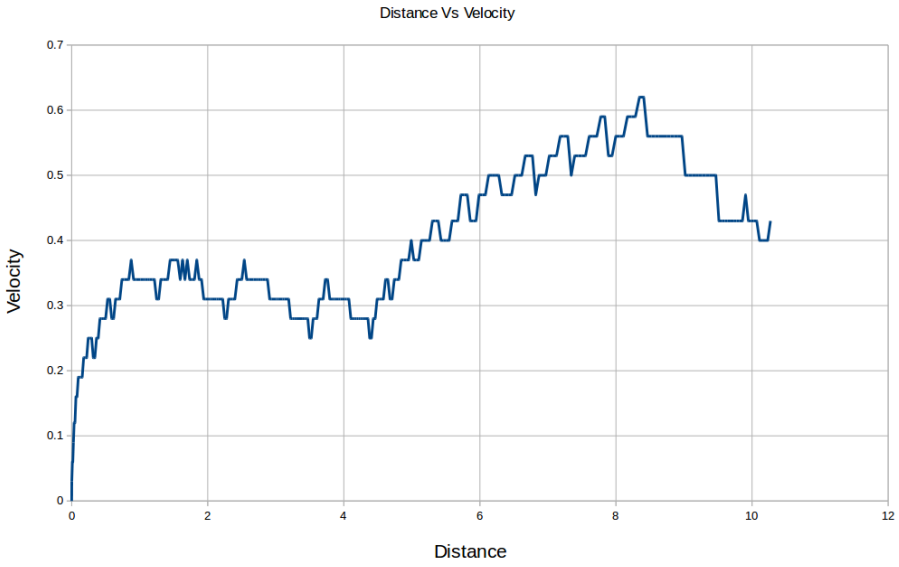


Figure 6.5: Velocity variation of ego vehicle with respect to time for overtaking and merging back into right lane, figures 6.2, 6.3 indicate behavior of car on track.

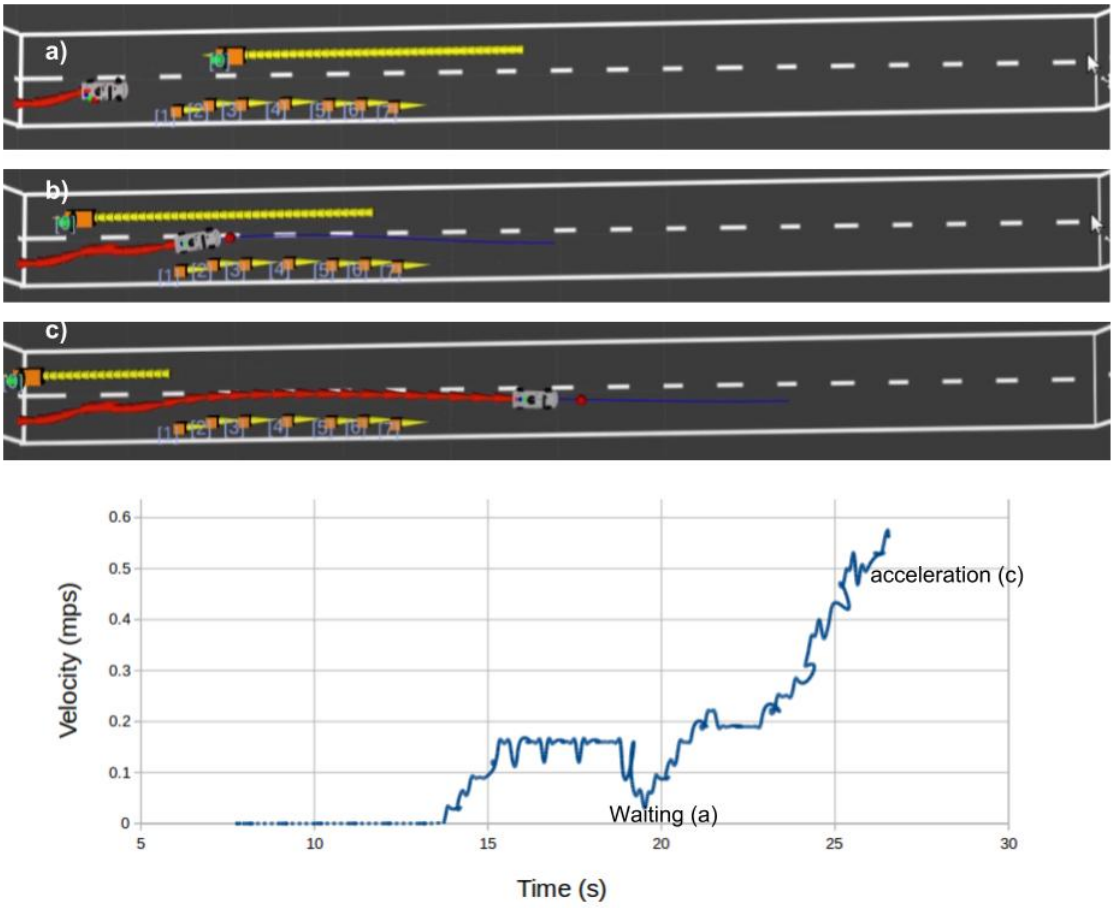


Figure 6.6: Lane blocked by series of static obstacles and vehicle in next lane driving opposite. Ego vehicle waits for the vehicle to pass then drives ahead avoiding the static obstacles. The time vs Velocity (smoothened values) charts shows how the ego vehicle starts driving, then waits for obstacle to pass and continues accelerating further after passing obstacles.

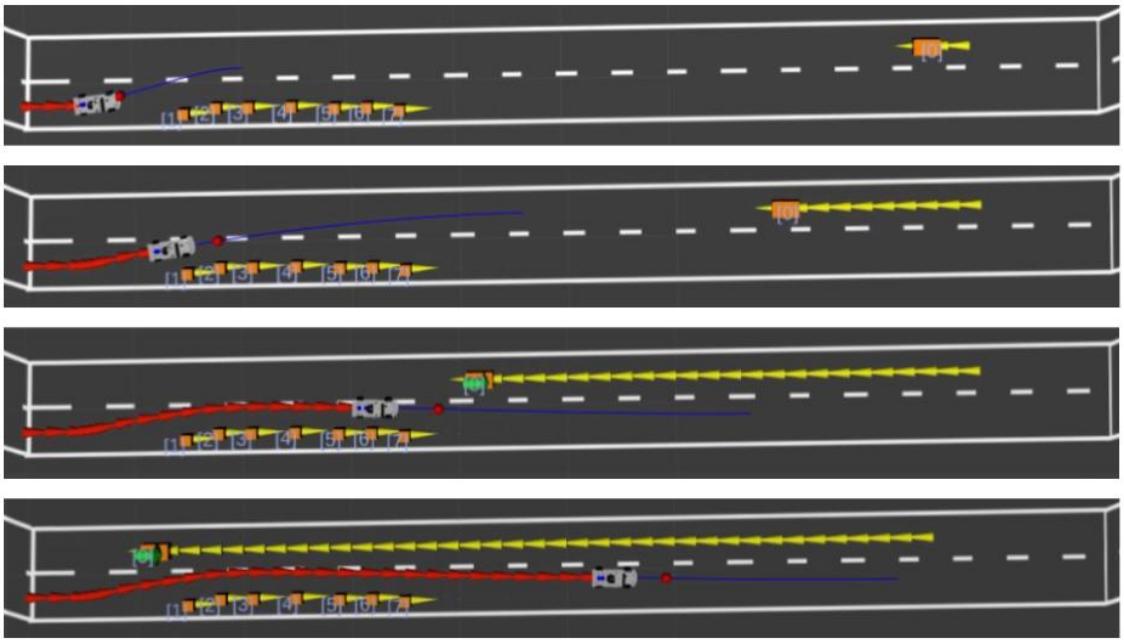


Figure 6.7: Obstacle is far away on the opposite lane, ego vehicle gets enough time to overtake the obstacles and shift into right lane.

dynamic obstacle ahead.

The ego vehicle may get stuck in the middle of the road if driving speed of ego vehicle is low. Because of short temporal horizon even though it takes more than temporal horizon time to complete the maneuver will be initiated by the planner and it may not see the fast moving dynamic obstacle faraway and starts to change the lanes. Once it's in opposite lane and sees a fast moving obstacle, ego will not to shift to right lane because of obstacles and will stop. Only if the opposite vehicle stops and provides enough slack ego will be able to drive forward. The planner does not allow reverse driving thus the ego has to switch to a different planner if the ego has to drive backwards in this scenario.

6.1.4 Road Blocked or Pedestrian Ahead

In the scenario presented in Figure 6.8, the road is blocked by a series of static obstacles, the ego accelerates initially when obstacles are far away, then slows down and finally stops when it cannot find route ahead.

A pedestrian on the road is considered similar to a road blocking, in this thesis pedestrians moving across the street are set to occupy the width of the lane. In the situation as shown in Figure 6.9 the ego vehicle initially drives at full speed, then the ego slows down (shorter blue line representing a slower speed), and the robot finally comes to a halt few meters ahead of the pedestrian. Buffer distance is a tunable

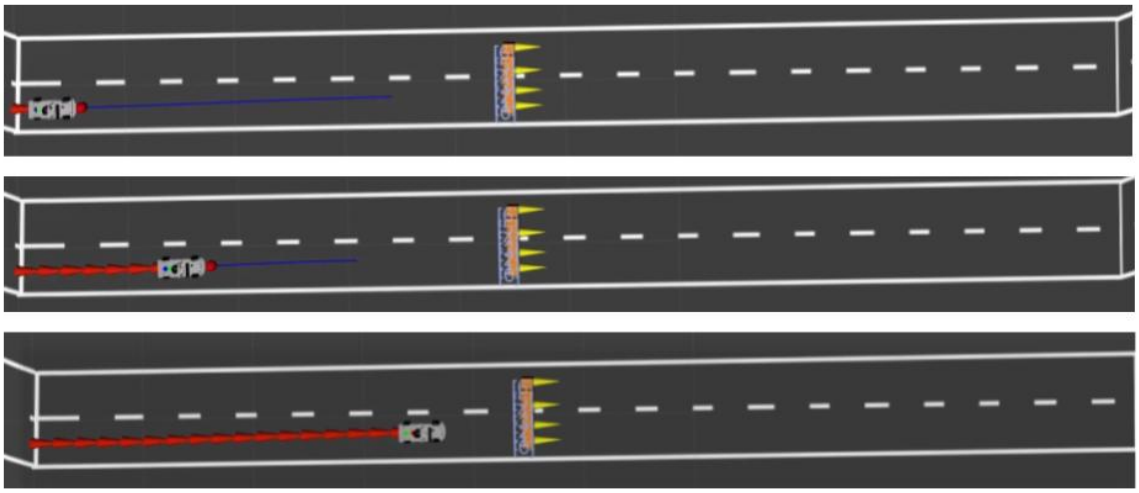


Figure 6.8: Road Blocked by series of obstacles, ego vehicle initially plans a path with acceleration, as it nears obstacles it slows down and finally stops.

parameter and currently at the maximum value for safety.

6.1.5 Dynamic Obstacles - other vehicles

The main objective of the planner is to adjust to the sudden changes in the environment caused by the dynamic obstacles in surroundings, here two sub-scenarios are described where the ego vehicle has to react to an unexpected breaking of vehicles ahead.

In the scenario presented in Figure 6.10, the obstacle ahead stops suddenly, and the ego vehicle has to find a path avoiding a collision. This situation is similar to when a car ahead stops to drop off a passenger or for a parking spot. In this situation, ego follows the obstacle ahead, once it stops ego slows down till it finds enough room in the left lane to drive ahead of the stopped dynamic obstacle and continues driving.

In the scenario presented in Figure 6.11 two dynamic obstacles are moving ahead of the ego and threshold for safety has been chosen to be 1s. The obstacles ahead stop suddenly representing a situation of a crash for the vehicle forward or emergency stop and ego vehicle comes to a halt in next 2s. Higher deceleration profiles used in planning allow for ego vehicle to act reactively and stop without collision.

6.2 Criteria Based Evaluation

In this section, the proposed planner is validated against the common criteria of evaluating any algorithm, i.e. optimality, feasibility, completeness, runtime, and final review of approach. Each subsection details on advantages and limitations of the planner concerning each criterion along with short comparisons to other approaches.

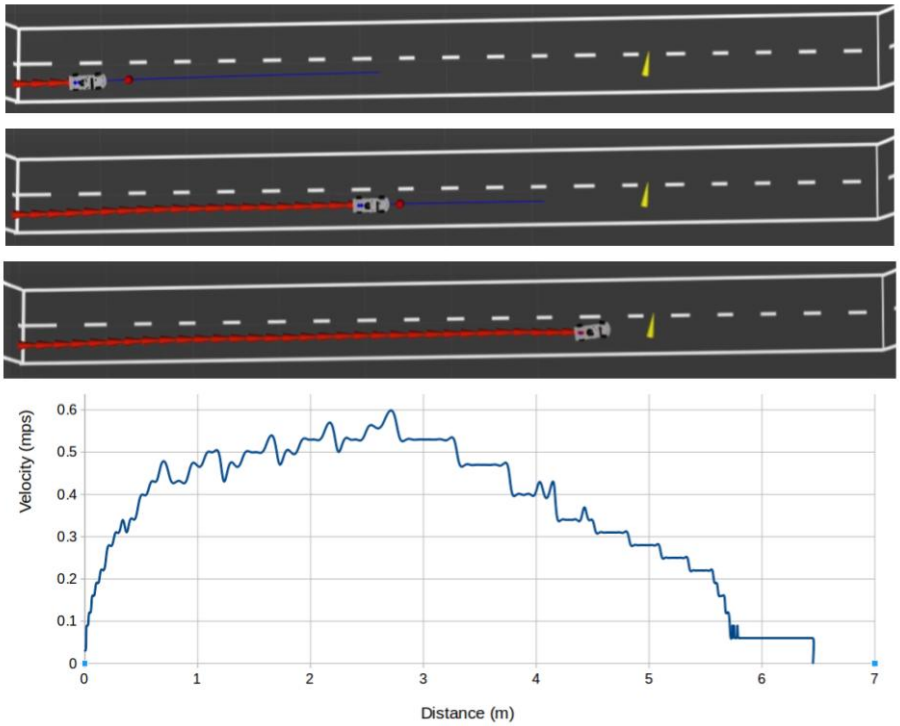


Figure 6.9: Pedestrian Ahead on Road, Pedestrian is assumed to occupy lane width. Ego vehicles behaves same as lane blocked. The distance Vs velocity graph shows how the ego accelerates, then slowly decelerated and finally halts. Ego stops with 1m gap to pedestrian.

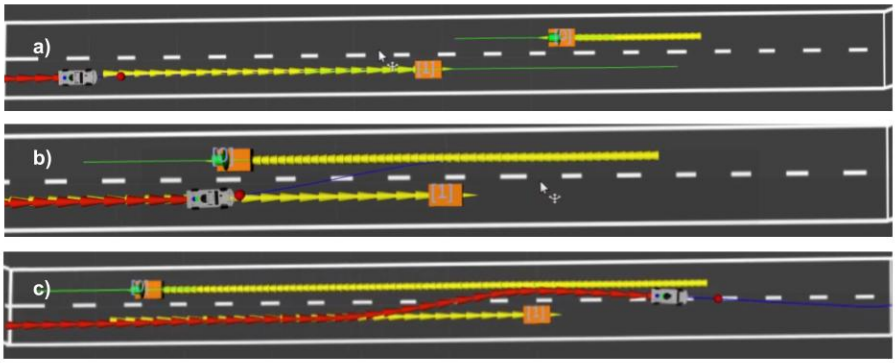


Figure 6.10: The scenario has one obstacle moving along the road and other moving opposite as shown in a), ego slows down as the dynamic obstacle in the same lane stops suddenly and starts planning path in next lane once other lane is free as shown in b) and finally ego returns to original lane as presented in c).

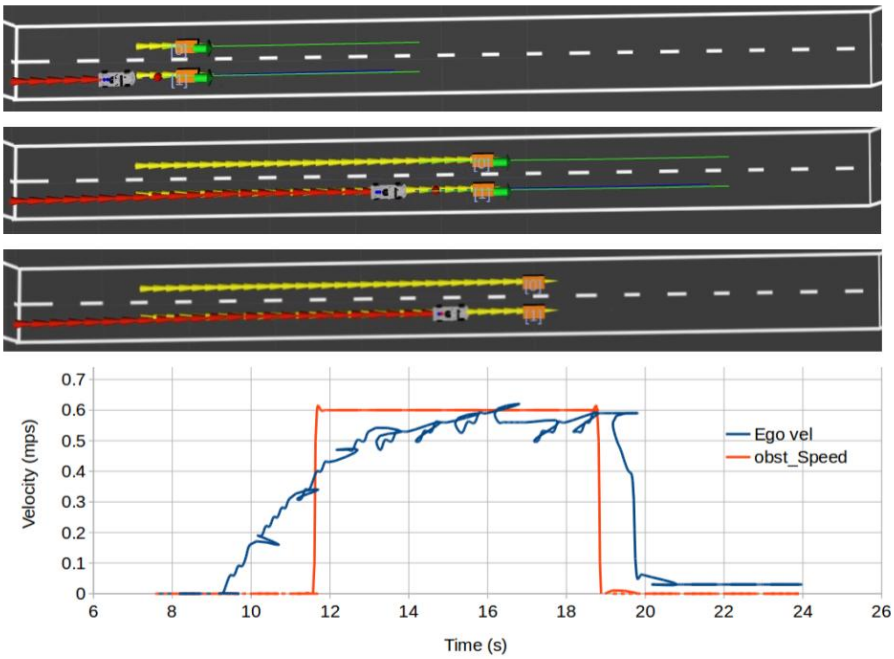


Figure 6.11: Ego vehicle follows two dynamic obstacles moving at $0.6ms^{-1}$ and they stop suddenly, ego vehicle decelerates quickly to stop from colliding. The chart displays velocity variation for ego vehicle with respect to obstacle speed.

6.2.1 Optimality

In this thesis, we discuss the optimality of the time horizon, subsection 4.5.1 already defines regarding various timing constraints chosen in this planner. A larger planning horizon will enable the planner to create a longer and better path, but due to the unpredictability of the environment, the plan created will not be valid after a specified duration, a broader horizon will also increase the run-time of the algorithm. The planner proposed in this thesis is only a local planner and always needs inputs from a behavioural layer or a global planner to choose target lane, velocity etc. thus a short planning horizon sufficient to bring the ego to a halt is suitable for the proposed planner.

An example of how horizon will affect optimal planning for the current planner is shown in figure 6.12. Here T0, T1 are the trajectories with horizon "T" and T2, T3 are trajectories with horizon "T'". In this condition, if a lane change has been requested then trajectory T1 is chosen, but with increased horizon trajectory T3 will be selected, depending on situation one is efficient sometimes and other in others. These situations can be improved by lane selection algorithm in the behavioural layer which looks for occupancy of different lanes and suggests the one best suitable lane. Similarly, if an exit has to be taken on the road, a long horizon would choose a plan with reduced speed compared to a high-speed path with the short horizon, this

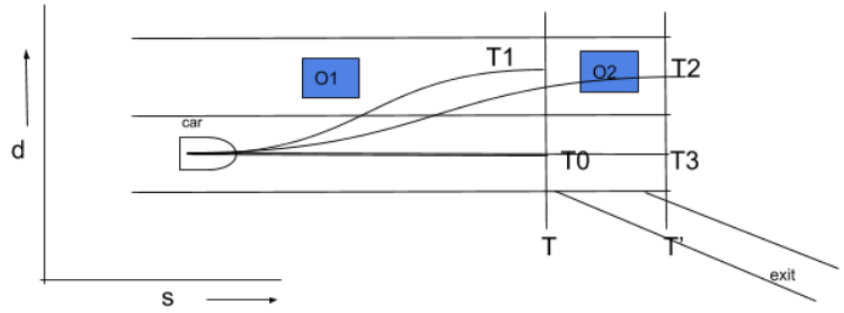


Figure 6.12: Horizon Optimality reference

can also be solved by having a velocity planner in the behavioural layer with more extended scenario analysis algorithm.

Another horizon generally in the discussion for a planner is spatial horizon which discusses how long is the path generated. As per the planning criteria, spatial horizon depends on velocity and temporal horizon, at low speeds spatial horizon is shorter and at high speeds larger temporal horizon. A shorter spatial horizon might result in not having a globally optimal solution, more substantial scenarios analysis will promote a local decision suitable globally. As shown in figure 6.13, following the original reference path(solid line) at low speeds will result in two times collision avoidance and shifting path. With scenario analysis, a new shifted reference path can be chosen which avoids many shifts improving comfort.

The resolution of the sampling in acceleration selection and lateral distance selection will also affect the optimality of planning, a chosen plan can only be optimal of the trajectories created by sampling, higher the number of samples, larger are the possibilities, and the best selection is possible.

In summary, the planner proposed in this thesis is locally optimal and needs support from a scenario analysis algorithm to choose globally optimal paths.

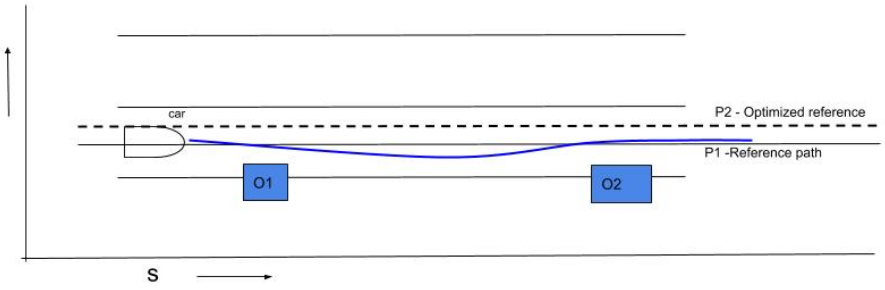


Figure 6.13: With current reference path (solid black line) at low speeds ego will follow path similar to suggested in blue curve. While with an optimised reference (dotted line) ego will move straight without many shifts

6.2.2 Feasibility

The ability of the vehicle to traverse the created trajectory determines feasibility. Generally, the curvature of the path, smoothness and accelerations determine whether a trajectory is feasible or not. The proposed planner creates feasible trajectories at lower speeds because of the third order splines used, and higher speeds require fifth or higher order splines to maintain continuity in path curvature and speed. Discussions in [45] [43] throw light on how to achieve higher degrees of smoothness, which approach is better in which driving conditions. As the intended application for this planner is a modelcar, constant acceleration profiles are used due to a limitation in the ability of the car to track small changes in velocity and inaccuracies in measurement. These can be easily replaced by a smoother higher order polynomial with a better hardware platform. Consistency in paths evaluated with respect to previous plan is another factor in testing feasibility, the current planner penalizes trajectories deviating from previous plan and also take into account current orientation of the vehicle in choosing a path. Thus creating smoother transitions from one state to another by respecting current driving curvature.

In summary, the planner creates feasible trajectories for the modelcar and needs update in polynomials used in trajectory creationa and representation to be used in complete cars.

6.2.3 Completeness

An algorithm is said to be complete if it can result in a solution every time. A motion planner can be called complete if it returns path if it exists in the space searched. Like many other sampling-based approaches the planner proposed in this thesis only probabilistically complete. That is, the probability of finding a solution approaches to one as the number of samples increases. If there are a higher number of samples in the configuration space, the higher are the chances of finding a solution. If a planner cannot find a solution within the sampled region, it forces the ego to perform an emergency braking manoeuvre.

In Figure 6.14, there are only two sampled end states and there is no solution found by the vehicle, by increasing the number of lateral samples a solution can be easily found. In general condition of completeness can be improved in two ways, first is to sample as many points as possible and as closely as possible in the solution space. The second method is to keep on sampling until an end solution is found or timeout has been reached. The former method will reduce the computational performance while the later can be complex and expensive. The planner proposed in this thesis implements a combination of both methods, initially as many samples as possible are created and then evaluates these samples based a pre-cost of sample till a feasible solution is found. It is recommended to achieve completeness for safety purposes in autonomous driving.

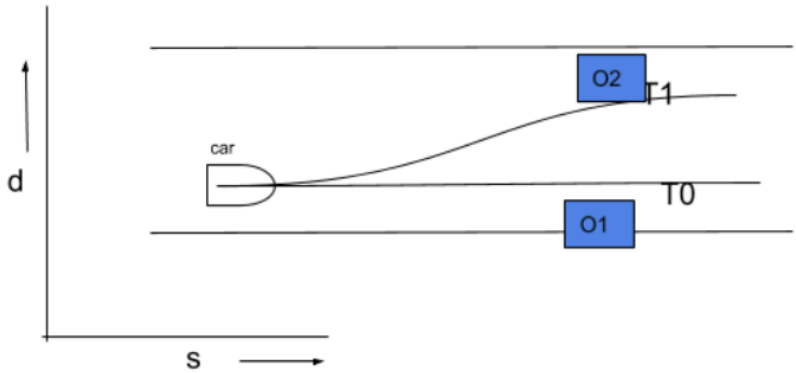


Figure 6.14: Probablistic Completeness

In summary, the proposed planner is probabilistically complete and current implementation with a limited number of acceleration profiles, and lateral shift samples are suitable for the modelcar, and higher sampling needs to be performed to improve completeness.

6.2.4 Runtime

Though computation power is available cheaply, it is important to create solutions which are cheaper and can be employed in large scale. In this case, sampling based approaches perform well and run on low computational hardware. In contrast lattice-based approaches such as [45] [58] [59] computationally expensive and require a GPU to run. Low computational costs mean larger chances to be adopted to a greater number of platforms.

To compute the complexity of planner lets consider n_a denote the number of acceleration/deceleration profiles, n_s denote the stopping deceleration profiles and n_l denote the number of lateral distance samples. Then the maximum number of samples created is $(n_a + n_s) * n_l$, generally stopping profiles are less as at higher deceleration lower number of lateral samples are chosen due to limitations in vehicle dynamics. This thesis employs a hybrid combination of two methods mentioned in 6.2.3 to achieve completeness. Therefore in the best case, only one trajectory is evaluated, and complexity is $O(1)$, and the worst case complexity is $O((n_a + n_s) * n_l)$.

Trajectory evaluation with respect to dynamic obstacles is an expensive process in the evaluation of trajectories. Simulation-based methods as discussed in [32] are expensive with complexity in terms of $O(n_o * n_n)$ where n_o is the number of obstacles and n_n is the number of simulation steps. This thesis employs a simple collision checking algorithm with constant time for evaluating one obstacle thus reducing the complexity to $O(n_o)$, number of obstacles. This process is not effective in intersections. Currently, a conservative approach to wait for other obstacles to pass is

used, a simple approach as discussed in [19] which has a performance better than the simulation-based algorithms can also be employed in future for intersections.

6.2.5 Deliberative Approach

The planner proposed in this thesis maintains a mix of deliberative and reactive approach. Deliberative by evaluating a trajectory completely before committing to it, this is important to create trajectories adhering to traffic, safe and comfortable. In general, all the planners evaluate all the sampled trajectories then choose the best based on different costs. The planner proposed in this thesis does not follow this convention, and once it finds the best trajectory, it stops evaluating the other trajectories as presented in subsection 4.5.6. It does not limit the real-time response of the trajectory as the sampling is chosen such that the worst case response time is within the hard real-time response required by the planner.

In summary, the planner achieves a reactive behaviour with higher accelerations and trajectory selection algorithm at the same time being deliberate in selecting the final trajectory for execution.

6.2.6 Final Review

The approach proposed in this thesis divides the problem of motion planning into subproblems across different layers minimising the computational effort at each level making the planner suitable for model cars. The trajectory creation and selection algorithm presented in this thesis exploits the cost functions and intended behaviour to converge to a solution without evaluating all the sample trajectories. The collision checking algorithm proposed in this thesis utilises the trajectory shape to detect collisions with static and dynamic obstacles in constant time over linear algorithms such as simulation-based approaches. The planning algorithm proposed follows a fixed time horizon resulting in a flexible spatial horizon enabling ego to plan independent of current speed. The planner plans such that the ego is collision free for next temporal horizon time (5 seconds).

The planner cannot work independently and needs inputs from a behavioural layer with scenario analysis algorithm to make optimal long-term decisions as discussed in previous subsections. The constant acceleration profiles used in the planner create high jerk trajectories; these are suitable for model cars but not for the actual vehicles. Functions related to path and velocity planning must be reimplemented with higher order polynomials for smoothness. The collision detection algorithm follows a conservative approach for the dynamic obstacles moving at intersections, and there is a significant scope for improvement to adapt to traffic at intersections or pedestrians moving across street.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The motion planning algorithm developed in this thesis is mainly aimed towards small car like robots operating in uncertain urban environments. The planner creates reasonable road parallel trajectories avoiding obstacles as required for on-road driving.

The major contributions of this thesis are as follows:

- Creating a planning framework suitable for small cars operating with low computational costs and inaccuracies in measurement.
- Heuristic based evaluation of samples (acceleration profiles and lateral shifts) to reduce the number of samples evaluated to find collision-free trajectory.
- Two step algorithm for detecting collision with static obstacles.
- Three step algorithm for detecting collision with dynamic obstacles.

The planner employs a combination of predefined acceleration profiles and lateral shifts to create samples; these sampled profiles are assigned pre-costs based on the target velocity and lateral shift. The sampled profiles create trajectories that vary from straight, constant velocity paths to one containing lateral shifts, acceleration and deceleration. The wide range of acceleration and deceleration profiles enable ego vehicle to react quickly in case of an emergency. The trajectories are evaluated based on the pre-costs, thus, only creating and assessing all trajectories in the worst case scenario. This heuristic algorithm saves computational effort while driving in obstacle free or sparse environments.

The created trajectories are evaluated to check if they are collision free and comfortable. As the planner is mainly aimed at the small car like robots, comfort is not a mandatory criterion in trajectory selection, and a major effort has spent in reducing computational effort in collision checking. The two-step collision checking algorithm proposed checks maximum at 4 points on the trajectory to decide whether it is collision free with respect to the static obstacle. It exploits the trajectory representation in Frenet frame and geometry of the path to achieve this. Similarly, a 3 step approach for collision checking in dynamic environments is developed, it detects if the ego vehicle is colliding with dynamic obstacles by checking time gap between the

ego and obstacle at the start and end of the common traversal area. Checking time gap allows the ego vehicle to maintain appropriate distance to obstacles independent of its velocity.

The proposed planner has been evaluated on a simulator with multiple scenarios similar to situations experienced in urban driving, i.e., with road blockades, pedestrians/vehicles moving laterally across the street, merging into traffic, overtaking a slow-moving obstacle etc. The planner successfully navigated the environment avoiding the traffic and stopping when no path could be found in the simulator. On the car platform without obstacles the planner performed reasonably because of the tracking errors from the sensing and control module.

7.2 Future Work

The developed framework is a step taken towards the development of trajectory planner for model car platform. This thesis presents various techniques to create trajectories and validate them at a lower computational cost. However, there is still a large scope for improvement and modifications to the present developed work. Some major improvements needed are as follows:

- The approximation technique used for path representation in Frenet frame and conversion between Frenet and Cartesian frame is not perfect. This could be improved by considering the curvature of the road and using spline approximation instead of lines.
- Improve trajectory creation and representation by using higher order polynomials to obtain continuity in path and curvature.
- Improve the collision checking for dynamic obstacles, mainly for objects moving laterally. The current proposal slows down or stops the ego vehicle when a laterally moving obstacle is found, it can be improved to estimate the future position of the dynamic obstacle for collision checking.
- Improve the evaluation of planner by testing in different road networks, traffic scenarios, on the car and determine failing scenarios. Improve the simulator to create continuous dynamic and random traffic to test limits of the planner.
- Improve the heuristics employed in the planner for calculating pre-costs of samples and also the cost functions in trajectory selection.
- Optimize code and increase number of samples in trajectory creation.

Bibliography

[1] AutoModelCar. Hardware (autonomos model v3). 14

[2] R. Behringer and N. Muller. Autonomous road vehicle guidance from autobahnen to narrow curves. *IEEE Transactions on Robotics and Automation*, 14(5):810–815, Oct 1998. 5

[3] R. Benenson, S. Petti, T. Fraichard, and M. Parent. Integrating perception and planning for autonomous navigation of urban vehicles. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 98–104, Oct 2006. 10

[4] P.S. Bokare and A.K. Maurya. Acceleration-deceleration behaviour of various vehicle types. *Transportation Research Procedia*, 25:4733 – 4749, 2017. World Conference on Transport Research - WCTR 2016 Shanghai. 10-15 July 2016. 24

[5] Z. Boroujeni, D. Goehring, F. Ulbrich, D. Neumann, and R. Rojas. Flexible unit a-star trajectory planning for autonomous vehicles on structured road maps. In *2017 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 7–12, June 2017. 7, 11, 29

[6] Alberto Broggi, Paolo Medici, Paolo Zani, Alessandro Coati, and Matteo Panciroli. Autonomous vehicles control in the vislab intercontinental autonomous challenge. *Annual Reviews in Control*, 36:161–171, 2012. 9

[7] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Springer Publishing Company, Incorporated, 1st edition, 2009. 5, 7, 9

[8] Yu Sang Chang, Yong Joo Lee, and Sung Sup Brian Choi. Is there more traffic congestion in larger cities?-scaling analysis of the 101 largest us urban centers. *Transport Policy*, 59:54–63, 2017. 1

[9] R. G. Cofield and R. Gupta. Reactive trajectory planning and tracking for pedestrian-aware autonomous driving in urban environments. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 747–754, June 2016. 12, 22, 24

[10] Gerald Cook. *Robot Navigation*, pages 324–. Wiley-IEEE Press, 2011. 17

[11] P. Czerwionka, M. Wang, and F. Wiesel. Optimized route network graph as map reference for autonomous cars operating on german autobahn. In *The 5th International Conference on Automation, Robotics and Applications*, pages 78–83, Dec 2011. 18

[12] DARPA. Route network definition file (rndf) and mission data file (mdf) formats. 18, 19

[13] M. Du, J. Chen, P. Zhao, H. Liang, Y. Xin, and T. Mei. An improved rrt-based motion planner for autonomous vehicle in cluttered environments. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4674–4679, May 2014. 8

[14] Dave Ferguson and Maxim Likhachev. Efficiently using cost maps for planning complex maneuvers. *Lab Papers (GRASP)*, page 20, 2008. 8

[15] Dave Ferguson and Anthony Stentz. Field d*: An interpolation-based path planner and replanner. In Sebastian Thrun, Rodney Brooks, and Hugh Durrant-Whyte, editors, *Robotics Research*, pages 239–253, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. 6

[16] Gaston A. Fiore and David L. Darmofal. A robust motion planning approach for autonomous driving in urban areas. 2008. 7

[17] Dahlem Center for Machine Learning and Robotics. Autonomos model car. 13

[18] Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. Informed rrt*: Optimal incremental path planning focused through an admissible ellipsoidal heuristic. *CoRR*, abs/1404.2334, 2014. 7

[19] L. Garrote, C. Premebida, M. Silva, and U. Nunes. An rrt-based navigation approach for mobile robots and automated vehicles. In *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, pages 326–331, July 2014. 7, 12, 53

[20] Daniel Gohring. Controller architecture for the autonomous cars: Madeingermany and e-instein. 35

[21] Daniel Graff. *Programming and Managing Swarms of Mobile Robots:A Systemic Approach*. PhD thesis, Technischen Universitat Berlin, 2016. 30

[22] Poul Greibe. Braking distance, friction and behavior. 24

[23] Tianyu Gu and John M. Dolan. On-road motion planning for autonomous vehicles. In *Proceedings of the 5th International Conference on Intelligent Robotics and Applications (ICIRA 2012)*, pages 588–597, October 2012. 7

[24] Tianyu Gu, John M. Dolan, and Jin-Woo Lee. On-road trajectory planning for general autonomous driving with enhanced tunability. In K. Berns H. Yamaguchi) Springer Verlag ISBN 978-3-319-08337-7 (eds.: E. Menegatti, N. Michael, editor, *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*, July 2014. 9

[25] THOMAS M. HOWARD. *Adaptive model predictive motion planning for navigation in complex environments*. PhD thesis, Carnegie Mellon University, 2009. 10

[26] L. Jaillet, J. Hoffman, J. van den Berg, P. Abbeel, J. M. Porta, and K. Goldberg. Eg-rrt: Environment-guided random trees for kinodynamic motion planning with uncertainty and obstacles. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2646–2652, Sept 2011. 6

[27] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller. Anytime motion planning using the rrt*. In *2011 IEEE International Conference on Robotics and Automation*, pages 1478–1483, May 2011. 7

[28] Christos Katrakazas, Mohammed Quddus, Wen-Hua Chen, and Lipika Deka. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. *Transportation Research Part C: Emerging Technologies*, 60:416 – 442, 2015. 6, 10, 12

[29] Lydia E. Kavraki and Jean-Claude Latombe. Probabilistic roadmaps for robot path planning, 1998. 6

[30] Joseph Kearney, Hongling Wang, and Kendall Atkinson. Robust and efficient computation of the closest point on a spline curve. In *In in Proc. 5th International Conference on Curves and Surfaces*, pages 397–406, 2002. 23

[31] J. Kim, K. Jo, W. Lim, M. Lee, and M. Sunwoo. Curvilinear-coordinate-based object and situation assessment for highly automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 16(3):1559–1575, June 2015. 22

[32] Sascha Kolski. *Autonomous Driving in Dynamic Environments*. PhD thesis, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 2008. 6, 7, 9, 11, 12, 52

[33] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 1398–1404 vol.2, Apr 1991. 6

[34] J. J. Kuffner and S. M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 2, pages 995–1001 vol.2, 2000. 6

[35] 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*, pages 1681–1686, Sept 2008. 7

[36] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, 17(5):1105–1118, Sept 2009. 7

[37] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA, 1991. 5

[38] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998. 6

[39] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, New York, NY, USA, 2006. 5

[40] X. Li, Z. Sun, D. Cao, Z. He, and Q. Zhu. Real-time trajectory planning for autonomous urban driving: Framework, algorithms, and verifications. *IEEE/ASME Transactions on Mechatronics*, 21(2):740–753, April 2016. 7, 9, 22, 26, 27

[41] Maxim Likhachev, Dave Ferguson, Geoff Gordon, Anthony Stentz, and Sebastian Thrun. Anytime dynamic a*: An anytime, replanning algorithm. In *Proceedings of the Fifteenth International Conference on International Conference on Automated Planning and Scheduling*, ICAPS'05, pages 262–271. AAAI Press, 2005. 6

[42] KRISTIJAN MACEK. *Autonomous vehicle navigation in dynamic urban environments for increased traffic safety*. PhD thesis, ETH ZURICH, 201o. 9

[43] D. Madås, M. Nosratinia, M. Keshavarz, P. Sundström, R. Philippsen, A. Eidehall, and K. M. Dahlén. On path planning methods for automotive collision avoidance. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 931–937, June 2013. 51

[44] Kristijan Maček. *Autonomous vehicle navigation in dynamic urban environments for increased traffic safety*. PhD thesis, ETH Zurich, 2010. 21

[45] Matthew McNaughton. *Parallel Algorithms for Real-time Motion Planning*. PhD thesis, Carnegie Mellon University, 2011. 7, 8, 9, 11, 12, 51, 52

[46] Arpan Mehar, Satish Chandra, and Senathipathi Velmurugan. Speed and acceleration characteristics of different types of vehicles on multi-lane highways. In *European Transport, 2013 - istiee.org*, 2013. 24

[47] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and

S. Thrun. Junior: The stanford entry in the urban challenge. *Journal of Field Robotics*, 2008. 9

[48] Michael W. Otte and Emilio Frazzoli. Rrtx: Real-time motion planning/replanning for environments with unpredictable obstacles. In *WAFR*, 2014. 7

[49] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, March 2016. 6

[50] Y. Rasekhipour, A. Khajepour, S. K. Chen, and B. Litkouhi. A potential field-based model predictive path-planning controller for autonomous road vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1255–1267, May 2017. 6

[51] James Reeds and Lawrence Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific journal of mathematics*, 145(2):367–393, 1990. 10

[52] News Report. Singapore: no more cars allowed on the road, 2018. 1

[53] WHO Report. Road traffic injuries, January 2018. 1

[54] M. Rufli and R. Siegwart. On the design of deformable input- / state-lattice graphs. In *2010 IEEE International Conference on Robotics and Automation*, pages 3071–3077, May 2010. 8

[55] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *CoRR*, abs/1708.06374, 2017. 30

[56] Ravi Shanker, Adam Jonas, Scott Devitt, Katy Huberty, Simon Flannery, William Greene, Benjamin Swinburne, Gregory Locraft, Adam Wood, Keith Weiss, et al. Autonomous cars: self-driving the new auto industry paradigm. *Morgan Stanley Blue Paper, Morgan Stanley & Co. LLC*, 2013. 1

[57] Hongling Wang, Joseph Kearney, and Kendall Atkinson. Arc-length parameterized spline curves for real-time simulation. In *In in Proc. 5th International Conference on Curves and Surfaces*, pages 387–396, 2002. 23

[58] Shuiying. Wang. *State Lattice-based Motion Planning for Autonomous On-Road Driving*. PhD thesis, Freie Universität Berlin, 2015. 7, 9, 22, 23, 52

[59] M. Werling, J. Ziegler, S. Kammel, and S. Thrun. Optimal trajectory generation for dynamic street scenarios in a frenet frame. In *2010 IEEE International Conference on Robotics and Automation*, pages 987–993, May 2010. 10, 26, 52

[60] Marios Xanthidis, Ioannis M. Rekleitis, and Jason M. O'Kane. RRT+ : Fast planning for high-dimensional configuration spaces. *CoRR*, abs/1612.07333, 2016. 6

[61] Xi Xiong, Jianqiang Wang, Fang Zhang, and Keqiang Li. Combining deep reinforcement learning and safety based control for autonomous driving. *CoRR*, abs/1612.00147, 2016. 6

[62] Wenda Xu, Junqing Wei, J. M. Dolan, Huijing Zhao, and Hongbin Zha. A real-time motion planner with trajectory optimization for autonomous vehicles. In *2012 IEEE International Conference on Robotics and Automation*, pages 2061–2067, May 2012. 7, 9, 11, 22

[63] Julius Ziegler and Christoph Stiller. Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IROS'09, pages 1879–1884, Piscataway, NJ, USA, 2009. IEEE Press. 22

Appendix A

Appendix

A.1 Static Obstacle Collision checking

Static obstacle checking described in section 4.5.4 works on the concept that it is sufficient to check the lateral distance (D) between the obstacle and the ego at minimum and maximum of ego D (lateral coordinate) in the longitudinal intersection region.

Let $f(x)$ be a function of lateral motion D in intersection interval $[a, b]$, $f(x) \geq f(c)$ (respectively, $f(x) \leq f(c)$), then $f(c)$ is the absolute minimum (respectively, absolute) local maximum value of $f(x)$ on $[a, b]$ and vice versa.

If $f(x)$ is continuous on $[a, b]$ and differentiable in (a, b), a point c in $[a, b]$ is a critical point of $f(x)$ if either $f'(c) = 0$ or $f'(c)$ does not exist. The

If $f(x)$ is continuous on $[a, b]$ and differentiable in (a,b), and if for some c in (a, b), if $f(c)$ is local minima or maxima then c must be a critical point. Any absolute maximum or minimum will take place at boundaries or critical points inside the interval (a, b).

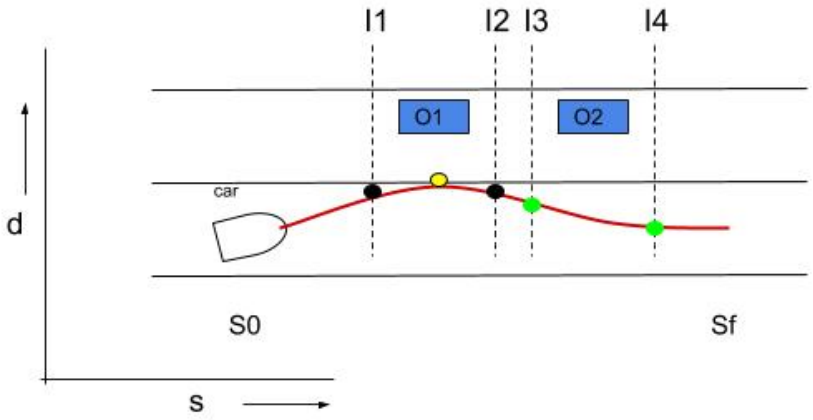


Figure A.1: Static Obstacle Collision check - Lateral distance between obstacle O1 and trajectory is checked at points with black dots, distance at green dots is checked for obstacle O2.

Consider $f(x) = x^3 - x^2 - x + 12$ with intersection interval in $[0, 2]$, critical points

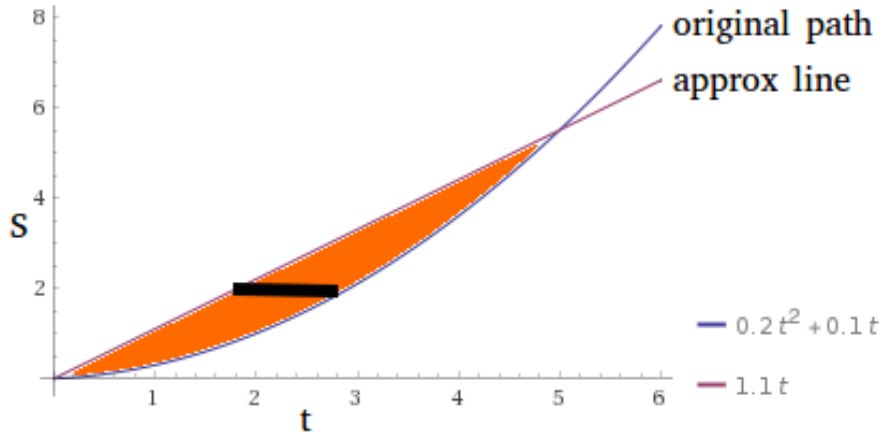


Figure A.2: The parabolic curve indicates change of position with respect to time and the approx line indicates a linear change. The orange shared region indicates the maximum time difference between a linear approximated and the original curve.

are at $x = -1/3$ which is not in intersection and $x = 1$ is in the intersection interval. Thus it is sufficient to check for values of $f(x)$ at [0,1,2] to find minima and maxima.

In the figure A.1, for obstacle O1 it is necessary to check collision at a critical point that lies in [I1, I2] represented with a yellow dot and at the borders represented with black dots. For the second obstacle O2, it is sufficient to check the boundaries of the interval [I3, I4]. If the critical points and the borders have an adequate lateral distance to the obstacle, then the rest of the points on trajectory will also have safe lateral distance, and it is not needed to check for collision at other points.

A.2 Dynamic Obstacle Collision checking

In the proposed algorithm in section 4.5.5, for collision checking with respect to dynamic obstacles time difference only at the borders where lateral and longitudinal coordinates intersect is checked. This assumption is valid if the maximum non-linearity introduced by the ego vehicle acceleration is not more than 2s. The next set of equations define why the hypothesis is correct.

Figure A.2 shows where there is a difference between the linear approximation and the parabolic path. A black line shows the maximum time difference in the orange region.

Equation of the parabolic path.

$$S_{curve} = ut + 0.5at^2 \tag{A.1}$$

Values of the function A.1 at start 0 and time horizon $t_h = 5s$

$$S_0 = 0; S_f = 5u + 12.5a \tag{A.2}$$

Corresponding approximated line equation $y = mx + c$, as both start at same point consider at origin.

$$S_{line} = \frac{5u + 12.5a}{5}(t) \tag{A.3}$$

Converting the A.1 as a function of distance, we get

$$t_{orig} = \frac{-u \pm \sqrt{u^2 + 2as}}{a} \tag{A.4}$$

Converting the A.5 as function of distance, we get

$$t_{approx} = \frac{s}{u + 2.5a} \tag{A.5}$$

Time difference between original and approximated values is

$$t_{diff} = \frac{-u \pm \sqrt{u^2 + 2as}}{a} - \frac{s}{u + 2.5a} \tag{A.6}$$

The maximum value of t_{diff} between start and end of the path gives us the value required, as the equation is in 3 unknown variables, it is not possible to solve it directly to find the maximum value. Therefore exhaustive values are plotted to find the maximum value. The Figure A.3 shows the values of time difference for different values of acceleration from $-0.7ms^{-2}$ to $-0.3ms^{-2}$, initial velocity of $0ms^{-1}$ to $1ms^{-s}$ is used to plot in the region of $[0, u + 12.5a]$ as s dimension.

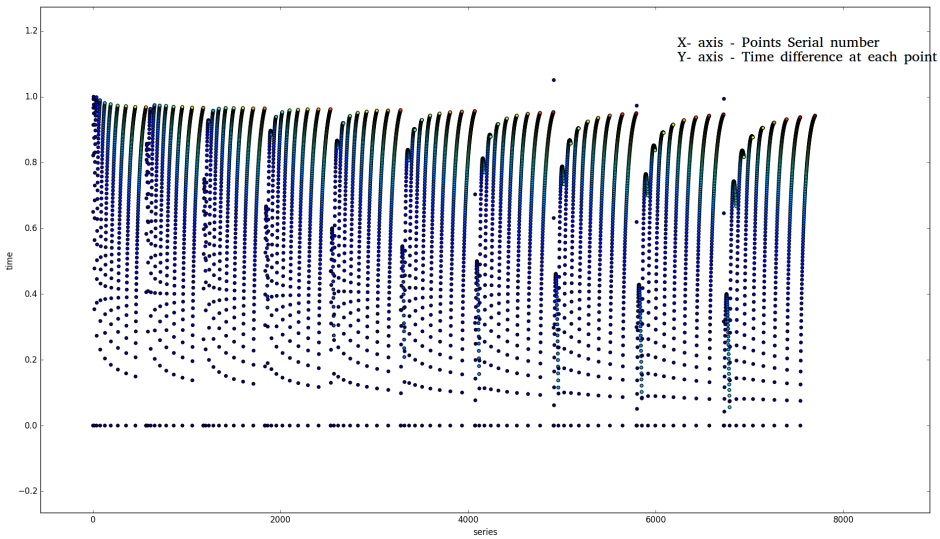


Figure A.3: Maximum time difference for different samples of different parameters of the equation A.6

The maximum difference at any point is 1s, thus the approximation holds good. The following python code is used to plot the results and find the maximum value.

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np
from math import sqrt

def frange(start,stop, step=1.0):
  while start < stop:
    yield start
    start +=step
t = [] #time difference
r = [] #series number
co =[] #color differentiator
val =0 #series counter
for x in range(10):#u
  for y in range(10):#a
    k = 5*x+12.5*y #max distance range in multiple of 10
    if k<=0:
      k=0
    for z in  frange(0, k, 1):#s
      try:
        u=x/10.0
        a = -0.7+(y/1.0)
        s = z/10.0
        t_diff = ((-u+sqrt(u*u + 2*a*s))/a)-(s/(u+2*a))
        t.append(t_diff)
        r.append(val)
        co.append(z)
        val+=1
      except Exception as e:
        pass
plt.scatter(r,t,c=co)
plt.ylabel('time')
plt.xlabel('series')
plt.show()
```