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

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. Prior approaches solved this problem through different methods having various complexity levels. This thesis proposes an on road motion planner with combination of path-velocity combination along with a low computational overhead collision avoidance system. The reactive nature tunability of the proposed thesis is suitable for urban driving scenarios.

Abstrakt

To be translated

Acknowledgements

Add acknowledgments to FU, professor, supervisor, lab TU - supervisor

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

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 great 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 huge societal impacts by reducing the fatalities in car accidents to changing the ecosystem of transportation. It is reported by World Health Organization that every year there are more than 1.25 million deaths and between 20-50 million people suffering non-fatal injuries worldwide because of road accidents [49]. 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 [7]. Increase in congestion is a result of increase in car ownership, to combat congestion cities like Singapore are banning new car ownership [48]. 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 [52] 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 needed is mainly in the areas of urban driving involving 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 there is a need for research into understanding these interactions.

Full-scale autonomous vehicles are expensive, time taking, and safety critical to test in emergency scenarios, though simulators provide easy ways to evaluate the

situations, physical aspect of the vehicle control is missing. Research into autonomous vehicles is an expensive endeavour with high costs on platforms, operational constraints and involves large teams. Model cars (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.

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 navigation/trajectory planning challenge is special as the module is responsible for understanding the behaviour of the surrounding objects and adjust the ego vehicle's behaviour accordingly following certain rules.

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 move on a planned route avoiding obstacles. There exist well-known methods within the domain of the robot/autonomous vehicle path planning that can be adapted to achieve trajectory planning for model car. Model cars are limited to the available resources such as computational power, perception data, accuracy in measurements etc. Thus, these approaches need to be tailor-made to suit the model car platforms.

Trajectory planning problem 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 minimize a performance index. The performance index could be one or multiple parameters such as minimizing the distance to 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 can be performed 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 smaller model car 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 simple 2 steps for detecting collision with static obstacles and 3 steps for dynamic obstacles.

1.5 Thesis Structure

The thesis is broadly organized 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. The approach is followed by implementation overview, evaluation and conclusion.

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 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 [33], [35] 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 [1], the research was accelerated by competitions such as DARPA Grand Challenge in 2006 and DARPA Urban Challenge(DUC) in 2007 [6]. 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 [6]. All participants in the competetion followed a similar approach 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 3 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 aim of building trajectories that are collision free and also adhere to kinematic and dynamic constraints of vehicle, road boundaries and traffic rules [25]. 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 [30] [46] [57]. In this approach 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. Grid-based approach is another generally used method 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 [37] [14] [29]. A comprehensive study of various approaches in motion planning for autonomous vehicles is presented in [25] [45].

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.

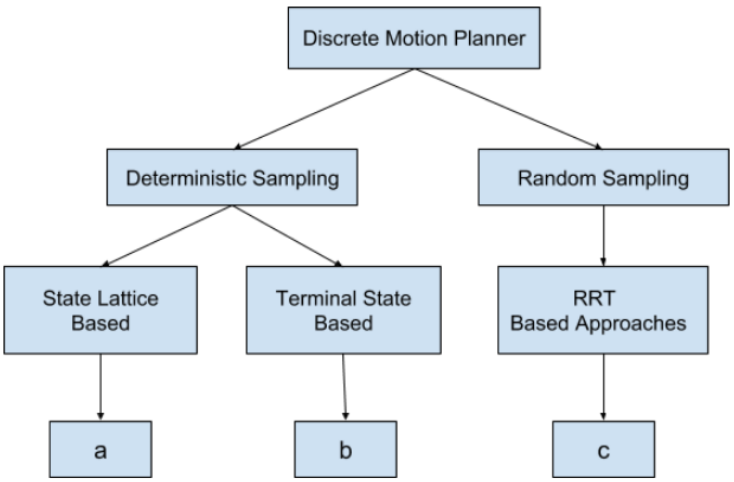


Figure 2.1: General Overview of On-road planners. a - [41] [54] [58] [20] [4] , b - [29] [36] [6], c -[17] [32] [15]

nge ter-
al state to
l search

2.1.1 Random Sampling Approaches

Rapidly-exploring Random Tree(RRT) technique was initially introduced by Steven M. LaValle in his work presented in [34]. 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[26](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 1 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[56]. To improve the performance of RRT, bidirectional RRT with two trees(Bi-RRT) has been proposed, though it improves the performance, handling discontinuities of two trees is difficult[31]. Environmentally guided RRT(EG-RRT) is another variant that incorporates information from scene analysis to guide samples and converge faster[23]. MIT has applied a closed loop prediction model into RRT(CL-RRT) in its autonomous vehicle at DUC [15], 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* [17] has been proposed which guarantee asymptotic optimality. In this approach, each state stores cost from 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 [16] 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 completely predictable and the computed path needs to be dynamically re-planned over the time. This needs either a new plan to be generated quickly or rapid correction of old path. The approach Anytime-RRT [24] creates an initial path using RRT and optimizes it on the go to create an optimal path, re-planning is triggered when the path is no longer valid. The method RRT^x [44] 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 in terms of motion planning for autonomous vehicles, especially in on-road driving conditions. RRT-based planners generate jerky and unnatural trajectories that contain many unnecessary turns[12], these are suitable for open areas such as parking areas. Road parallel trajectories are preferred in structured environments.

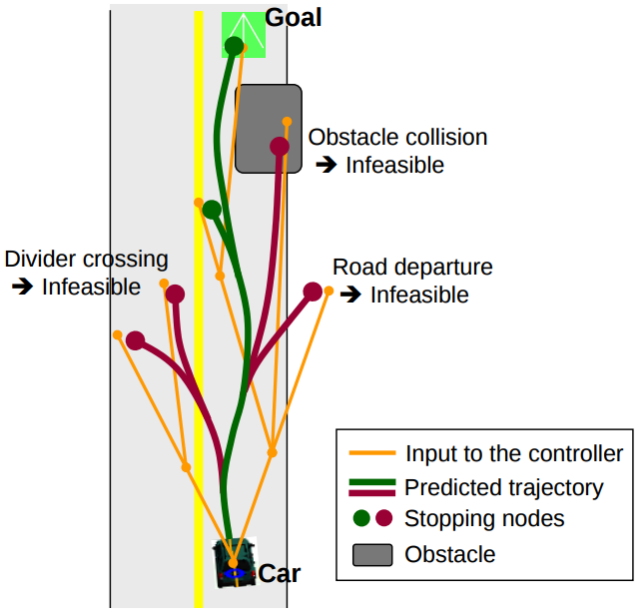
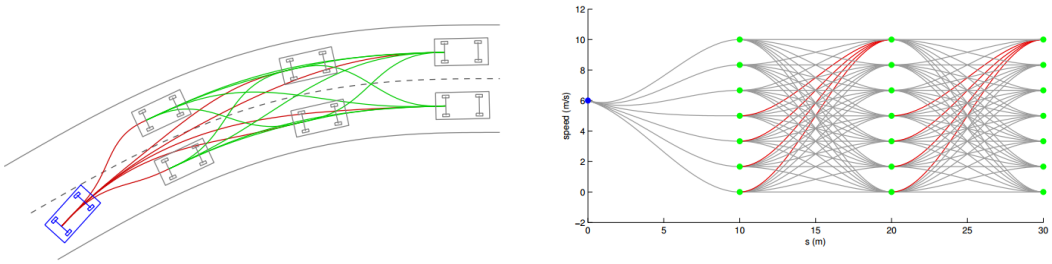


Figure 2.2: MIT Darpa Urban Challenge - RRT based planning approach

2.1.2 Lattice Planners

The lattice planners use a discrete representation of the planning area with multiple states, each state is multi dimensional with dimensions such as position, acceleration, velocity, time, curvature, heading etc. These states are connected together and the problem then reduces to finding a path from the initial state to the final state in the lattice. This approach is generally well suited for non-holonomic robots and highly constrained areas such as road networks[13]. 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 [13] uses a multi-resolution state lattice with high resolution near start and goal locations and lower resolution in middle to reduce computational complexity. Continuity of path and curvature are constraints in path planning, these are addressed in the research [50] by defining a 4D configuration including 2D position, heading and steering. In [41] 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 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 thousands and increase exponentially if the 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 [41], the research published in [58] utilizes quartic



(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, the red curves are beyond the acceleration limit, and the grey curves are valid ones

Figure 2.3: Lattice Planner - Path velocity representation in [58]

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 [21] 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 [54] 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 [6] [43] [29] [5] [36] [38] perform a variation of local search. The generated candidate trajectories are evaluated with a set of cost functions for collision checking and comfort to finalize the final trajectory. Paths with lateral shifts can generally 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[22]. Partial motion planning is another technique used to search locally

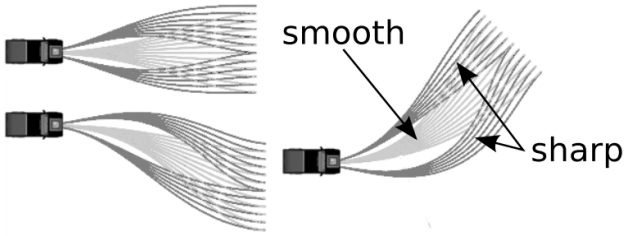


Figure 2.4: CMU Darpa Urban Challenge - Local Search Forward projected trajectories

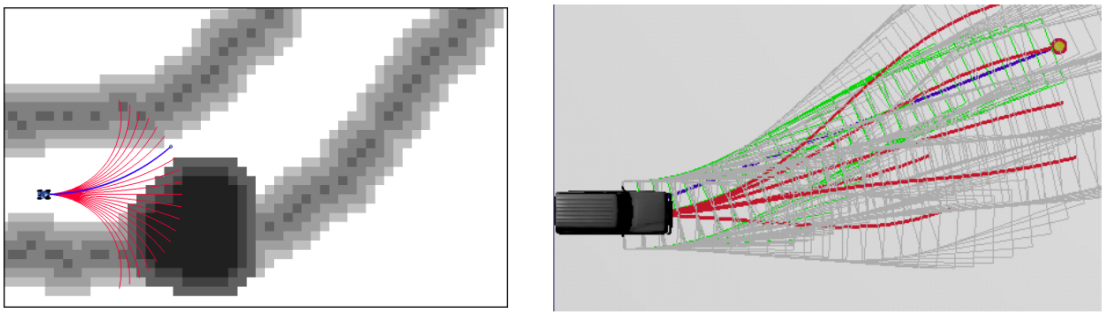
reducing computational power significantly [2]. In summary, local search algorithms can create short-term trajectories at a low computational cost with limitations in the manoeuvres robot can perform, these are especially suitable for low-speed and on-road driving.

2.2 Trajectory Representation

Trajectory or path representation is another important 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 [25]. Splines and polynomials are used for creating road parallel trajectories, fifth order polynomials produce jerk minimizing trajectories[55] and other formats are also employed by different planners. Reeds-shepp curves, a version of Dubian curves allow forward and backward driving [47] thus making them suitable for complex manoeuvres in parking lots, obstacle course etc.

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 [58] 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 optimizing factor, [4] penalize breaking to prefer long-range trajectories, [41] penalizes shorter horizons to ensure minimum horizon. Major criteria for trajectory



(a) Static Environments - Cost of path is computed based on cost of the cells it traverses through, darker the cell higher the cost.

(b) Dynamic Environments, Static Obstacles- For series of trajectories vehicle shape is forwarded and collision checking is performed

Figure 2.5: Collision Checking [29]

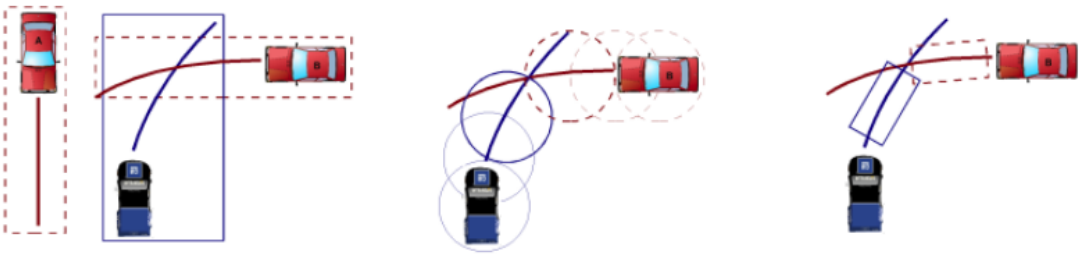


Figure 2.6: Collison checking for Dynamic Obstacles [29]

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 [29], for collision checking in static environments, 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 shwon in Figure 2.5a. Finally, trajectory with 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, [29] 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 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 [41], 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 close 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 [17] collision checking is performed by forward predicting 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 [8] 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 [25]. 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

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 3 modules mentioned are controlled with an Arduino Nano which communicates the data 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 for localizing car based on markers on the roof. The Figure 3.3 presents connections across different modules present in the car.

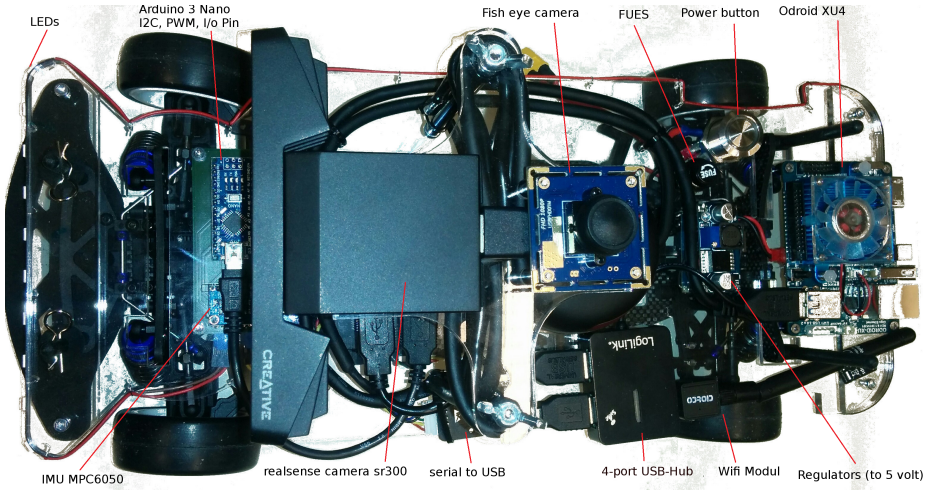


Figure 3.2: Internal components of Model Car

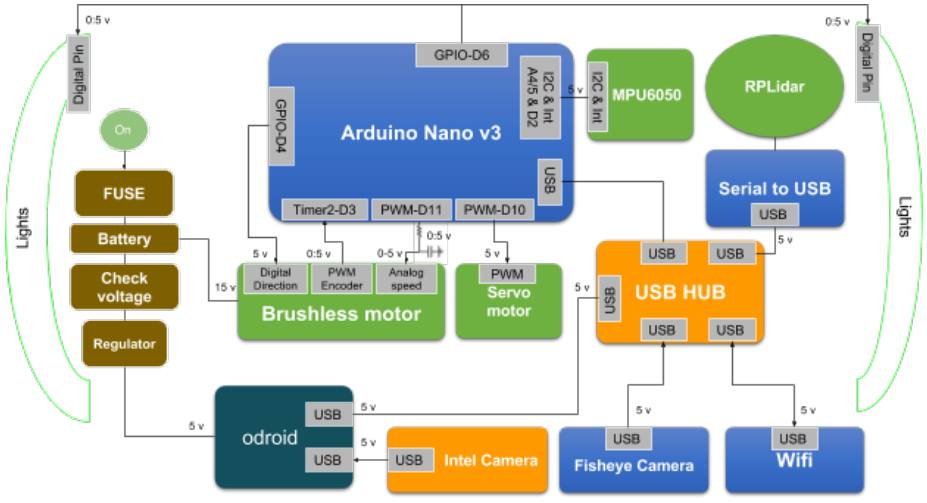


Figure 3.3: Module Connections

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.

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 roof and localize car, line detection, traffic sign detection etc.

In summary, 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

Mention tha ego vehicle, model car, r bot are used interchange- ably in the document

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. The aim of this chapter is to describe path planning techniques developed in this thesis to drive the ego vehicle safely to 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 localization 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 the understanding of surroundings in terms of 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 about 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 main focus of this thesis, components in light blue are adopted for this thesis and other modules in sky blue are preexisting or simulated.

4.2 Localization

Localization module is responsible for providing the current state of the vehicle in terms of position, orientation, speed(linear and angular) and acceleration. The localization module implemented on the modelcar has two sub components, Vehicle Odometry and Global Position estimation using Visual GPS. Odometry is calculated

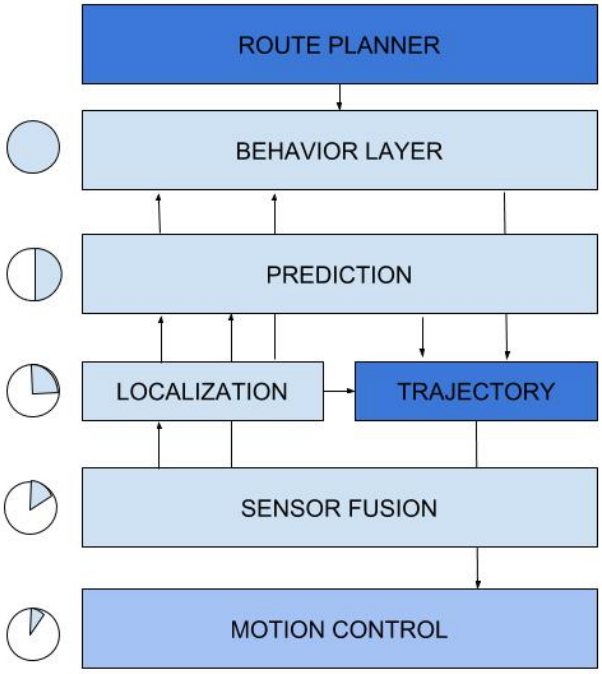


Figure 4.1: Path Planning Module

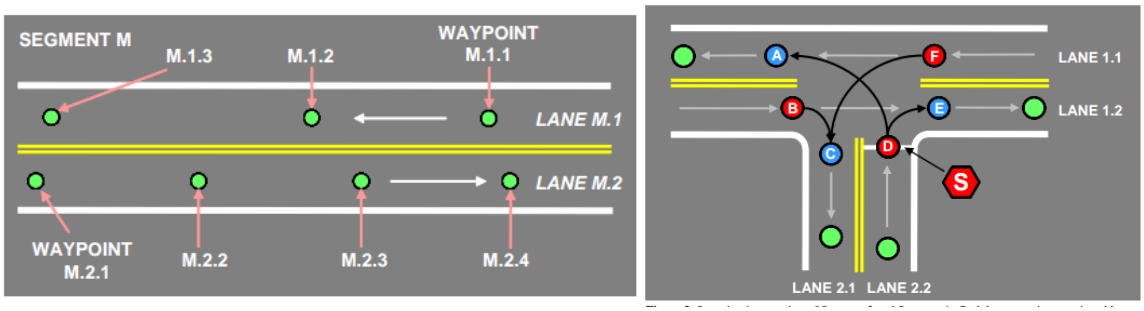
with dead reckoning [9] with speed information from motor and Yaw information from Inertial Measurement Unit (IMU). The localization module combines odometry with the information received from a visual GPS node (tracks markers on roof) to correctly estimate the state of the ego vehicle.

4.3 Prediction

Prediction and Sensor fusion modules receive the data from various sensors such as Cameras, LIDAR, and 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 vehicle current 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)" [11] [10] is



(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: Path velocity representation in [58]

used to represent the route network. Next subsections details further about RNDF and how global reference route is calculated.

4.4.1 RNDF

This section details about the RNDF file [11] which defines the road network(set of roads/ areas connected together) over which the vehicle can traverse. This representation of road is developed by DARPA 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 way points along the driving direction. More significant information about way points 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 way points at start/end to another. Figures 4.2a 4.2b 4.3 [11] represent different portions of the route representation and Figure 4.4 details regarding one of the route network of map used in Lab experiments.

foot note - source of images

4.4.2 Path Representation and calculation

Add Appendix for how to create the RNDF file for model car

The data in RNDF is represented in the form of a tree with connections across way-points. The global path from source to destination is the shortest path between the way-point closest to the ego vehicle's current position and the way-point closest to the destination. The shortest path found is sub divided into sub-paths based on the segment to which the way points belong.Once the ego vehicle is at the end of one sub-path it receives a notification from the trajectory planner that a goal has been

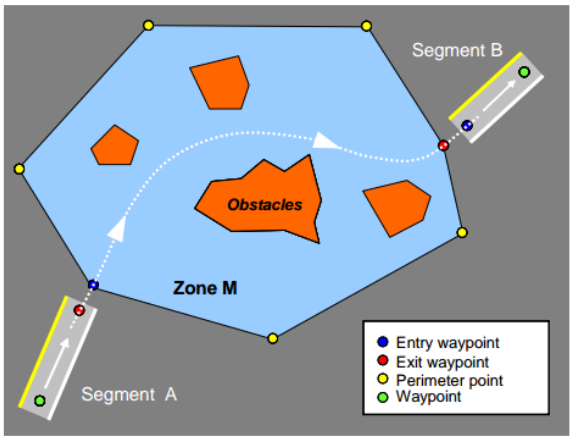


Figure 4.3: Zone Representation in RNDF - Connection between Segments and Zone

reached, then the route planner transmits the next sub path 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 modeling the road in trajectory planning stage.

4.4.3 Behavioral Layer

ck if this uld be ved to re- d work

Behavioral layer plays an important role in path planning, it is responsible for understanding the scenario and making decisions according to various traffic rules, constraints and choices that will make driving more efficient for the vehicle. For example, it decides on lane and speed to drive based on constraints such as emptiness of lane, other vehicles in lane, next needed exit and turn, speed limit etc. Behavioral layer is a huge 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 behavior using mouse clicks for lane change and speed information from map.

4.5 Motion Planner

This section discusses the planning algorithm to create short-term trajectories in accordance with the global path to reach destination. 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 modeling 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. The core of this section is 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

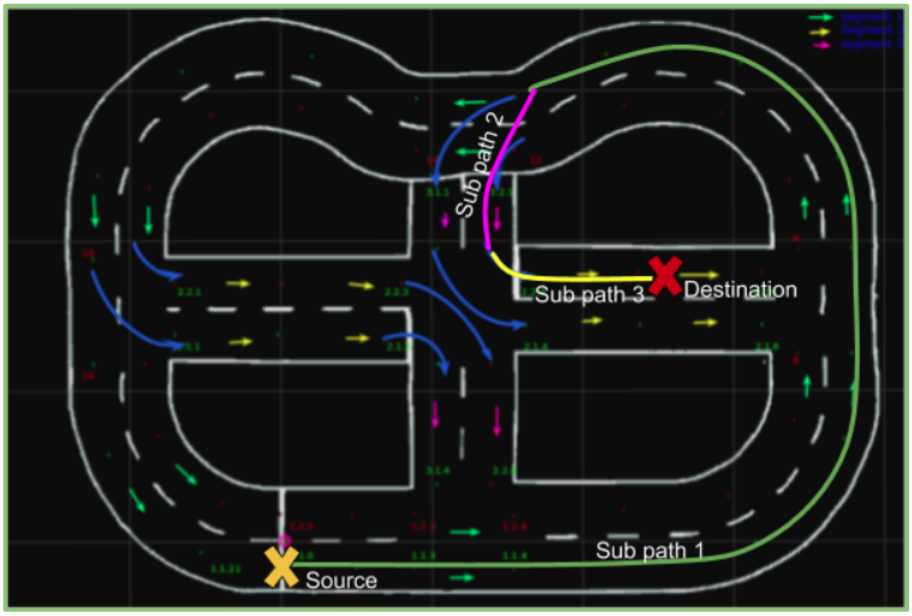


Figure 4.4: Division of shortest path in road network into Sub paths across different segments

obstacles. Next subsection 4.5.6 details on how a final trajectory is selected from the set of evaluated trajectories for the trajectory follower to follow.

4.5.1 Temporal Horizon

Time is an important aspect for planning in dynamic environments and there are several timing variables associated with planning. This section is mainly adopted from the doctoral thesis [40]. 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 measure of how far into the future trajectory of the ego vehicle is planned. Second is the prediction horizon T_p, it is 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 previous cycle is executed in 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 behavior of these vehicles is not controlled by the ego

check if this
name is nee
and add foo
note to link
"Autonomo
vehicle nav-
igation in d
namic urba
environment
for increase
traffic safety
of Macek
kristijan do
toral work

vehicle. Thus the constraint $T_s \leq T_p$ should be valid. This creates an uncertainty in modeling of the environment, thus the execution duration of current plan T_e beyond T_s is not sensible. This is due to fact that 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 with respect to goal objective, uncertainty also increases with time. In general a farsighted uncertain motion plan potentially directing the vehicle towards goal is better, but this plan needs to be re-evaluated and re-executed in short intervals for correctness.

ck if this uld be ved to im- mentation

In general behavior 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 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 even higher update frequency can be chosen as the planner is fast enough to react.

e to see imum dis- ce ahead h current ed, min- m time to ng car to t at good ed etc etc.

4.5.2 Path Modelling

Planned global path is in Cartesian coordinate system. One of the problems with the Cartesian coordinate system is that the due to variation in curvature of 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, [58] [59] [54] [36] [8] [28] are some of the research works in which Frenet frame is adopted. In this method center of the lane/road or preplanned global path is used as reference longitudinal coordinate (S) and perpendicular distance with respect to the lane center is considered as lateral coordinate (L/D) as represented in Figure 4.5 [54]. Thus once converted, (S,L) coordinate system essentially is a straight road.

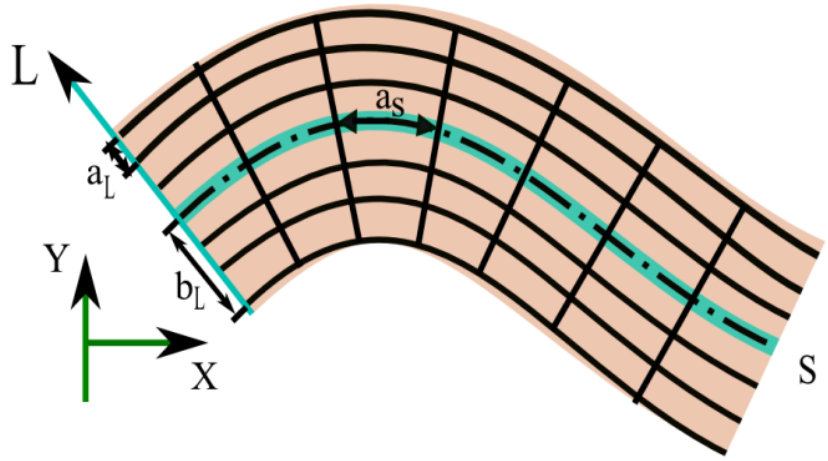


Figure 4.5: SL coordinate system laid over XY coordinate system - [54]

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 [8]. This method is computationally inexpensive and provides required level of accuracy for modelcar. To convert an xy coordinate to SL coordinate, (x,y) is projected onto the current path represented by straight lines joining way points 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 way points, a point at a perpendicular distance L gives the x,y coordinate. We assume that the path between two way points is linear which reduces the computational complexity in approximation. This approximation how ever approaches zero error when the spacing between two way points approaches zero. Adding dense way points in the curves significantly reduces the approximation error. There are different methods discussed in [53], [27] 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 road and gets compressed in concave side of the reference line. This is especially an issue in curves with lower radius of curvature, as it affects velocity planning thus leading to discomfort in some cases. There is a wide research in topic of velocity and path smoothening which counter these affects.

In summary, curvilinear coordinate system makes planning easier but needs extra computation in 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 needs to be addressed.

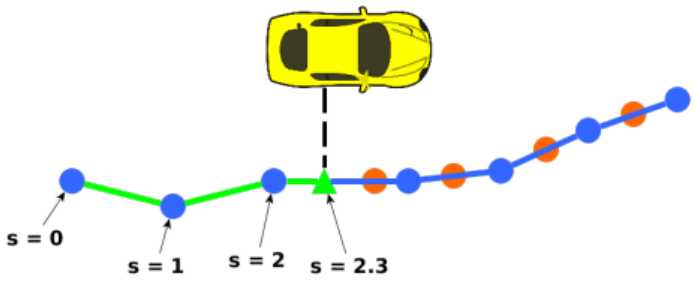


Figure 4.6: Showing projection of the current position of the car shown by green triangle to s coordinate system. Each circle represents a node, and the lines between them are links. [8]

VOLVO
rence ref-
ıce in foot
e and also
ove the or-
e dots

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 behavior 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, are smooth, continuous and comfortable. Chapter 2 discusses various planning techniques used by different planners.

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 maneuver. 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 measurement errors in the ego, once state approximation is improved the planner can be switched to trapezoidal acceleration profiles as shown in sub-figure4.7. Generally, A1 represents an acceleration of around $2m^{-2}$ and A8 of up to $-8m^{-2}$ 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 $-4.5m^{-2}$ [19] [42] [3]. 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

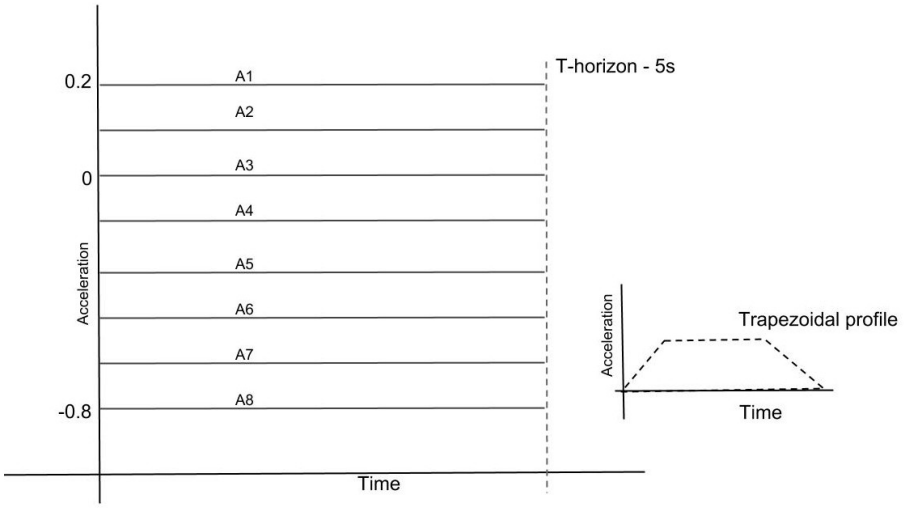


Figure 4.7: Different acceleration profiles a car can follow from current state.

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 depends also on the capabilities of the ego vehicle and controller to track these fine trajectories.

Acceleration profiles discussed above solve the problem of longitudinal planning but to avoid obstacles the ego vehicle should also plan lateral (sideways) shifts in its 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. [55] suggests that at lower speeds it is advantageous to map lateral shift as a function of distance and at 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 [36], 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

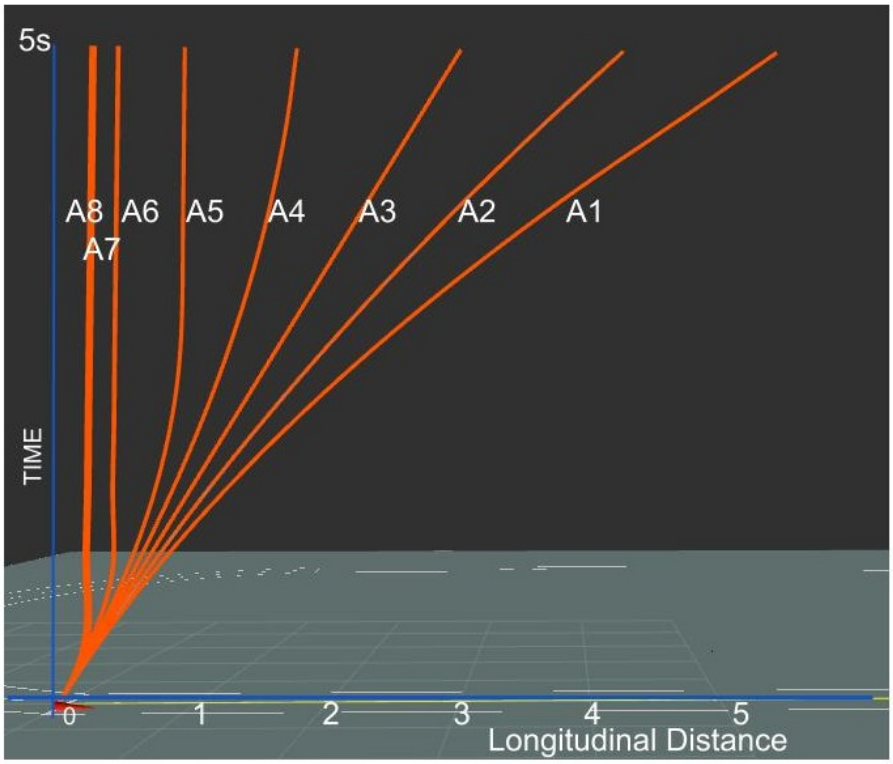


Figure 4.8: Distance traversed vs time with accelerations A1-A8 from 4.7 in time horizon(5s), Initial position = (0,0) velocity = $0.6ms^{-1}$, target velocity = 1.5 & 0 ms^{-1}

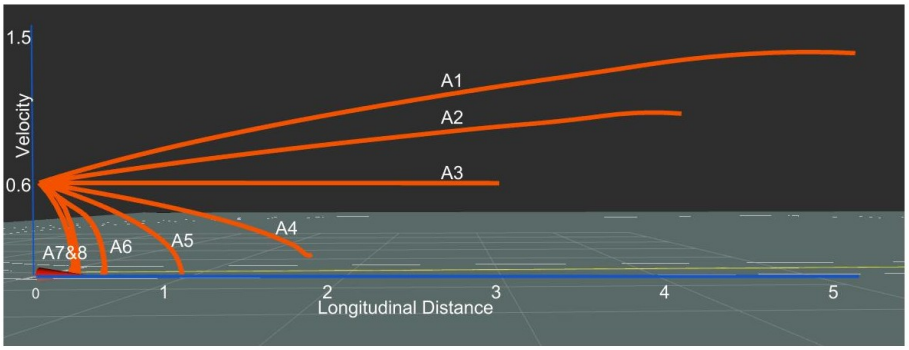


Figure 4.9: Distance traversed vs Velocity achieved with accelerations A1-A8 from 4.7 in time horizon(5s), Initial position = (0,0) velocity = $0.6ms^{-1}$, target velocity = 1.5 & 0 ms^{-1}

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

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.

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

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, density of this space can be increased by increasing the acceleration profiles and lateral samples.

check if own figure is needed - remove that sampling terminal states line

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.

Check if this below suggestion which is not implemented OK to be written here - or mention in implementation due to time constraints

4.5.4 Checking for Static Obstacles

The main 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. A road parallel model in Frenet frame is employed for collision checking ignoring the orientation of obstacles with respect to road as described in [41]. Using a road parallel method and dilating the obstacle for safety reduces computational complexity.

In the first step of collision checking, obstacle coordinates are transformed into Frenet frame and represented by a length and width parallel to road. In second step, the trajectory in consideration is checked if it has a collision in longitudinal dimension (s coordinate) for length of the obstacle. As shown in Figure 4.12 trajectories T0,T1

check if all things described in road parallel model for collision checking should be discussed here

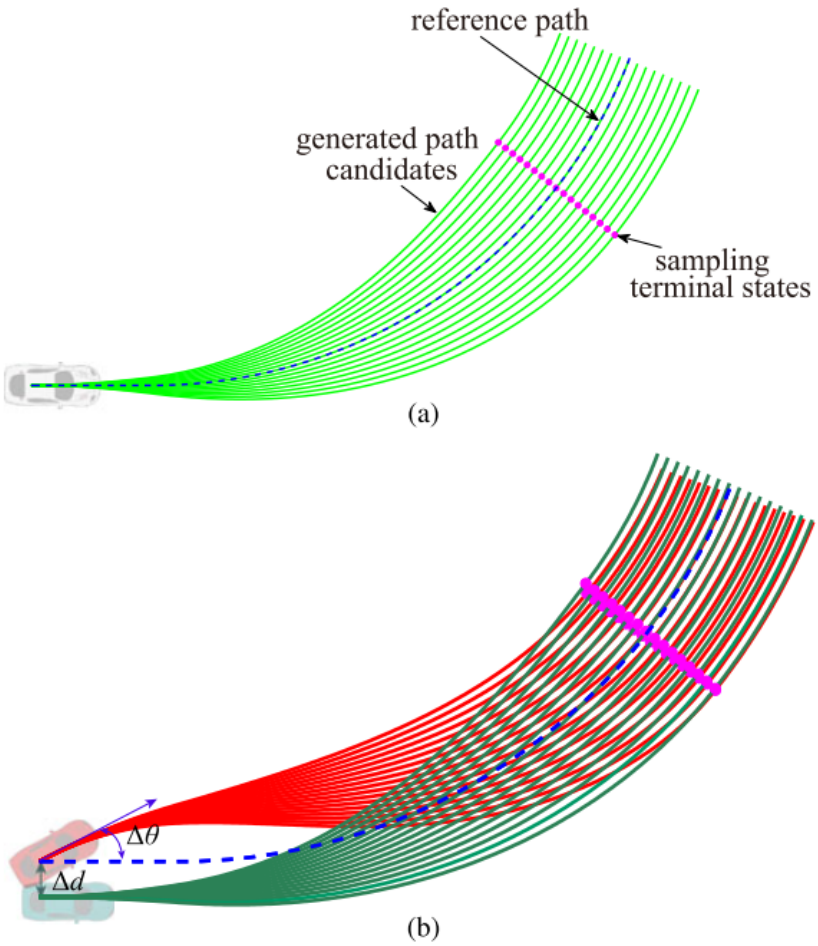


Figure 4.10: Path candidates generation results. (a) $l_0 = 0$ and $\theta_0 = 0$ (b) $l_0 = \Delta_d$ and $\theta_0 = \Delta\theta$. [36]

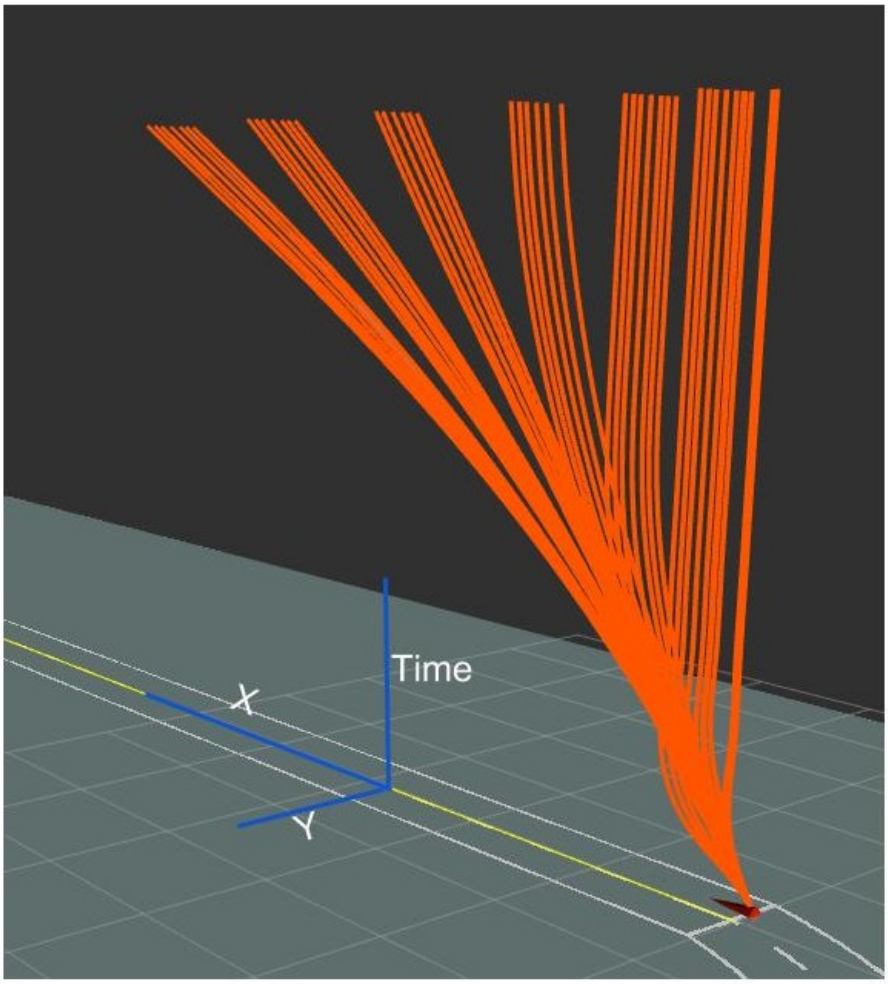


Figure 4.11: Total search space in xyt

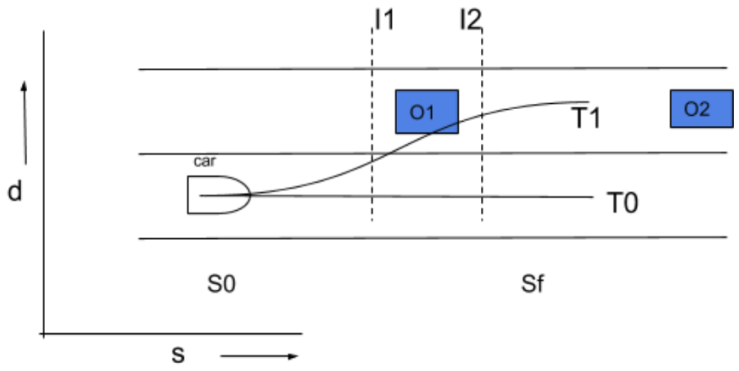


Figure 4.12: Collision check for static obstacles

intersect in S for obstacle O1 and not for obstacle O2. The next step is to find this intersection region, I1 to I2 in Figure 4.12, 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 greater than safety value as described in 4.7.

$$|d_{\text{ego}} - d_{\text{obst}}| > car\_width/2 + obstacle\_width/2 + safety\_margin \tag{4.7}$$

resent the
ameters
erms of
$_w, o_w$ etc

It is clearly 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. Due to properties of cubic polynomials it is sufficient to check for the collision at start, end, middle and min max D if that falls in the intersected path.

4.5.5 Checking for Dynamic Obstacles

In dynamic environments such as cities it is key to plan trajectories predicting the position of other obstacles and not colliding into them. Predicting the future behavior of obstacles is a challenging part in 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 for planning duration similar to [4] or moving in opposite direction in lane or moving across the lane based on 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 or orientation will be evaluated in next cycle thus keeping the vehicle safe from collision.

The collision check for dynamic obstacles has one extra check in time dimension as compared to checking for static obstacles, it is inspired from forbidden regions calculation as discussed in [18]. In step one the intersection in S coordinate for obstacle and ego vehicle is found, here the length of obstacle is dilated over the distance traveled by obstacle as represented by dotted line ahead of 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 similar way to static obstacle collision check. If there is 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, buffer of 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 [51] on safety distances for self driving cars. This thesis implements simple 2 second rule to safety and this is a tunable parameter which can be used to increase driving aggression. Figure 4.14a indicates a similar situation for the collision in time dimension for scenario presented in Figure 4.13 with difference that the obstacle is in different lane.

To check collision in time dimension, time difference between the ego vehicle and the obstacle at the S dimension intersection borders is checked as shown in figure **??**. If the gap is greater than 2 seconds at all the times and doesn't change sign then there is no collision, Figure 4.14a and 4.14b present a situation where the time to collision between ego vehicle and obstacle reduces but there is no collision. If the time difference between ego vehicle and the obstacle has an opposite sign at the start and end of the intersection region then it detected as collision. As difference indicates time to collision, and time and distance are continuous a change in sign in time difference indicates that in between at some point there was a collision(difference = 0). Figure 4.15a and 4.15b present a situation where the projected trajectory of the ego vehicle will collide with dynamic obstacle. Figures 4.16a represent situation where the projected trajectory of ego vehicle collides with obstacle moving in opposite direction and Figure 4.16b represents where the ego vehicles projected trajectory will not collide the vehicle in opposite lane.

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, trajectory is considered to be in collision if there is a collision in S dimension and there will anyways be collision in D dimension due to dilation of the pedestrian width to 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.

Collision checking in dynamic environments approximated as path from start to end with a linear velocity, in reality this is not true and there will be acceleration

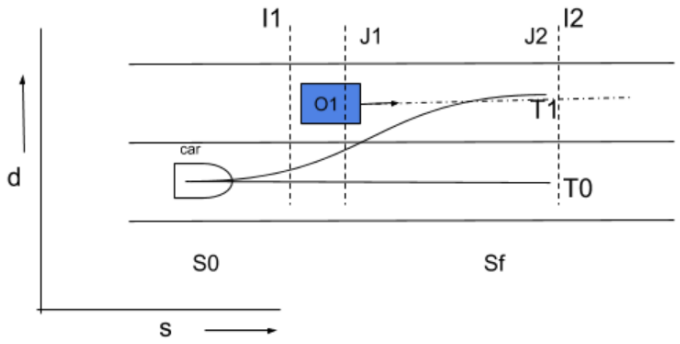
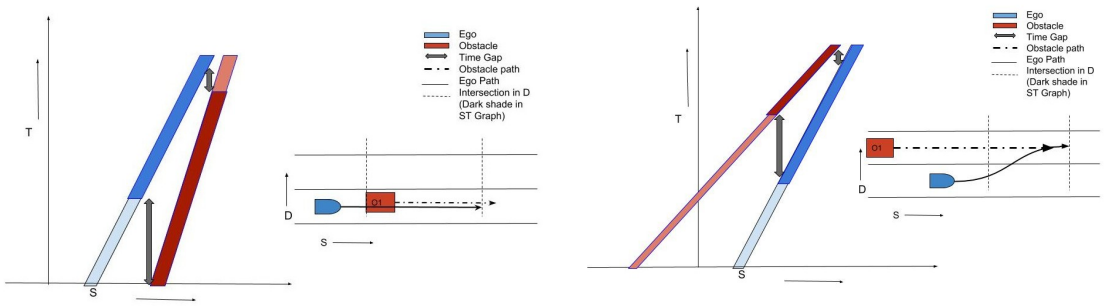


Figure 4.13: Collision check for Dynamic obstacles



(a) Ego vehicle projected trajectory getting close to slow moving obstacle ahead

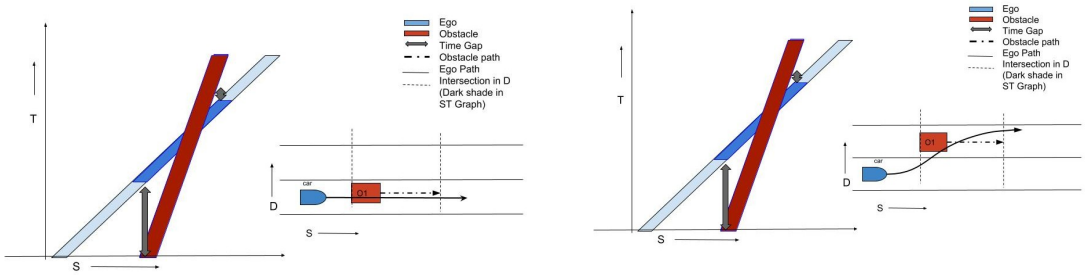
(b) Ego vehicle projected trajectory getting close to a fast moving dynamic obstacle in next lane while performing a lane change

Figure 4.14: Ego vehicle getting close to dynamic obstacale but not colliding

and deceleration by ego vehicle. Figure 4.17 presents scenario where the ego vehicle with $0.1ms^{-1}$ velocity is accelerated at $0.4ms^{-2}$, the corresponding approximated line for 5s if line with uniform velocity of $1.1ms^{-1}$. For this approximation to be safe the time difference between the original path and the approximated line should never be greater than $2s$, the maximum time difference as presented by the black line is $0.8s$. The difference is affected mainly by the acceleration and deceleration values with higher values causing higher shift. Due to limits on maximum speed, acceleration and minimum speed of zero, the approximation holds good for the 5s horizon. To reduce the safety margin to less than 1s, simply adding checks at every second instead of the start and end are sufficient.

uld I write mathem- al proof t this ap- ximation ks here or ppendix??

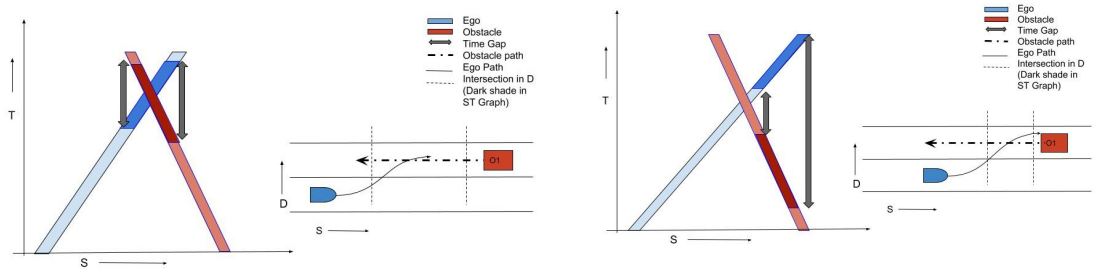
npared to



(a) Ego vehicle projected trajectory colliding with obstacle ahead

(b) Ego vehicle projected trajectory colliding with obstacle ahead while performing lane change

Figure 4.15: Ego vehicle projected trajectory colliding with obstacle in future.



(a) Ego vehicle projected trajectory colliding with obstacle moving in opposite direction

(b) Ego vehicle projected trajectory performing a lane change and not colliding with obstacle in next lane

Figure 4.16: Ego vehicles interaction with

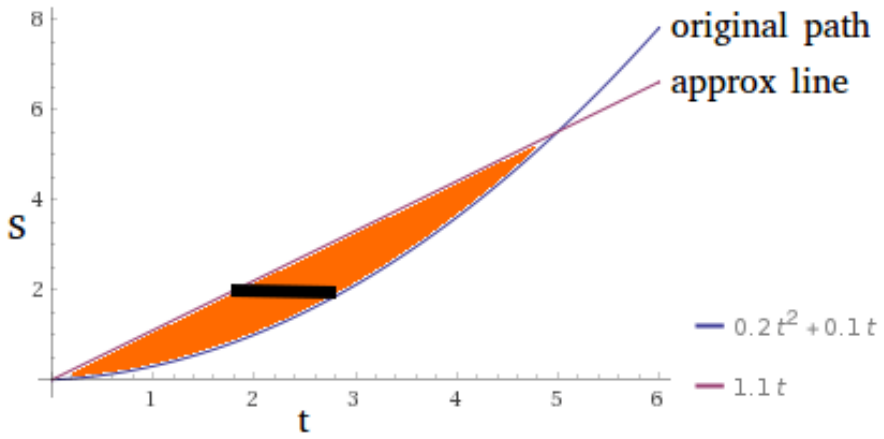


Figure 4.17: Approximation of Parabolic Path to a line

4.5.6 Cost Functions and Trajectory Selection

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

link to chapter s in related ks

This thesis implements a simple cost function as represented in 4.8, explicit smoothness costs are not considered as the target validation platform is a model car with no humans inside and limitations of model car to track a fine trajectory.

$$cost = |V_a - V_t| + |a_t| + |(d_t - d_e) * k1| + |(d_p - d_e) * k2| \tag{4.8}$$

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 4.8 promote higher accelerations if the target and current velocity difference is 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 closer to the previous selection thus limiting the shifts in path selection and maintaining continuity.

Initially all the sampled trajectories are assigned the costs based on the cost function 4.8 and sorted, then the trajectory with the lowest cost is evaluated for collisions first. The list is sorted again and evaluated till the top of the list has lowest cost and the trajectory is evaluated.

4.5.7 Velocity Planning

Velocity planning is an important aspect of the planer, in general behavioral layer defines the target velocity based on speed regulation, other traffic participants, required behavior, 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 4.9.

$$V_{max} = min(\sqrt{Acc_{MaxLat}/|k(s)|}, V_{limit}) \tag{4.9}$$

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

4.6 Trajectory Follower

Check whether to refer to MIG trajectory controller or write in short about the controller.

Chapter 5
Implementation

This chapter details on the implementation of the concepts discussed in 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.

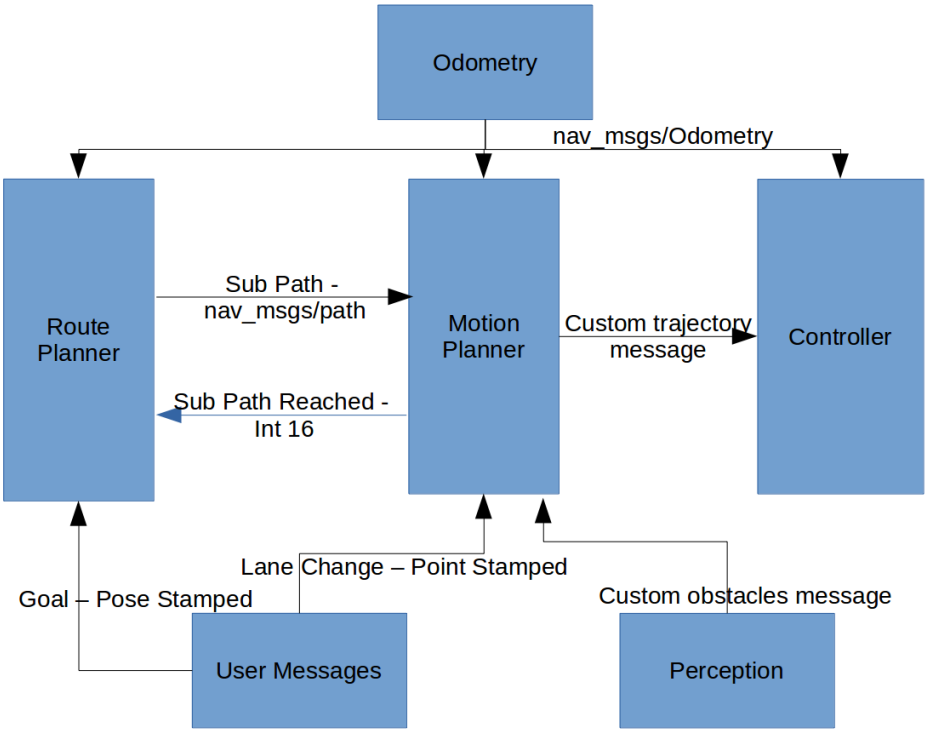


Figure 5.1: Message Flow across Modules

5.1 Route Planner

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

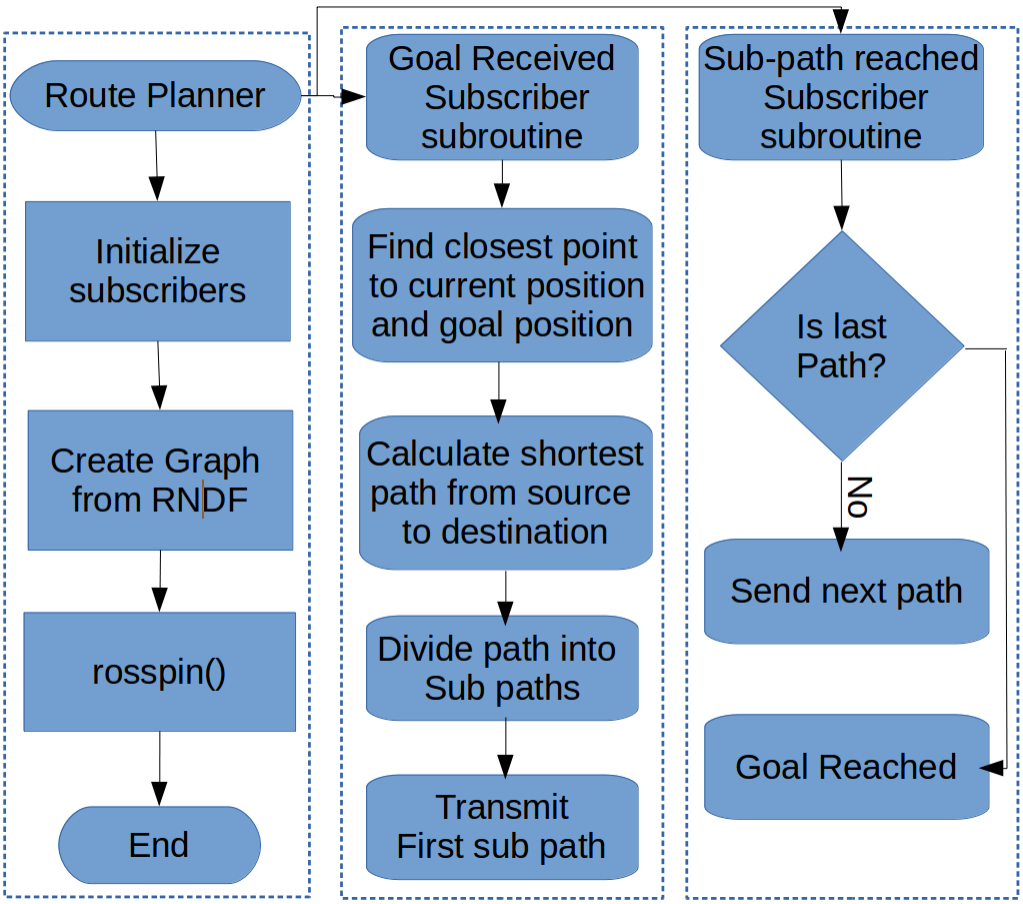


Figure 5.2: Route Planner Functions

```
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 sub divided into three sub classes names 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

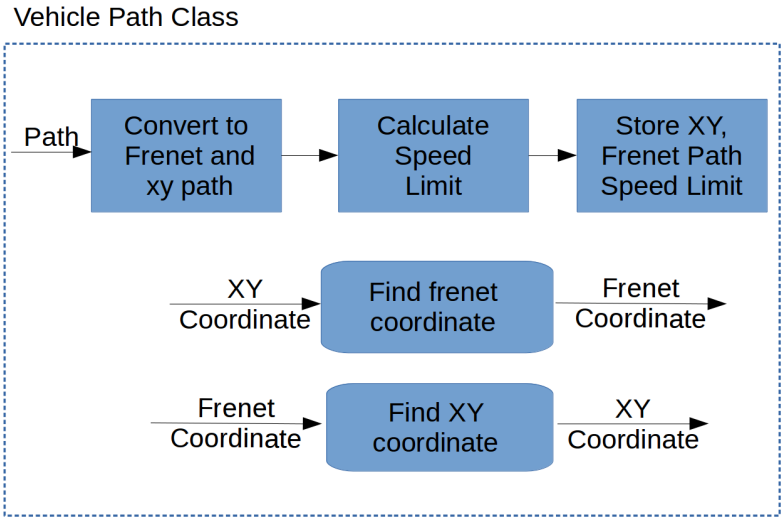


Figure 5.3: Vehicle Path Class

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 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 affects 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 find the best trajectory which is collision free and closer to destination. Once the best trajectory is found it converts to the format specific to the controller and transmits it. Figure 5.5 describes the interactions between different functions in motion planning.

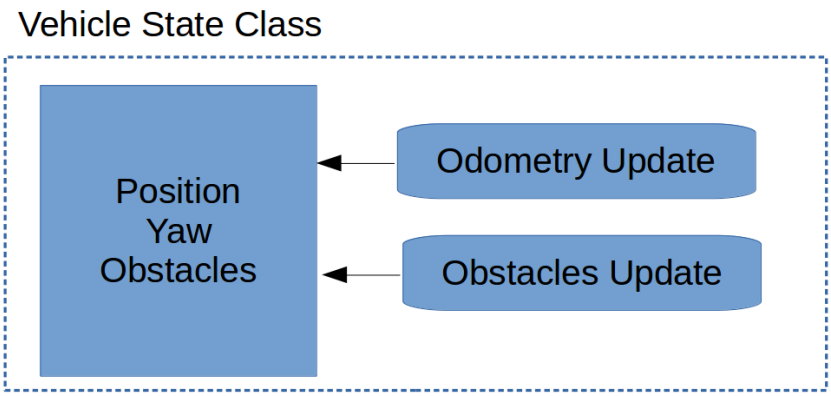


Figure 5.4: Vehicle State Class

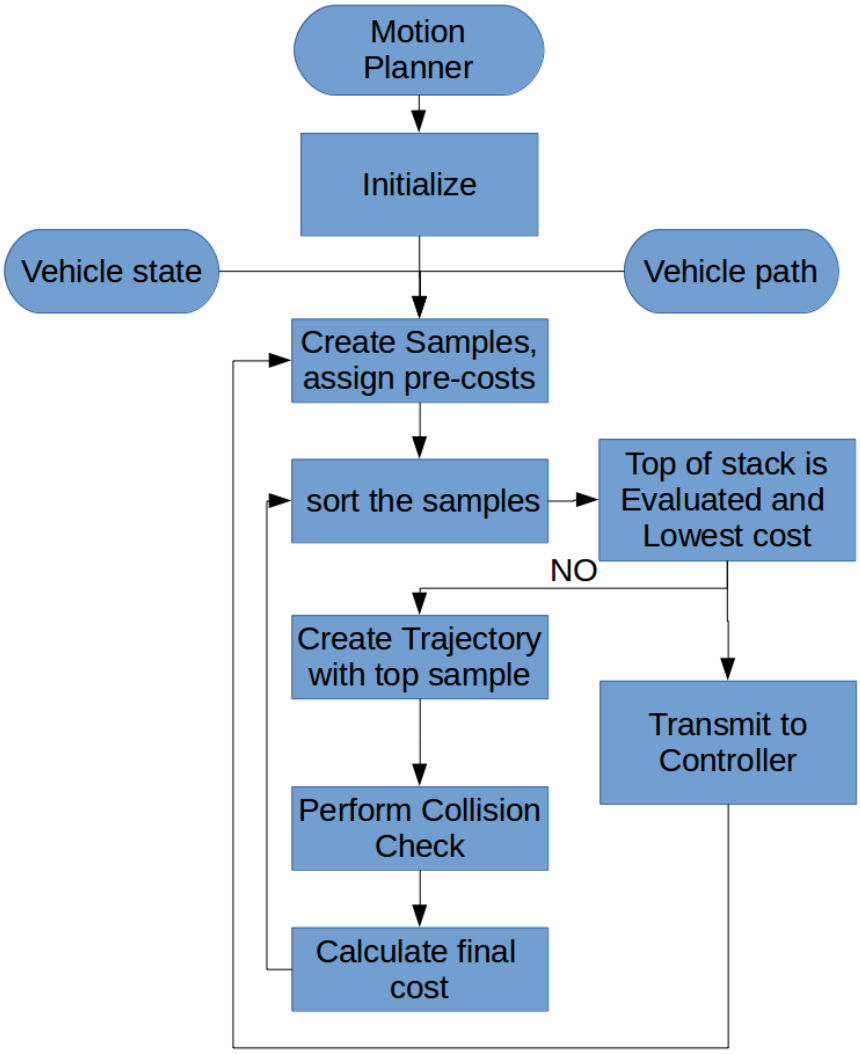


Figure 5.5: Motion Planner Functions

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 organized as follows: Section 6.1 discusses systematic evaluation of the planner by exposing it to various scenarios equivalent to on-road driving conditions. The next section 6.2 discusses a criteria based evaluation for the planner similar to any algorithm in the form of 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 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 on each experiment.

6.1.1 Lane blocked

In driving scenario 6.1, driving lane is blocked by a static obstacle and a slow moving obstacle is in the next lane, here ego vehicle drives slowly till it finds enough room in the next lane, once obstacle is avoided the robot continues to shift into intended lane and drives with increasing speed.

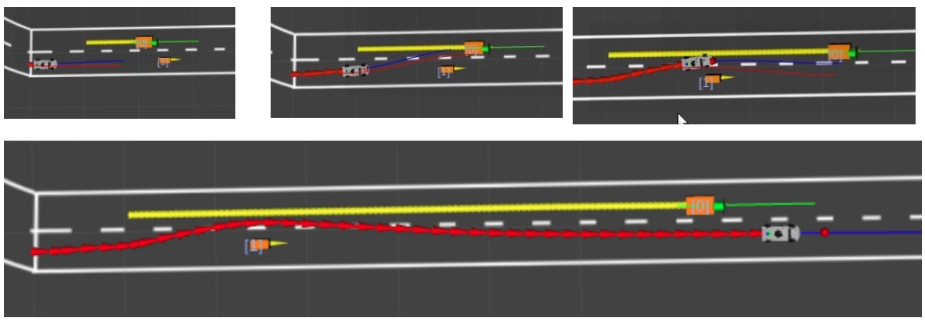


Figure 6.1: Driving Lane Blocked

6.1.2 Slow Moving Traffic

In situation 1 presented in Figure 6.2 ego vehicle starts changing into left lane once a slow moving obstacle is encountered, once the obstacle is passed it starts driving into right lane again. In situation 2 presented in Figure 6.3 there are two slow moving obstacles ahead, once the ego vehicle overtakes the initial obstacle it shifts to original lane as intended but it encounters the second slow moving obstacle and shifts to left lane again. This behavior is caused because of locally optimal cost functions driving the ego vehicle into intended lane without knowledge of long term planning information. A behavioral layer with longer scenario analysis horizon will result in better path selection.

Figure 6.2: Slow Moving Traffic Situation 1

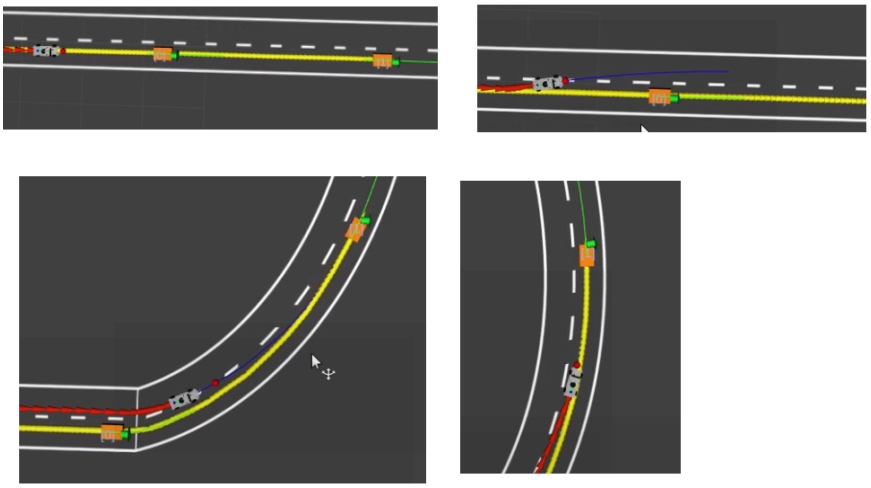


Figure 6.3: Slow Moving Traffic Situation 2

6.1.3 Merging into traffic

In scenario presented in Figure 6.4, lane changing is requested to merge into the traffic in left lane. Here ego vehicle speed is $1ms^(-1)$ and obstacle speed is $0.6m^(-1)$. Initially lane change does not occur as cost functions are tuned to maintain speed over maintaining required lane. As the vehicle enters the curve, target driving speed

is reduced and the vehicle merges into the traffic in left lane. Depending on which portion of the lane the ego vehicle is in i.e, near intersections or exits target lane should have higher priority over maintaining speed and during rest of the regions target speed should be of higher priority to reach destination quickly. Cost functions implemented in this thesis provide flexibility in tuning behavior of the ego vehicle.

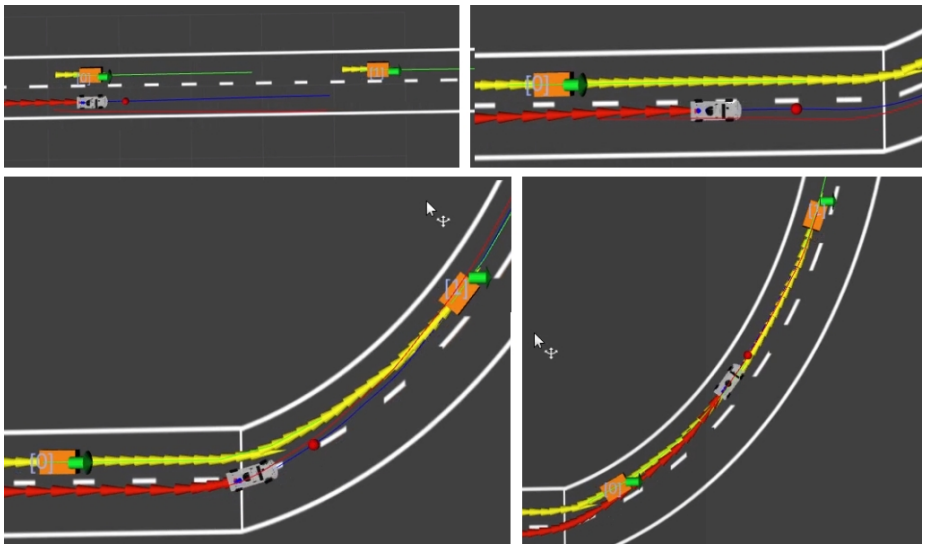


Figure 6.4: Merging into Traffic

6.1.4 Merging into next lane with opposite traffic

In scenario presented in Figure 6.5 driving lane is blocked by a series of obstacles and the left lane is occupied by a moving obstacle. Ego vehicle starts slow in the driving lane and waits till the obstacle is passed in the left lane and starts driving forward.

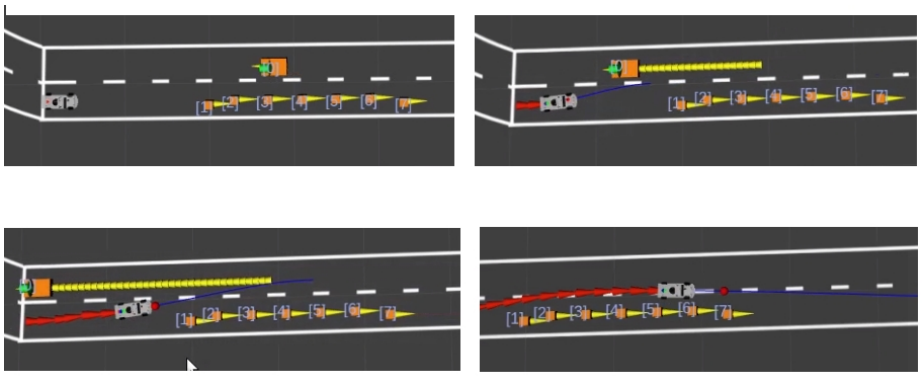


Figure 6.5: Lane blocked by series of static obstacles and vehicle in next lane driving opposite

This situation may lead to ego vehicle getting stuck in middle of the road. If driving speed of ego vehicle is low, temporal horizon limits ego vehicles lookahead distance into future. If a fast moving obstacle in left lane not visible in 7 seconds(5 planning and 2s of safety) of temporal horizon then the ego vehicle starts lane change and if there is time to abort it will abort and if not the ego vehicle will stop in middle of the lane due to no path ahead, if the obstacle proceeds without stopping for ego vehicle. This can be avoided by a behavioral layer with longer spatial scenario analysis horizon. As the planner discussed in this thesis is not created for controlling the vehicle to drive backwards, a different planner equivalent to off road planner must be used.

6.1.5 Road Blocked or Pedestrian Ahead

In scenario presented in Figure 6.6, road is blocked by a series of static obstacles, the vehicle enters empty left lane, slows down and finally stops when it cannot find route ahead.

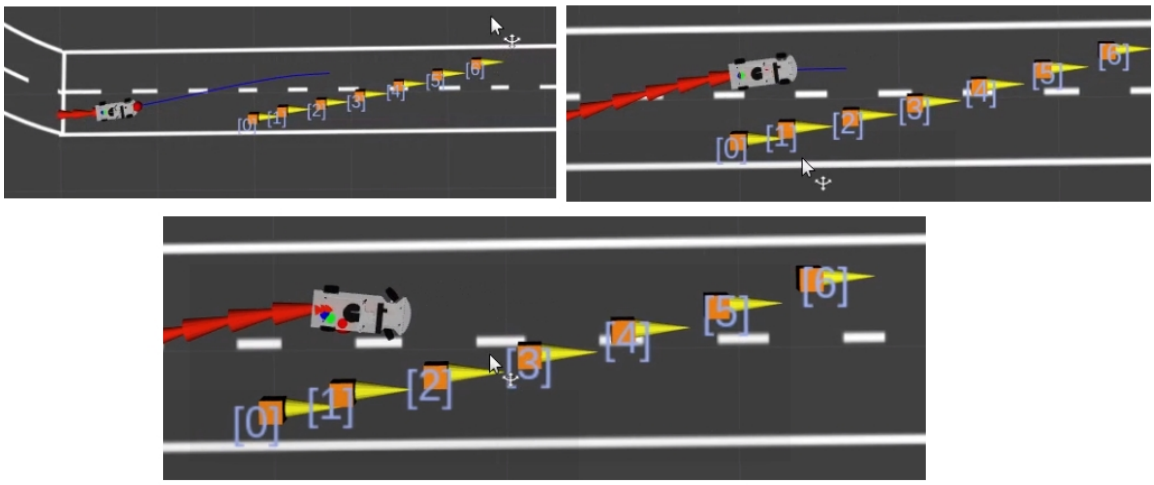


Figure 6.6: Road Blocked by series of obstacles

A pedestrian on road is considered similar to a road blocking, in this case as shown in Figure 6.7 the ego vehicle initially drives at full speed, then the vehicle slows down(shorter blue line representing a slower speed) and the robot finally comes to halt few meters ahead of the pedestrian. This is a tunable parameter and currently at maximum value for safety.

6.1.6 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 sudden breaking of vehicles ahead.

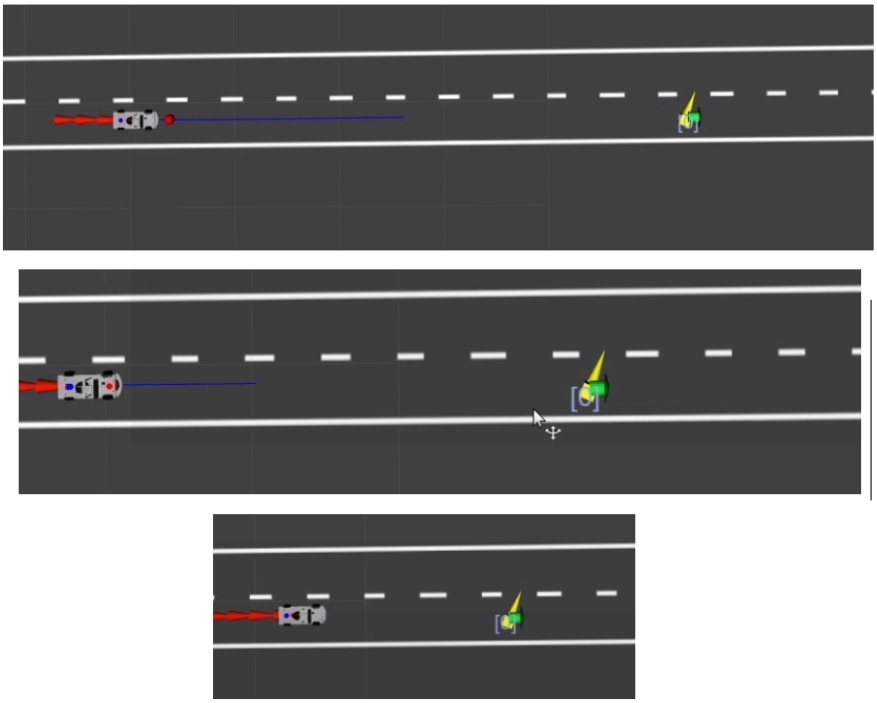


Figure 6.7: Pedestrian Ahead on Road

In scenario presented in Figure 6.8, there are two situations. In situation 1 the car aborts a lane change when the slow moving dynamic obstacle in left lane is detected, then once the dynamic obstacle is passed the vehicle shifts to left lane to avoid the stopped dynamic obstacle in driving lane. This situation is similar when a vehicle ahead stops to drop off a passenger or waiting for parking spot. In situation 2, the car doesn't choose lane change initially and slows down till it finds enough room in left lane to drive ahead of the stopped dynamic obstacle.

In scenario presented in Figure 6.9 there are two situations with different thresholds for safety, in situation 1 a safe 2s+ distance to obstacles ahead is chosen, here the ego vehicle stays far away from the vehicles ahead and when it stops it stops relatively farther from the vehicles ahead. In situation 2, the threshold has been adjusted to 0.5s leading to a aggressive behavior of ego vehicle. The ego vehicle drives closer to the obstacles ahead and when the dynamic obstacles ahead stop suddenly, distance between the ego vehicle and the obstacles ahead is very narrow.

6.2 Criteria Based Evaluation

In this section, proposed planner is validated against the common criteria of evaluating any algorithm, i.e. optimality, feasibility, completeness, runtime and approach.

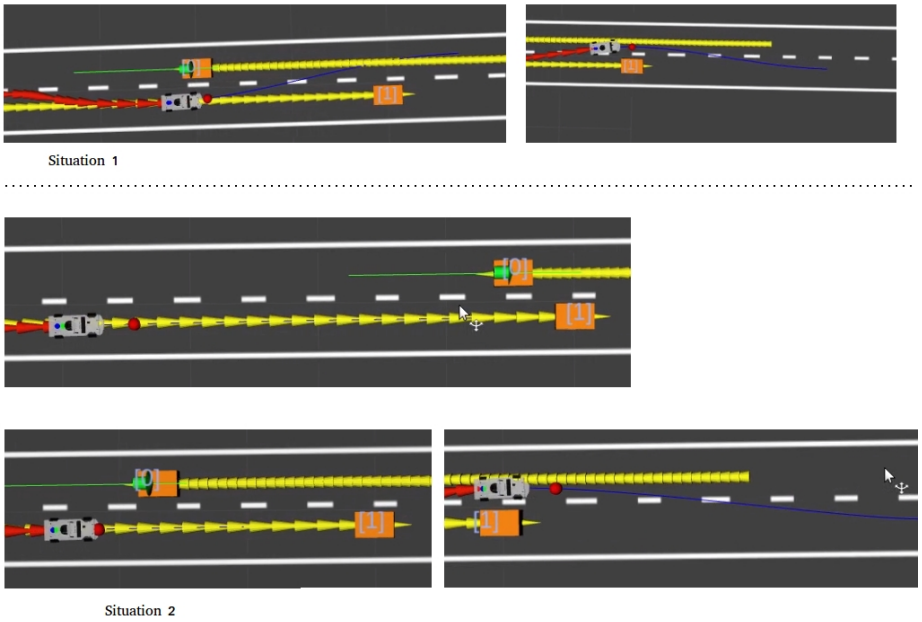


Figure 6.8: Dynamic Obstacle Ahead stops in middle of road

6.2.1 Optimality

In this thesis we discuss about 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 planner to create a longer and better path but due to the unpredictability of the environment, plan created will not be valid after certain duration, a larger horizon will also increase the run-time of algorithm. The planner proposed in this thesis is only a local planner and always needs inputs from a behavioral layer or a global planner to choose target lane, velocity etc thus a short planning horizon is suitable for this proposed planner.

An example of how horizon will affect optimal planning for current planner is shown in figure 6.10. 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 chosen, depending on situation one is efficient sometimes and other in others. These situations can be improved by lane selection algorithm in behavioral layer which looks for occupancy of different lanes and suggest the one best suitable lane. Similarly if an exit has to be taken on road, a long horizon would choose a plan with reduced speed compared to high speed path with short horizon. This can also be solved by having a velocity planner in the behavioral layer.

ve to future
ks

Another horizon generally in discussion for a planner is spatial horizon which discusses how long is the path generated, as per this planning criteria at low velocities the spatial horizon considered is very small thus the planner may not make right

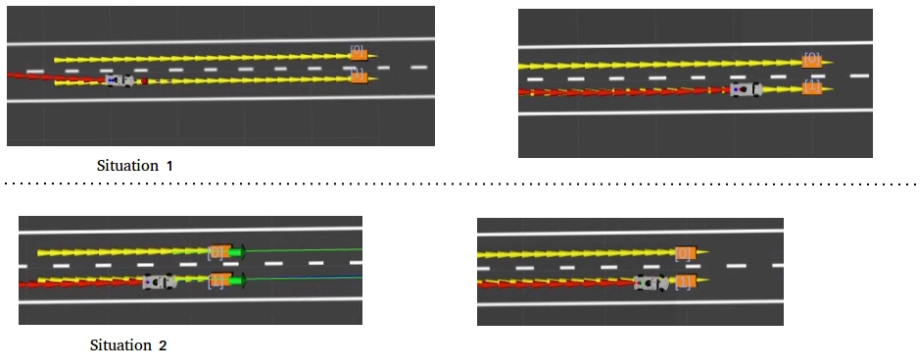


Figure 6.9: Two Dynamic Obstacles ahead stop suddenly

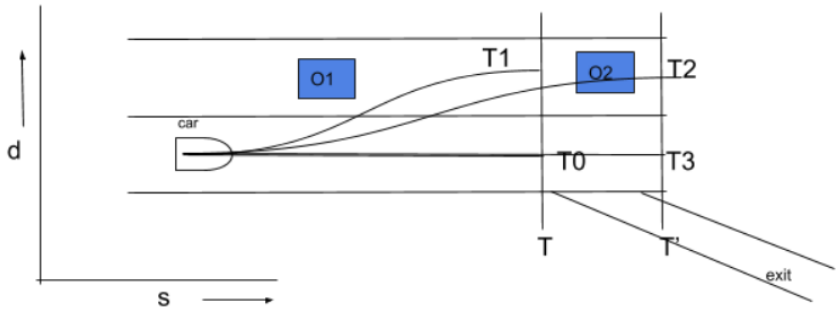


Figure 6.10: Horizon Optimality reference

decisions because of conditions like missing the obstacles ahead etc. This shortcoming can be improved by creating a spatial path with longer horizon in behavioral layer at lower speeds and allowing the local planner to follow new spatial path than following the global reference path. This is an efficient method as the path planning is generally less expensive than trajectory planning. As shown in figure 6.11, following the original reference path(solid line) will lead to trajectories that turn a lot causing discomfort due to obstacles on the side of road that enter the road, thus by using an optimized reference path(dotted line), ego vehicle can plan efficiently even using short horizons.

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 a best selection is possible.

From the above discussion it can be stated that a planner that has a longer spatial horizon for path planning and short time horizons for trajectory planning will lead to an efficient planner.

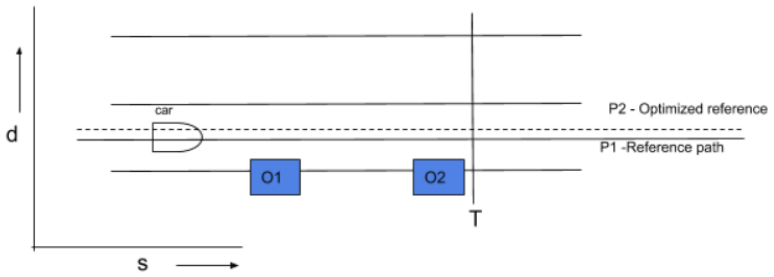


Figure 6.11: Optimized Reference Path

6.2.2 Feasibility

Ability of the vehicle to traverse the created trajectory determines feasibility, generally 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, higher speeds require fifth or higher order splines to maintain continuity in path and speed. Discussions in [41] [39] throw light on how to achieve higher degrees of smoothness, which approach is better in which driving conditions. As the intended application for this thesis is a modelcar, constant acceleration profiles are used due to limitation in ability of the car to track small changes in velocity and inaccuracies in measurement. These can be easily replaced by a smoother higher order polynomials with a better hardware platform. Consistency in paths evaluated with respect to previous plan is another factor in feasibility, the current planner penalizes trajectories deviating from previous plan and also takes into account current orientation of the vehicle in choosing a path. Thus creating smoother transitions from one state to another by respecting current driving orientation.

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, probability of finding a solution approaches to one as the number of samples increases. If there are higher number of samples in the configuration space then higher are the chances of finding a solution. If a planner cannot find a solution within sampled region it forces the car to go into emergency manoeuvre.

In Figure 6.12, 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. Second method is to keep on sampling till 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 also. The planner proposed in this thesis implements a combination of both methods, as many samples as possible but evaluates only till a feasible solution is found. It is recommended to achieve completeness for safety purposes in autonomous driving.

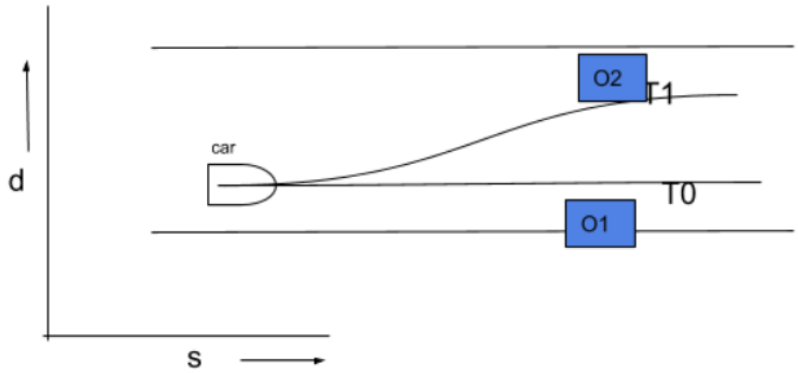


Figure 6.12: Probablistic Completeness

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 generally fare well and run on a low computational hardware. In contrast lattice based approaches such as [41] [54] [55] 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 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 evaluation of trajectories, generally simulation based methods as discussed in [29] are generally expensive generally 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 [17] which has a performance better than the simulation based algorithms can also be employed in future.

te about
execution
e for dif-
nt number
amples,
ximum ex-
tion time,
imum time
n evalu-
on result etc

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 a best trajectory it stops evaluating the other trajectories as presented in subsection 4.5.6. This 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.

nparison
e doesn't
ke sense as
re are no
nbers to
npare with
er plan-
s, theoret-
compar-
ns can be
sented in
ted work
orms of
rt comings
different
nners. Here
y be just
a table
in CMU
DISS shui
sis with
nparision to
er planner

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 target 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 evaluating all trajectories only 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 small car like robots, comfort is not a mandatory criterion in trajectory selection and major effort has been kept 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. The planner has been tested on the car platform without obstacles and performed reasonably because of the tracking errors from the 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] 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

[2] 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

[3] 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

[4] 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. 6, 10, 30

[5] 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

[6] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Springer Publishing Company, Incorporated, 1st edition, 2009. 5, 6, 9

[7] 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

[8] 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, 23, 24

[9] Gerald Cook. *Robot Navigation*, pages 324–. Wiley-IEEE Press, 2011. 18

[10] 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

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

[12] 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. 7

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

[14] 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

[15] Gaston A. Fiore and David L. Darmofal. A robust motion planning approach for autonomous driving in urban areas. 2008. 6, 7

[16] 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

[17] 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. 6, 7, 12, 48

[18] Daniel Graff. *Programming and Managing Swarms of Mobile Robots:A Systemic Approach*. PhD thesis, Technischen Universitat Berlin, 2016. 31

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

[20] 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. 6

[21] 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

[22] THOMAS M. HOWARD. *Adaptive model predictive motion planning for navigation in complex environments*. PhD thesis, Carnegie Mellon University, 2009. 9

[23] 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. 7

[24] 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

[25] 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

[26] Lydia E. Kavraki and Jean-Claude Latombe. Probabilistic roadmaps for robot path planning, 1998. 7

[27] 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

[28] 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

[29] Sascha Kolski. *Autonomous Driving in Dynamic Environments*. PhD thesis, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 2008. 6, 9, 11, 47

[30] 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

[31] 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. 7

[32] 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. 6

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

[34] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998. 7

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

[36] 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. 6, 9, 22, 25, 28

[37] 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

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

[39] 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. 46

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

[41] Matthew McNaughton. *Parallel Algorithms for Real-time Motion Planning*. PhD thesis, Carnegie Mellon University, 2011. 6, 8, 10, 11, 27, 46, 47

[42] 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

[43] 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

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

[45] 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

[46] 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

[47] 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

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

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

[50] 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

[51] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *CoRR*, abs/1708.06374, 2017. 31

[52] 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

[53] 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

[54] Shuiying. Wang. *State Lattice-based Motion Planning for Autonomous On-Road Driving*. PhD thesis, Freie Universität Berlin, 2015. 6, 9, 22, 23, 47

[55] 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, 25, 47

[56] Marios Xanthidis, Ioannis M. Rekleitis, and Jason M. O'Kane. RRT+ : Fast planning for high-dimensional configuration spaces. *CoRR*, abs/1612.07333, 2016. 7

[57] 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

[58] 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. 6, 8, 9, 10, 19, 22

[59] 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

Proof thats its sufficient to check only at borders and only if global min or max exists in the intersection interval for collision checking with static obstacles

Proof that the approximating collision checking for dynamic obstacles as straight line is fine.