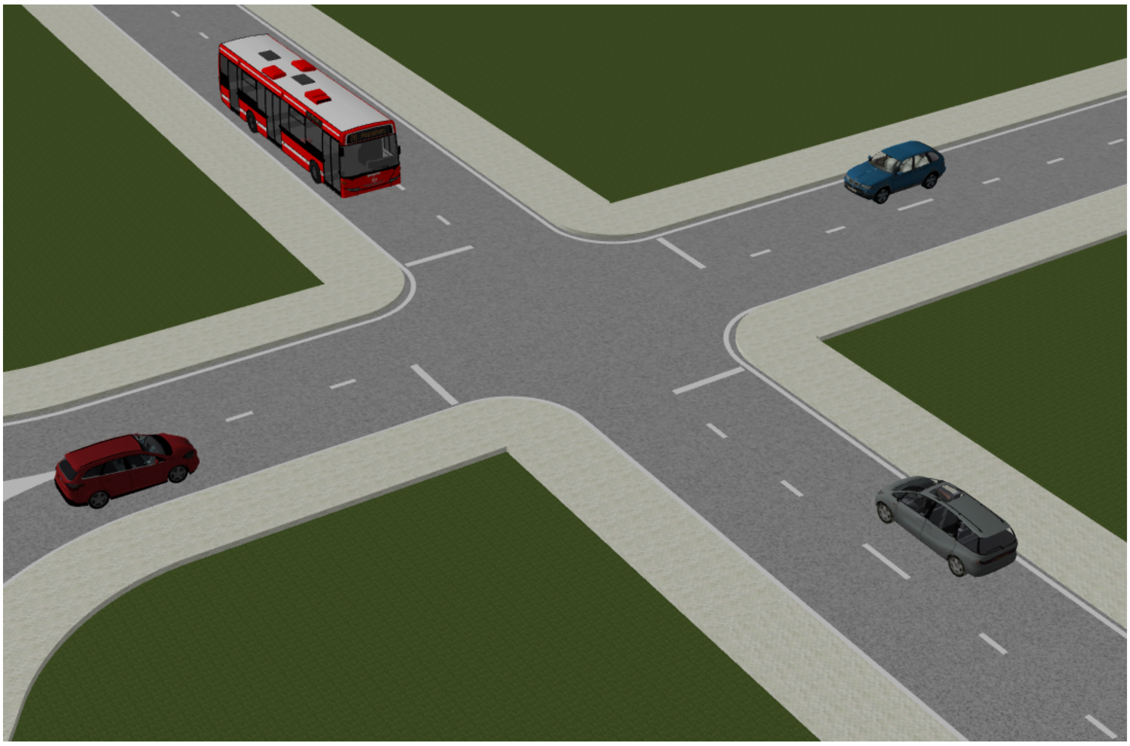
CHALMERS
UNIVERSITY OF TECHNOLOGY



Velocity planning approach for autonomous vehicles

An approach to address traffic negotiation for autonomous vehicles

Master's thesis in Systems, Control and Mechatronics

Abhiram Rahatgaonkar

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2015

Master's thesis 2016:NN

Velocity planning approach for autonomous vehicles

An approach to address traffic negotiations for autonomous vehicles

ABHIRAM RAHATGAONKAR

Department of Signals and Systems
Chalmers University of Technology
Gothenburg, Sweden 2016

Velocity planning approach for autonomous vehicles
An approach to address traffic negotiation for autonomous vehicles
ABHIRAM RAHATGAONKAR

Supervisor: George Dibben, Scania AB
Examiner: Paolo Falcone, Signals and Systems, Chalmers University

Master's Thesis 2015:NN
Signals and Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A traffic junction environment modelled in PreScan showing a typical traffic situation.

Typeset in LaTeX
Printed by [Name of printing company]
Gothenburg, Sweden 2015

Velocity planning approach for autonomous vehicles
An approach to address traffic negotiation for autonomous vehicles
ABHIRAM RAHATGAONKAR
Department of Signals and Systems
Chalmers University of Technology

Abstract

Motion planning being one of the core functionalities of autonomous driving system, has been studied and investigated with great interest by the developers across industries and academia. Collision avoidance is one of the major challenges within trajectory planning problem (TPP). This master thesis is inspired from the 'concept of decomposition of trajectory planning problem in to path planning and velocity planning' which results in great reduction in the computational complexity of the TPP. The work in this project is focused on the velocity planning with the known path so that the collision with the moving vehicles is avoided. Concept of space time scheduling is used along with the A* (A star) search algorithm in a particular fashion to achieve the time optimal speed profile. The approach developed in this work is deterministic but has possibility to work with probabilistic motion model of the vehicles in the traffic. It also has great potential to work with the different road geometries and different vehicle maneuvers such as merging, yielding, intersection, etc. Hence, it is very promising approach for velocity planning with possibilities to work with variety of scenarios.

Keywords: Collision check, velocity planning, A* search, space-time

Acknowledgements

This masters thesis is a result of all the knowledge I have gained along with the work experiences I have had until now. I am grateful to the Chalmers University of Technology for the education in advanced engineering along with the well nourished international study environment. Courses studied at Chalmers have been very useful during the development work of the thesis. I would also like to acknowledge the Indian Institute of Technology Guwahati for undergraduate education that made me capable to study and work on the global level. I am very thankful to Scania AB for the opportunity to work on the master thesis in the area of autonomous driving.

This work has been closely supervised by Mr. George Dibben (Scania AB) who has not only given his valuable time for discussing ideas and issues but also provided all the support for carrying out the thesis work. His experience in industry has been very helpful while making wise choices at several important occasions during the thesis work. Prof. Paolo Falcone from the Chalmers University of Technology has provided valuable inputs time to time while critically examining the thesis work. This work is also deeply influenced by the inputs from the experts working in the industry. Expertise of Mr. Marcello Cirillo (Scania AB), Mr. Magnus Granström (Scania AB) and Mr. Niclas Evestedt (Linköping University) in the 'motion planning' have resulted in a technically strong approach.

ABHIRAM RAHATGAONKAR, Gothenburg, 09 2016

Contents

List of Figures

List of Tables

1
Introduction

Autonomous driving is one of the advanced engineering areas where most of the automotive industries are paying attention today. Even though there exists several working prototypes already, there are many critical situations and problems that are to be solved to have a truly autonomous vehicle. Even though commercial autonomous vehicles are yet to come on the street, driver's assisting systems for special scenarios are being introduced in vehicles today. This can be seen as the first step towards automation. Automatic cruise control is one such example. Since such system cannot be realized without accurate sensor readings, lot of efforts are also put into the sensor technology and sensor fusion to get best out of the control algorithms.

A fully autonomous driving vehicle must be capable of making its own decision about its motion. This requirement highlights many critical challenges in the development of autonomous driving system. One of these challenges is autonomous trajectory planning which is essential for autonomous driving systems. Trajectory planning of an object consists of path planning and velocity planning. Hence, trajectory planning problem for autonomous vehicles becomes complex due to continuously changing environment. Another important problem that goes hand in hand with motion planning is the negotiation with other vehicles at busy intersections. Traffic negotiation is also dependant on traffic rules that provide priority to some vehicles over others in particular traffic scenario.

This project shall investigate motion of a bus at a typical traffic junction in order to generate a speed profile to achieve collision free motion with moving vehicles. Approach for the velocity planning of the bus moving along a known path (or pre-calculated path) is developed in this work. Functionality of the approach is simulated on the software called *PreScan* during the development process. Motion planning also needs to address negotiation with other vehicles in the traffic. Even though this work focuses on traffic junction, approach presented here has possibility of modification to different traffic scenarios and road geometries. Having a generalised approach with possibility to extend and modify it to complex situations is one of the motivations behind this work.

1.1 Background

A vehicle that can carry out all the real time driving tasks without any real time input from a human operator can be considered as a fully autonomous vehicle.

Even though autonomous driving is emerging today as an advanced engineering concept, it has been an active area of research for considerable amount of time. The Carnegie Mellon University Navigation Laboratory (NavLab) has been working in autonomous navigation area since 1984 [8]. In order to improve the safety, comfort and environmental friendliness of transportation, various semi-autonomous driving functions have been introduced in the vehicles by automotive industries and innovation giants [10, 13].

Fully autonomous vehicle implies a self governed vehicle without any human operator in all possible driving scenarios. In order to achieve this, vehicle must acquire all the abilities in terms of sensing and decision making. Hence, every single sensing and controlling function is being investigated and developed in dedicated research groups across academia and industries [11, 13, 12]. Inter-vehicle communication capabilities are also being developed for integrated transport systems for higher efficiency and driving performance [9]. As far as scope of this project is concerned, only single vehicle control is considered. Inter-vehicle communication is out of the scope of this project.

1.2 Related work

A fully autonomous vehicle must be able to perceive, predict and react appropriately to environment in similar or even better way than a normal human driver does. This results in a need of a system architecture for autonomous driving system that will incorporate various aspects of human abilities that play key role in driving. Needless to say that trajectory planning is just one of sub-blocks of the autonomous driving system. As the level of automation increases, complexity of system also increases.

In an attempt to explore and highlight the variety of challenges in autonomous driving, a fully autonomous approach was proposed in [2]. This paper also provides a basic system architecture consisting of various system blocks and corresponding functionalists. Similar planning framework is presented in the [3]. It focuses more on behaviours planning level for perceiving and avoiding static and dynamic obstacles. Based on these two works, one can identify important key system blocks in autonomous driving system:

- Sensors and digital maps
- Sensor fusion and environment Perception
- Localization
- Trajectory planning
- Motion control

Merging of two vehicles at a highway ramp is presented in [1] by applying concept of cooperative driving. It also includes acceleration and deceleration of the merging vehicle in decision making process. In [2], path generation is formulated as a non linear optimization problem with objective function penalising rapid changes in speed, acceleration and yaw rate.

In [7], an entire overview of autonomous driving systems is described. This work is very useful to understand concepts and challenges involved in each subsystem of the autonomous driving system. Trajectory planning is achieved by generating thousands of candidate trajectories and an optimal trajectory is dynamically calculated. Controller then selects throttle, brake and steering actuation by minimising the deviation from the planned trajectory and maximizing the comfort. This work also shows that the optimization can play a key role in trajectory planning problems.

One particular approach that is appealing in terms of computational complexity and generalization aspect is given by Kant and Zucker in [5]. Trajectory planning approach for a point object is proposed in this paper. This paper provides a systematic procedure of decomposing trajectory planning problem into two parts: path planning and velocity planning. It assumes that the trajectory of moving obstacles known. Based on static obstacles, an optimized (shortest) path is computed. Once the path is available, velocity is planned in a way to avoid the dynamically moving obstacles. This is achieved by an innovative approach of spacetime plot which enables us to generate a velocity profile along given path to avoid all the potential collisions in various future instances. Furthermore, the velocity profile can be adjusted to the speed, acceleration and yaw rate limitations along the path while reaching the destination in the fastest possible way.

The path planning between two points is dependent on two important factors: Static obstacles and dynamic obstacles. Through decomposition approach, it is possible to make a path using static obstacles and then generate a velocity profile that avoids the dynamically moving obstacles. Employing such trajectory planning scheme in real time will be able to address collision avoidance in real time. This kind of decomposition of problem results in the great reduction in computation and complexity of the problem [5]. Path planning problem itself has complexity of exponential order in terms of degrees of freedom of an autonomous vehicle. This complexity becomes huge for the practical scenarios with the variable shape and number of obstacles. Hence to make it computationally less demanding, approach of decomposition of trajectory planning problem is worth investigating for motion planning of autonomous vehicles on the streets.

2
Problem Definition

This chapter provides overall background for understanding the detailed approach presented in the next chapter. It consists of problem description and requirements of velocity planning function. Introduction to various terms and concepts used for developing this approach are also presented in this chapter.

2.1 Problem Description

Considering the idea of the decomposition of the trajectory planning problem discussed in previous chapter, this project will address the second subproblem of velocity planning. Path planning is the first step which depends on the static obstacles. However, velocity planning is the second step that deals with the collision avoidance with moving obstacles. Traffic junction or circles are usually quite constrained road geometries for heavy vehicles such as buses and trucks. Because of the long lengths, these vehicles usually do not have much of a room to change their path along the motion at such geometries. Hence, path is usually planned in the beginning of such maneuvers based on geometry of the road and static obstacles. This is also the reason why most of the times, these vehicles end up following a similar path at a particular road geometry. Therefore, this project focuses on velocity planing of vehicle with the preplanned path.

Objective Developing on-line velocity planning approach for a bus moving along a known (pre-calculated) path to avoid the collisions with the moving obstacles. Following points provide better idea about the problem statement:

- Single vehicle control: Only controllable vehicle is Bus which is called as ‘Ego vehicle’
- Random Traffic situations Other objects such as other vehicles, pedestrians, cyclists are not controllable and their trajectories are unknown to the bus throughout the maneuver
- Full autonomous driving: Inter vehicle communication or communication with outside station does not exist as far as Ego vehicle is concerned.

2.2 Function requirements and desired attributes

Following requirements are needed to be fulfilled through the approach proposed in work.

2.2.1 Function Requirements

1. Collisions occurring at all the sides of ego vehicle are avoided during entire motion
2. Ego vehicle always eventually reaches the destination
3. Velocity is planned to reach the destination at earliest possible time by allowing ego vehicle to travel as fast as possible
4. Non-negative velocity (Motion in forward direction)
5. Real time re-planning in order to incorporate continuously changing traffic scenario
6. Ego vehicle always keeps a safe distance from the moving vehicles

Characteristics and attributes of this velocity planning approach are given below. It is important for any approach to have possibilities of expansion and modification in the future to become very strong candidate in the long development process.

2.2.2 Desired characteristics of the velocity planning approach

1. Computations in the process should be realisable on actual hardware platforms
2. Ego vehicle avoids collision with all the vehicles present on the street with the planned velocity
3. Ego vehicle identifies the possible opportunity to enter the traffic junction
4. The approach should be general so that it can be adapted to work with different traffic scenarios and road geometries
   - Merging
   - Yielding
   - Intersection
   - Avoiding collision with obstacles of different sizes
   - Road geometries such as traffic junction, traffic circle, lane entrance, etc.
5. Scope of extension of the same approach to the more complex traffic scenarios and road geometries

2.2.3 Performance parameters

This thesis is a concept development of a velocity planning approach for collision avoidance. Simulation results in the chapter 4 provide a proof of concept of the fact that this approach is able to fulfill requirements mentioned above along with illustration of various performance characteristics. Therefore analysis of the values of performance parameters such as safety distance from obstacles and travel time is out of the scope at this stage of development.

2.3 Available information

Following information is assumed to be known for velocity planning of the vehicle.

- Path of Ego vehicle (Bus): Pre-calculated using on-line path planning or a fixed path for the entire motion
- State of Ego vehicle (Bus)
    1. Speed
    2. Position
    3. Heading direction
    4. yaw rate
- States of all other vehicles present in the environment
    1. Speed
    2. Position
    3. Heading direction
    4. yaw rate

2.4 Introduction to basic terms

This section will go through basic terms that are used in this work.

2.4.1 Ego Vehicle

Vehicle under investigation and the one that is being controlled is called as an 'Ego Vehicle'. A Scania Omni bus is the only Ego vehicle in this project for which the velocity is to be planned for collision avoidance. Therefore, this is also the only vehicle on the street for which entire path is known (or pre-calculated before velocity planning). The ego vehicle is assumed to be equipped with radar and other sensors that provide information required for velocity planning. Relevant physical parameters of the bus are stated in the table A.1 in the appendix.

2.4.2 Environment

Environment around ego vehicle include road, footpath, lanes, road dividers, traffic lights, fellow vehicles, pedestrians, cyclists, buildings, road markings, etc. Motion planning depends on the environment to a great extent as trajectories of the vehicles are constrained by the objects and the attributes of the environment. Figure 2.1 shows a typical scenario in the traffic junction environment. In the scope of this thesis, moving obstacles in the environment include cars. The approach proposed here can be extended for other moving obstacles such as pedestrians, cyclists and stationary objects, etc.

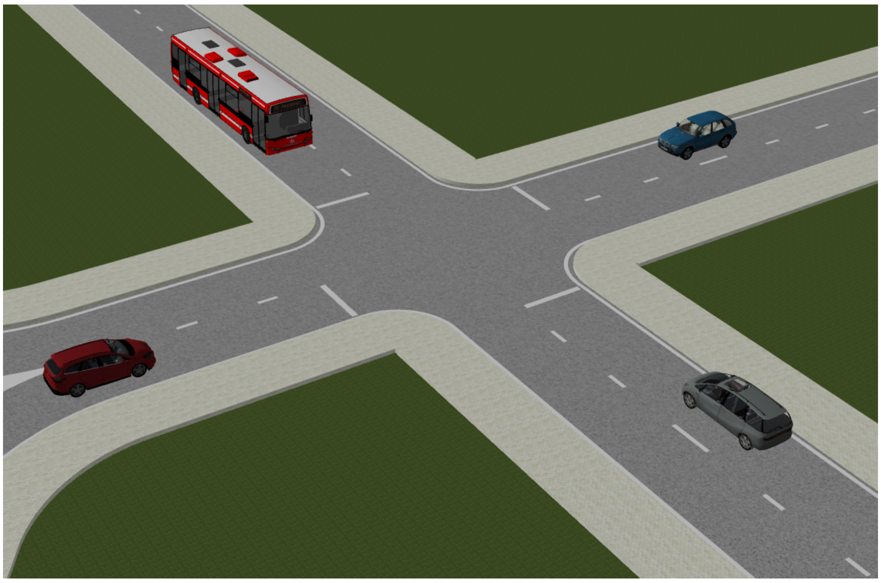


Figure 2.1: Environment: Traffic junction in *PreScan*

2.4.3 Reference Frames

In this work, approach is built in the moving frame of reference of the ego vehicle (s, t). In this reference frame, s (space) denotes the distance travelled along the path and t denotes the time. Co-ordinate transformation is needed to represent data from one frame of reference in to other.

2.4.4 Guide point

In 4 wheel vehicles like a car, bus or truck (with 2 axles), rotation of the vehicle happens around the center of the rear axle (figure 2.2). If this point is set as the guide point, then the motion of this point along a smooth curve ensures smooth lateral acceleration and steering action. Curvature of the path (or a road) changes linearly along its length. Locus of a points traced by the rotation center of a moving vehicle has a linear change in curvature along its length. Therefore, rotation center is the best choice to set up as a guide point as it will ensure smooth change in curvature and corresponding steering action [4]. Setting up the rear axle centre as a guide point can reduce a lot of computation and analysis of vehicle dynamics in order to avoid any discontinuity in the critical parameters such as lateral acceleration, steering angles and yaw rate.

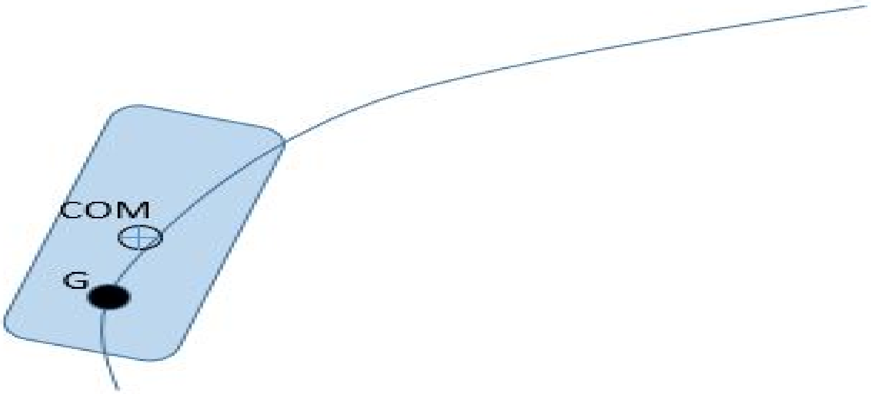


Figure 2.2: Guide point is center of rear axle instead of center of mass or center of geometry for a 4 wheel vehicle

2.4.5 Path follower (Speed Controller)

Path follower is a controller that generates the control inputs such as acceleration, steering angle, deceleration in order to achieve the desired path with desired speed. In this project, an in-built path following controller (from modelling software *PreScan*) is used for execution of planned motion which is acting as a separate entity from the motion planner. This controller is referred as a 'path follower' in *PreScan* GUI. A schematic diagram of the path follower is given below in the figure 2.3. It takes in the desired speed and current speed of the ego vehicle as inputs along with the path to be followed and generates control signal for actuators. Desired speed input in the figure 2.3 is generated by 'Velocity planning function' which is developed in this thesis.

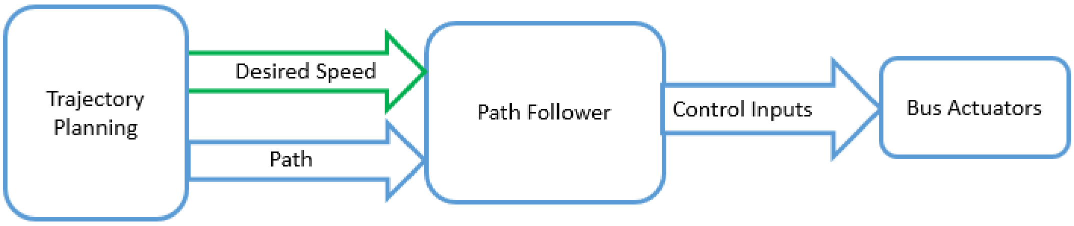


Figure 2.3: Path following controller: Input and output signals

2.5 System Overview

This section provides an abstract overview of the system architecture of the entire autonomous driving system with its key component blocks and flow of signals.

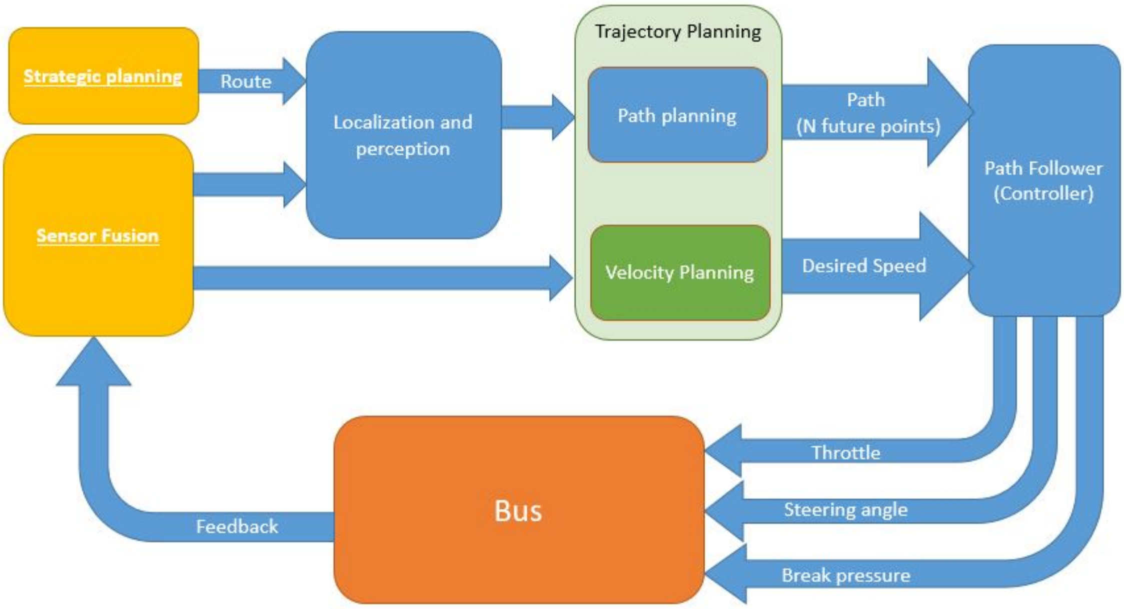


Figure 2.4: Overview of the autonomous driving system with connections between sub-blocks

Trajectory planning requires inputs from localization, sensor fusion and strategic planning to plan the trajectory (path and speed). This calculated path and speed are inputs to the controller that generates the control inputs for the actuators of the ego vehicle. In this work, velocity planning is the focus of this work. As explained earlier in section 2.3 limited amount of information is assumed to be known for velocity planning. Hence, no additional information is assumed to be known in terms of localisation, perception and strategic planning.

2.6 Related theory

In this section, basic theory behind the velocity planning approach is presented along with the concepts that are applied for developing this approach.

2.6.1 Space time analysis

Space time analysis uses the representation of the motion of an object along its path (space) as a function of time (t) i.e. $s(t)$. So, it is a two dimensional plot (or map) that can represent where ego vehicle is allowed or forbidden to be present at a given time. This kind of representation helps in visualising complex motion of multidimensional objects in just two dimensions of distance and time.

2.6.1.1 Space time plot for a point moving along a path

Space time approach for a point moving along a fixed path is presented in [5]. Following figure 2.5 shows a point moving along a fixed path and a rectangular object that intersects this point during time interval $[t_1,\ t_2]$. $[s_1,\ s_2]$ is the path segment where intersection is taking place. It is important to note that the space coordinate i.e. 's' is nothing but the distance along the path of a point. Hence, s starts from zero at the beginning of the path and increases monotonically along the length of path.

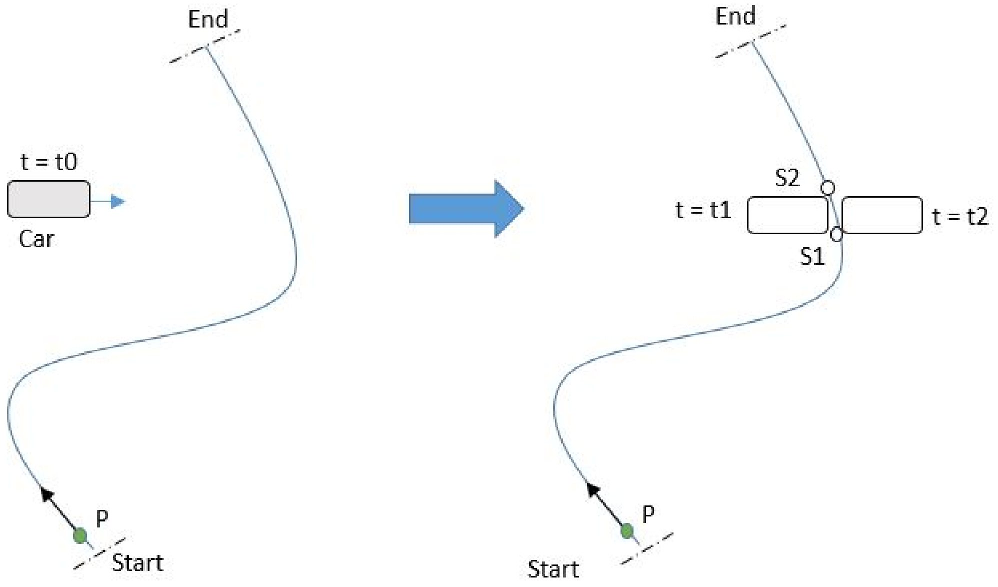


Figure 2.5: Intersection of the path of a point and path of a rectangular obstacle during time interval $[t_1,\ t_2]$ and space interval $[s_1,\ s_2]$

Following space-time map (plot) in the figure 2.6 represents above information of intersection in a graphical way.

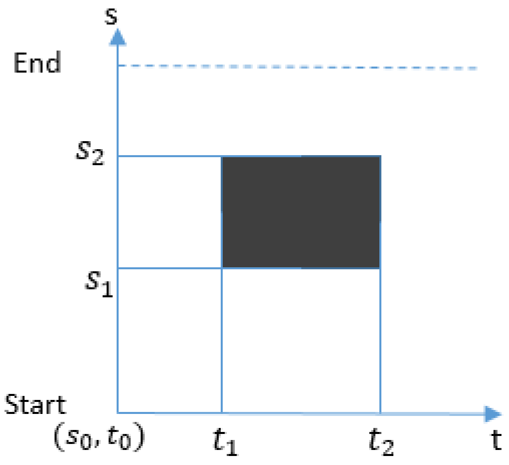


Figure 2.6: Shaded region representing the collision in space-time map for a point moving along a fixed path

In figure 2.6, the shaded region represents a collision in space and time. Since our goal is to avoid the collision, the space time curve for the desired motion of a point must not pass through this shaded region. Figure 2.7 shows two possible profiles that can lead to a collision free motion without putting a brake along the path. Theoretically, it is possible to have infinite number of curves in the space-time plot that do not intersect with the shaded region.

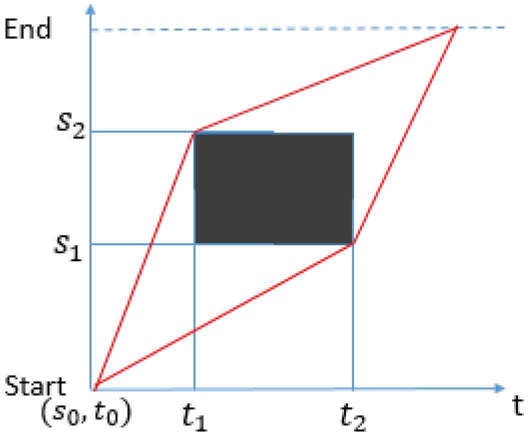
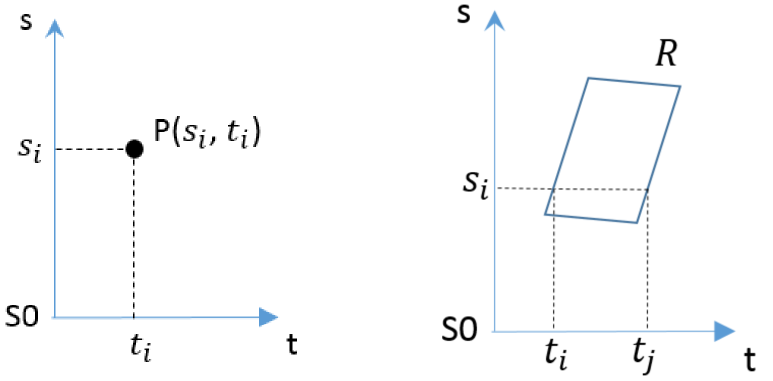


Figure 2.7: Red color curve shows the path travelled by ego vehicle in the space time domain in order to avoid the collision represented by shaded region

Interpretation of collision region in the space time map: Figure 2.8 shows two space time plots with a collision point and a collision region respectively. Any collision point $P(s_i, t_i)$ (figure 2.8a) on the space time map for the ego object moving along a desired path implies that there will be a collision with a moving obstacle if the ego object is present at position s_i at time t_i.

Similarly, a closed (bounded) collision region R (figure 2.8b) in the space time map for the ego object moving along a desired path implies that during a time interval (t_i, t_j) there will be collision with a moving obstacle if ego object is present at position s_i along its path.



(a) Collision point on the space-time plot

(b) Collision region on space-time plot

Figure 2.8: Interpretation of space-time plot

Understanding this interpretation is very important in order to extend this concept of space time scheduling for 2D objects. This interpretation will be useful in designing collision check method to generate space-time map.

2.6.1.2 Space time map for a 2D object moving along a path

It should be possible to find out the space time map for any 2D object moving along any path and with interaction with moving obstacles. However, because of the sizes and geometries of the objects, collisions can happen even when the obstacle is not completely crossing the path of the ego vehicle. Hence, it is important to identify the types of collisions that can happen when a rectangular objects are moving on a plane along a their respective paths.

Figure 2.9 shows three types of potential collisions of a rectangular object moving along a path. In the figure 2.9, Car 1 has a path that is intersecting with the bus. Car 2 has the path that is merging with the bus and hence leading to the collision in some future instance. However, Car 3 does not have any intersection with the path of the bus, but because of the path of the bus and its long size, part of the bus will intersect with the part of the car at some future instance.

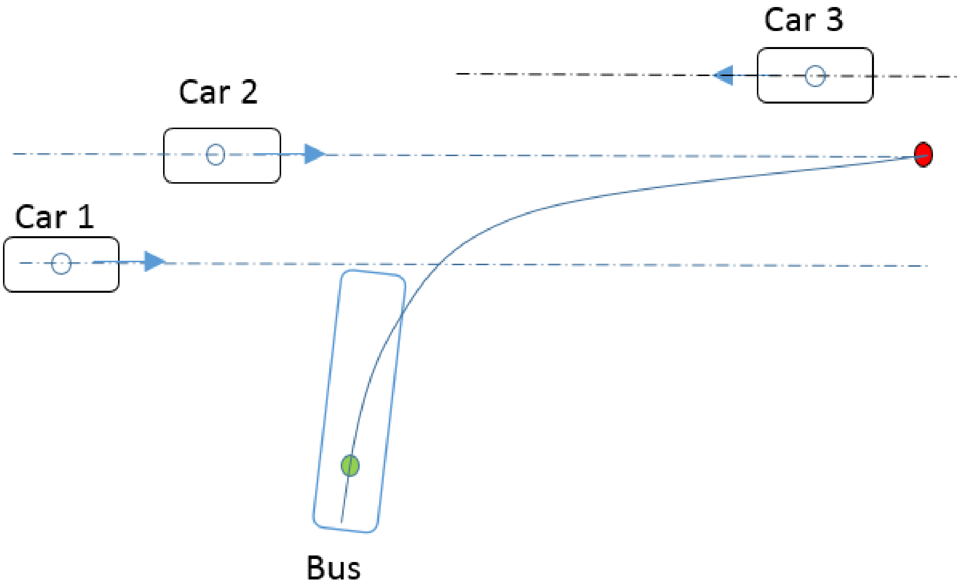


Figure 2.9: Different types of collisions between two moving vehicles

Based on the interpretations that we develop in the section 2.6.1.1, it is possible to create a space time map using collision checking. Collision checking is effective in detecting collisions for all 3 cases discussed above. This method will be discussed further in the section 3.3 in the next chapter.

2.6.1.3 Space time map characteristics and constraints

In order to generate a feasible velocity profile in space-time map, it is important to consider following constraints that come naturally with the formulation of this problem. So, any curve defining motion of the object in space-time map must obey following constraints in order to have a realistic motion ($s(t)$).

- Monotonicity of time
- Non-negative velocity
- Maximum velocity limits along the path points
- Maximum acceleration constraint

Maximum allowable speed on the path is dependent on the curvature of the path in order to avoid skidding and rolling over. Furthermore, there is a traffic speed limit on the road. These speed limits should be considered while planning speed of the vehicle. In the current work, speed limit is kept constant for the entire motion of the ego vehicle (bus). Non-negative speed is considered to avoid reverse direction motion in the scope of velocity planning problem. Maximum acceleration constraint and the variable speed limits along path are not considered in current work. However, the current approach can be modified to in the future work to include these constraints.

2.6.2 Motion model

Motion of surrounding vehicles is predicted using coordinated turn (CT) model. In CT model, the rate of change of heading angle $\omega(t)$ is constant, i.e. $\dot{\omega}$ is zero. A discrete time state space expression of the CT model is given in equation 2.1.

In order to improve the motion prediction, an alternative motion model would be to use perceptive elements into the motion prediction, such as lane information and indicators of the vehicle predicting the future part of the vehicle (Discussion in section 5.2.4). Non-zero linear and angular acceleration in the motion model can also improve motion prediction of the vehicles.

$$\underbrace{\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{v}(t) \\ \dot{\phi}(t) \\ \dot{\omega}(t) \end{bmatrix}}_{\dot{X}(\mathrm{t})} = \underbrace{\begin{bmatrix} v(t)cos(\phi(t)) \\ v(t)sin(\phi(t)) \\ 0 \\ \omega(t) \\ 0 \end{bmatrix}}_{\mathrm{A(X(t))}} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} q_c^v(t) \\ q_c^\omega(t) \end{bmatrix}}_{q_c(\mathrm{t})} \tag{2.1}$$

Where, $(x,\ y)$ is the position of the vehicle and v is the velocity. ϕ is heading angle and its rate of change is ω. Here, q_c^v and q_c^ω are the noise in the linear and angular speed terms. Noise is not considered in the motion model used in this work at this stage of development. After the successful proof of concept, it should be possible to introduce uncertainty in the motion prediction model in later stages of development as discussed in the section 5.2.4. Hence, this work is based on deterministic motion model of vehicles. Noise terms will vanish from the expression ($q_c(t) = [0\ 0]^T$) and pure deterministic model will remain. This continuous time model is converted to discrete time model using Euler's method (refer equation 2.2) which can be directly used for the prediction of vehicle's position in the future instances.

$$X(k+1) = \Delta T A(X(k)) + X(k) \tag{2.2}$$

ΔT is a regular time interval at which future position is predicted. Section 3.3.2 contains detailed discussion about selection of ΔT.

2.6.3 Collision detection

Top view of a vehicle can be considered as a rectangle. There are various ways to detect collisions between the rectangles. In this work, 'Separating Axis Theorem' is applied for checking collision between two rectangles. This theorem is a special case of the 'Hyper plane separation theorem' [19, chapter 2, p. 46]. This theorem is valid for the collision check of two convex polygons.

Separating axis theorem for rectangles: If there exists a line separating two rectangles, then the rectangles are not intersecting. Also, for two non-intersecting rectangles, there exists a line parallel to at least one of the edges of any one of the rectangles that separates two rectangles from each other.

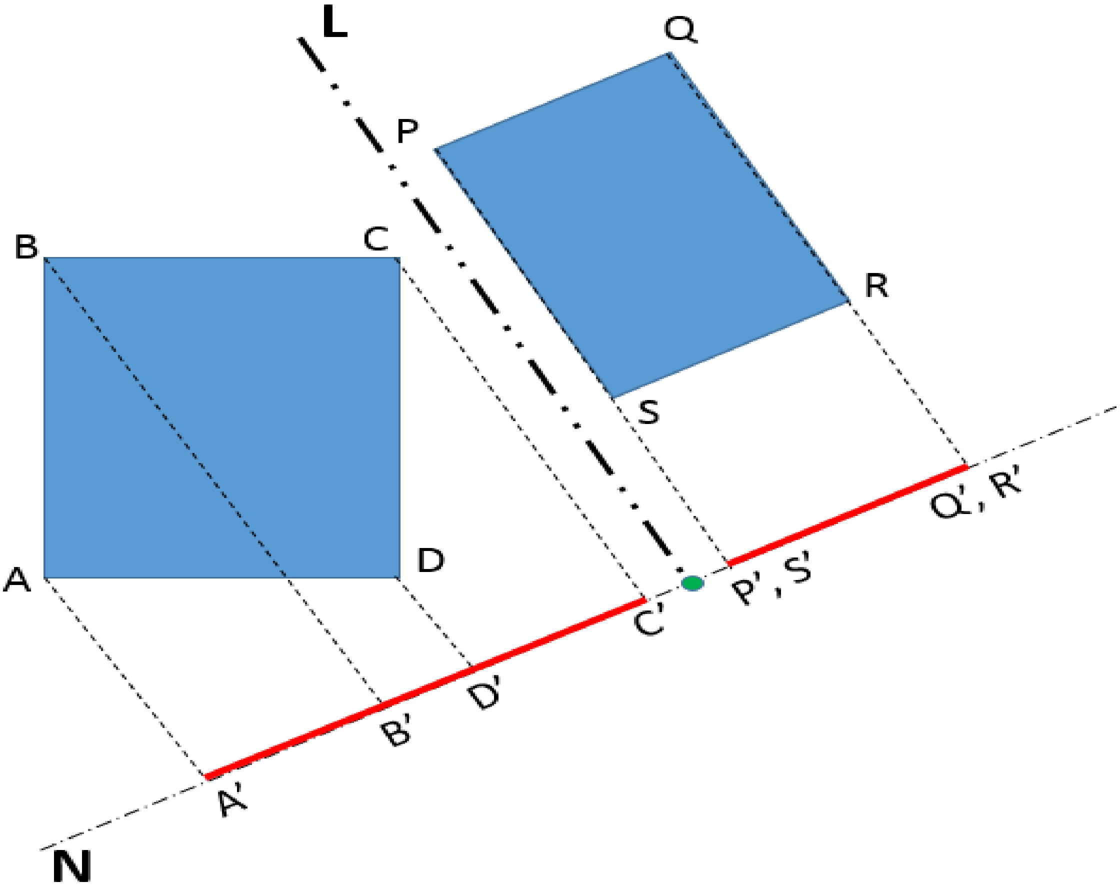


Figure 2.10: Illustration: Separating axis theorem

Refer to the figure 2.10 where two rectangles are separated a by line L which is parallel to the edge PS of the rectangle $PQRS$. Line N is normal to the line L. Since line L is separating two rectangles, their projections on line N will also be separated. Projection of each rectangle on line N is a line segment ($P'R'$ and $A'C'$). Since the rectangles are separated, the segments are also separated i.e. non overlapping. This way, collision detection can be implemented for any position and orientation of two vehicle footprints. By checking for any two adjacent sides of each rectangle for non-overlapping projections, presence or absence of collision can be detected.

2.6.4 Homogeneous Coordinate Transformation

In order to generate the footprints of vehicles in the space, coordinates of each vertex of footprint need to be transformed according to translation and rotation of the vehicle. Homogeneous coordinate transformation is used for this purpose. Since the geometries involved in this work are rigid bodies, phenomena such as shearing, scaling are not considered here. Translation in XY plane and rotation around Z-axis are the only two phenomena that are to be considered during coordinate transformation in this work. Consider following example in the figure 2.11. Point P is located in the local frame at a fixed position.

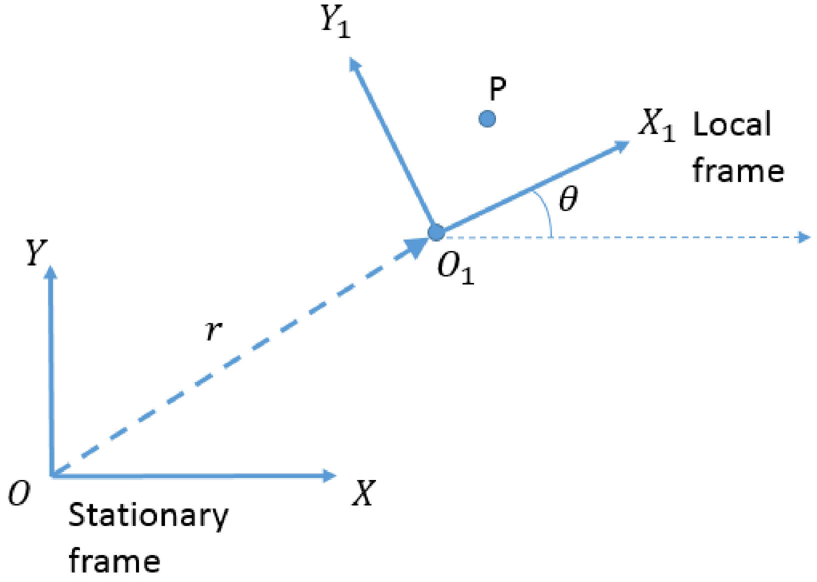


Figure 2.11: Transformation of coordinate system of a point P from local frame $X_1O_1Y_1$ to global frame XOY

Homogeneous transformation matrix for translation in XY plane and rotation in θ.

$$T = \begin{bmatrix} cos(\theta) & -sin(\theta) & r_x \\ sin(\theta) & cos(\theta) & r_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

As the local frame moves and rotates in the plane, position of the point P with respect to the global frame ($\overrightarrow{OP}$) is given by equation 2.4. Local frame is having translation $\vec{r}$ and rotation θ with respect to the stationary frame (global frame).

$$\overrightarrow{OP} = T \times \overrightarrow{O_1P} \tag{2.4}$$

2.6.5 A* search

Time optimal speed profile in the space-time map is generated using A* algorithm and it will be explained in details in the section 3.4.2. A* is a path search algorithm for finding the shortest path between two points (or nodes) in the grid space [6]. Hence, it can also be used to find the shortest obstacle free path between two points. A* is the best-first search meaning that search is directed along the best estimated direction.

In the first step of algorithm, nodes around the start node are explored in the search for the potential successor. Node with the minimum function value (f) is selected as a successor node. In the next step, nodes around the successor node are explored for its successor. This goes on until the target node is reached. Function value is calculated using equation 2.5. $f(n)$ for each node(n) is evaluated as the sum of cost to reach the node ($g(n)$) and heuristic ($h(n)$) for node n.

$$f(n) = g(n) + h(n) \tag{2.5}$$

$g(n)$ is the actual cost to reach current node 'n' from the start node. Heuristic $h(n)$ is an estimated cost to reach the target node from current node (n). Since it is an estimated cost to reach the end node, there is no specific value. One way to calculate heuristic is to calculate distance between current node (n) and target node. However, heuristic cost must be admissible in order to ensure the optimality (shortest path). This is explained further in the section 2.6.5.

During the search, algorithm maintains two lists: *Open list* and *Closed list*. Closed list contains the list of all the nodes that have already been explored and open list contains list of all the successors (of already explored nodes in closed list) that are yet to be explored. Nodes in the closed list are not explored. All the nodes that represent obstacle (or collision) are put into the closed list in the beginning of the search so that collision free shortest path is generated. During the search for the shortest path in the search space, the node with the minimum function value ($f(n)$) is selected from the open list for further exploration. This results in a 'best first' search. Open list ensures that all the possible paths are considered so that resulting path is the shortest path. As the search reaches the target node, the search stops by adding the last node into the closed list. By traversing back through the parents of the last node in the close list, optimal path between target node and start node is obtained.

Admissible heuristic: In order to ensure the optimal solution to A* search, it is important to have an admissible heuristic [20, chapter 4, p. 97]. The heuristic is admissible if the value of the heuristic is always less than the actual cost to reach the end node from the current node. Hence, the heuristic should not exceed minimum possible cost to reach the target node from current node. One admissible heuristic would be $h = 0$ which is basically Dijkstra's algorithm. A zero heuristic will result in more computation leading to a slower search operation. On the other hand, larger heuristic results in the faster search operation. However, overestimated heuristic does not ensure the optimality. Therefore it is beneficial to look for a finite positive admissible heuristic to ensure optimality along with high search speed with A*.

Branching factor: Branching factor in the context of A* search algorithm is the number of possible connections from a given node to the neighbouring nodes [20, chapter 3, p. 74]. Figure 2.12 shows examples from two search algorithms with different branching factors.

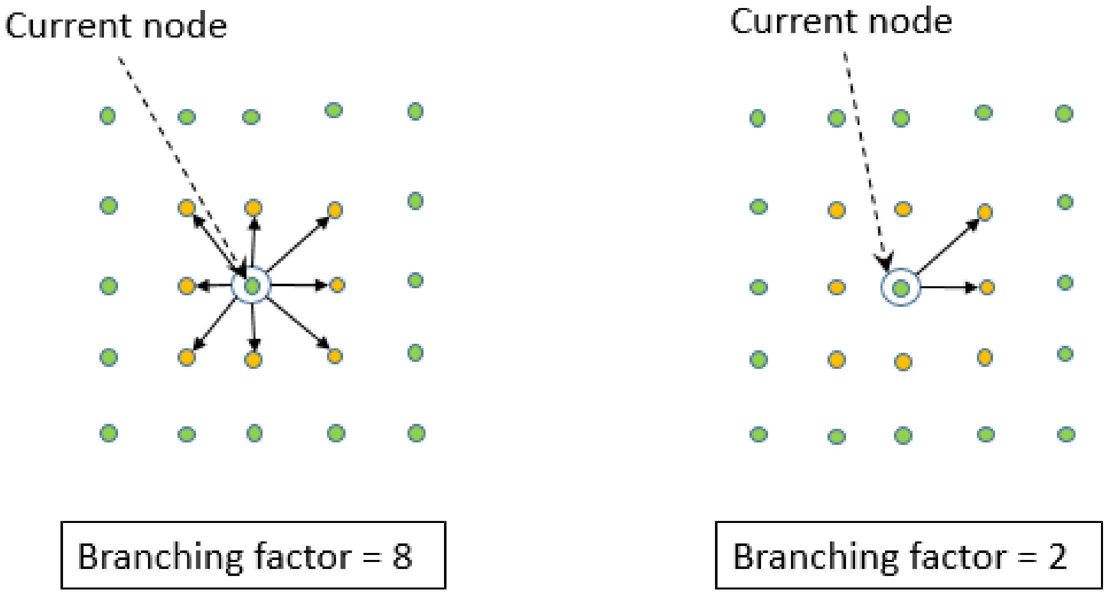


Figure 2.12: Branching factor in the grid search

If search is allowed only in the particular direction, then the branching factor decreases. This term will be relevant later in the section 3.4.2.

3

Approach

Velocity planning approach developed in this work is presented in this chapter. This approach is based on the concept of space-time analysis presented in [5]. As discussed in the earlier sections, velocity planning is one of the two sub-problems of the trajectory planning problem which include path planing and velocity planning. Entire velocity planning approach is broken down into two steps:

- Formation of space-time map using collision check
- Generating time optimal velocity profile using A* algorithm

Furthermore, on-line velocity planning implies re-planning after certain time interval. At every new planning instance, a new velocity plan is generated using the proposed method in this chapter. Frequency at which velocity is re-planned is referred as planning frequency and it will be discussed in the next chapter in section 4.1.2.

3.1 Assumptions

Following are the two important assumptions based on which this approach is developed.

- Bus follows the path with a good enough accuracy with the help of path following controller so that its actual position on the road is approximated with the corresponding position on the fixed path
- Road is flat

3.2 Discretization of Space-time map

This work is focused on discretized space-time map which is generated using the collision checking. Discretized space-time map (or plot) in the figure 3.1 represents collision with two vehicles. Spacing along the space coordinate is called space interval Δs and the spacing along the time co-ordinate is called as time interval (Δt). These values are selected in a specific way and will be discussed in the later sections. Total length of time in the future for which the motion of the fellow vehicles (moving obstacles) is predicted is called as 'time horizon'. Similarly, 'space horizon' is defined as the length of the ego path considered in the space time analysis at a given planning instance. Space horizon is nothing but the distance along the path for which velocity is planned. Figure 3.1 shows how space and time horizons define the size of the space-time map. In the subsequent sections, systematic method for generating space-time map will be presented.

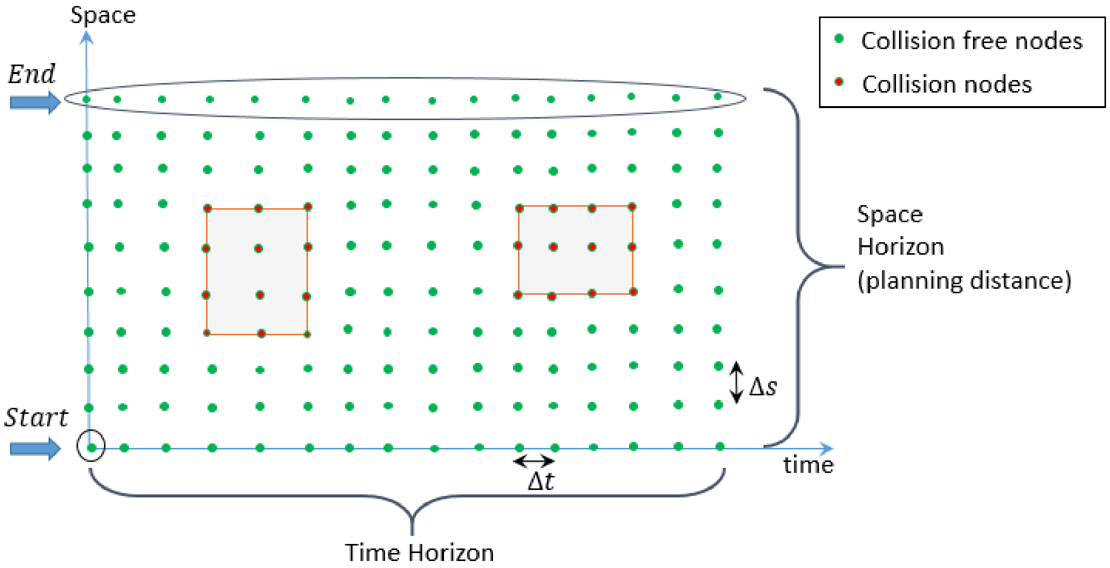


Figure 3.1: Discretized space-time map

3.3 Collision checking

Collision checking needs to be done in a specific fashion to get the required information in space-time domain. Since the main purpose of velocity planning is to control the position of the ego vehicle along its path, it is quite intuitive to calculate the footprints of the ego vehicle along its path. However, ego vehicle is forbidden to occupy certain path segments during specific time intervals for collision avoidance with other moving obstacles. These time intervals are based on the motion of other moving obstacles and therefore, vehicle footprints are calculated for current and future time instances using motion model.

In figure 3.2, paths of Car_1 and Car_2 are unknown and to be predicted using Coordinate turn model (CT model). Actual path for the bus is already known (predefined or preplanned path).

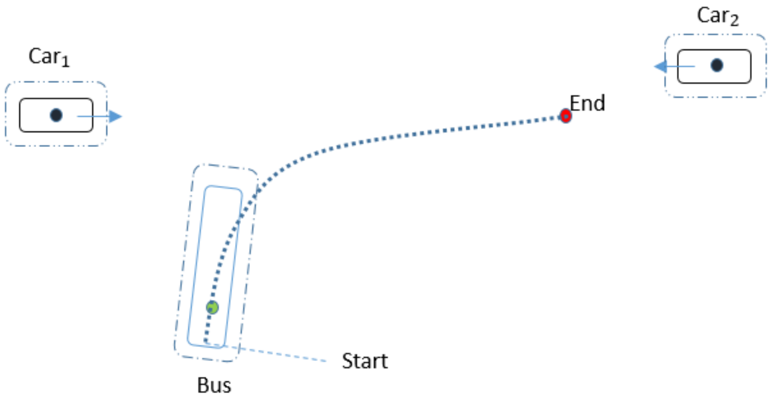


Figure 3.2: Collision checking

3.3.1 Inflation of the vehicle size

Ego footprint is generated at fixed space interval along its path. So, it is important to ensure that the intermediate positions of the ego vehicle do not indulge into any collision. This can be achieved by enlarging (inflating) the sizes of the vehicle footprints before collision checking. Collision checking between these inflated footprints will add a safety distance between the vehicles. However, the velocity planning as a result of such collision checking is more conservative. Following figure 3.3 shows how the area swept by the inflated vehicle footprints (green colour) at the fixed space interval on the path covers the area swept by the actual footprints (blue colour) of the vehicle along entire path.

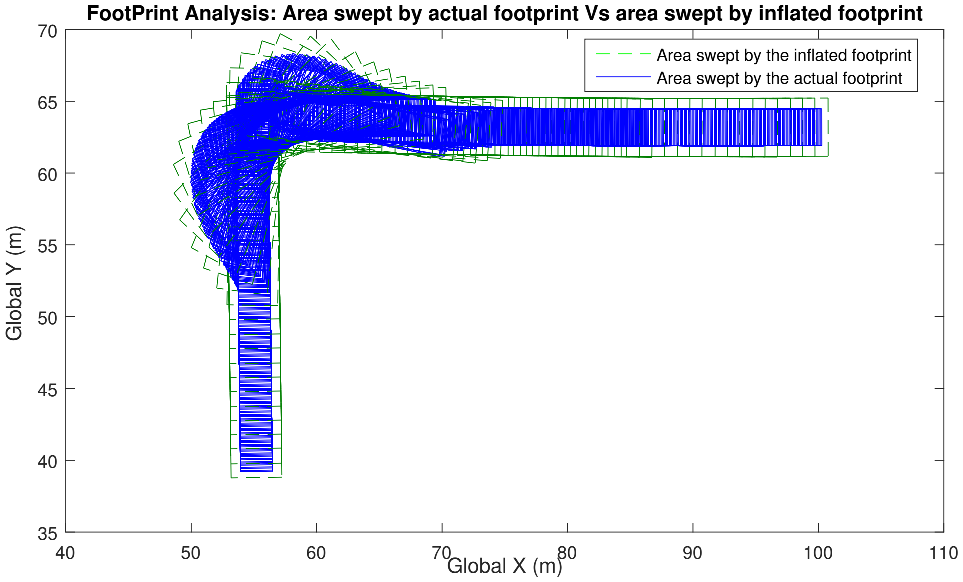


Figure 3.3: Inflated bus footprints ($Length = 16.5m$, $Width = 3.55m$) at 1m interval along the path covering actual bus footprints ($Length = 14m$, $Width = 2.55m$) at 0.1m interval along the path

How much inflation in size is required at a given space interval of Δs is subject to the investigation and out of the scope of this work. Table 4.1 in results chapter includes the inflated dimensions used in the simulation.

3.3.2 Vehicle footprints along time

Future position of each fellow vehicle is predicted using coordinated turn model at a fixed time interval Δt. Equations describing the motion model are presented in section 2.6.2. Based on the predicted positions of the vehicles, corresponding footprints are obtained with the help of homogeneous coordinate transformation explained in section 2.6.4. Figure 3.4 shows prediction of footprints of two vehicles (Car_1 and Car_2) along their predicted paths at fixed time instances (t_1, t_2, ... so on). These time instances form the time axis in the space time grid as shown in the figure. Yaw rate is zero for the first car implying straight line motion and non-zero for the second implying circular motion.

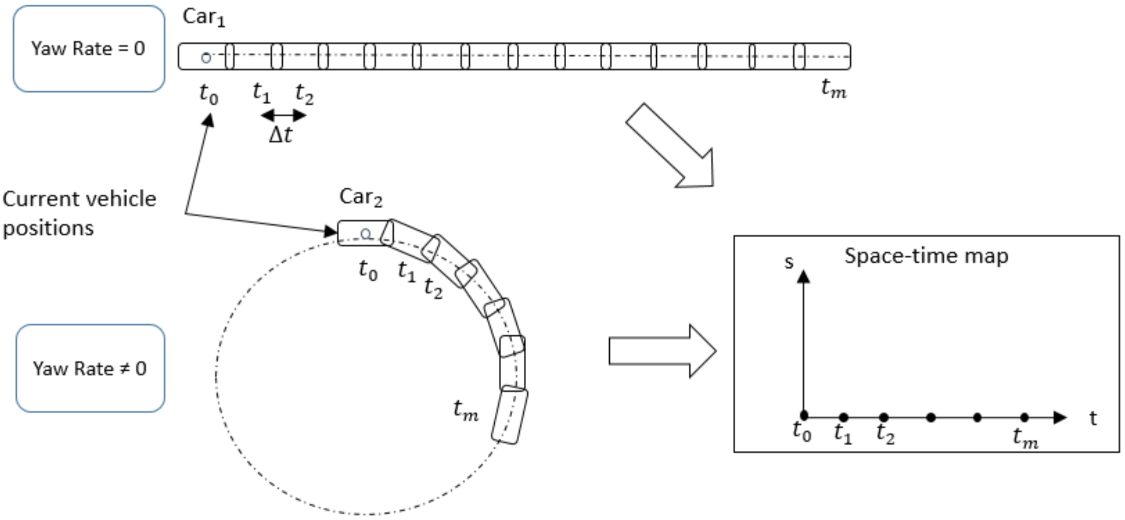


Figure 3.4: Prediction of vehicle footprints using CT model

Δt is chosen in a way such that the every two adjacent car footprints just overlap with each other leaving no space between them. If there is a gap between two adjacent footprints, collision happening in this gap cannot be captured by the collision checking. This time interval (Δt) can be calculated using the length and speed of the vehicle as shown in equation 3.1. Here, $\alpha \in [0, 1]$ is a factor that affects degree of overlapping of two consecutive footprints. $\alpha = 1$ implies footprints are just in contact at the edges while $\alpha = 0$ implies complete overlapping. Suppose a particular car has dimensions as: $(Length \times Width \times heigth) \equiv (L_v \times W_v \times h_v)$.

$$\Delta t_{car} = \frac{Length_{car}}{Velocity_{car}} \times \alpha \tag{3.1}$$

Different sizes and speeds of the vehicles (cars) will result in different time intervals. However, the outcome of collision check against all the vehicles is mapped on the same space-time map. This is achieved by storing the information of collision in a single large Boolean matrix called 'Collision Matrix' (refer to the section 3.3.4). Therefore common time interval (Δt) is chosen for collision check using equation 3.2.

$$\Delta t \leq min\{\Delta t_1, \Delta t_2, \Delta t_3, ..., \Delta t_K\} \tag{3.2}$$

In this work, a time interval of $\Delta t = 0.5sec$ is considered which is fixed throughout the simulation. $\Delta t = 0.5sec$ obeys above inequality in the equation 3.1 for given vehicle speeds in the simulations.

3.3.3 Ego vehicle footprints along the space

Footprints of the ego vehicle lie along its known (or pre-calculated) path. Figure 3.5 shows illustration of generating ego vehicle footprints at regular space intervals along its path. In order to use these footprints for the collision checking, the ego vehicle (bus) should follow its desired path with a very good accuracy (assumption in section 3.1). Distance between between two consecutive ego footprints is referred as 'space interval' (Δs). Δs is also the spacing along space axis in discretized space-time map. Figure 3.5 shows how discrete positions along the path of the ego vehicle corresponds to discrete space positions in the space-time map. Space interval (Δs), time interval (Δt) and maximum ego speed (V_{max}) are related parameters and their values for the simulation will be calculated in the section 4.1.2.

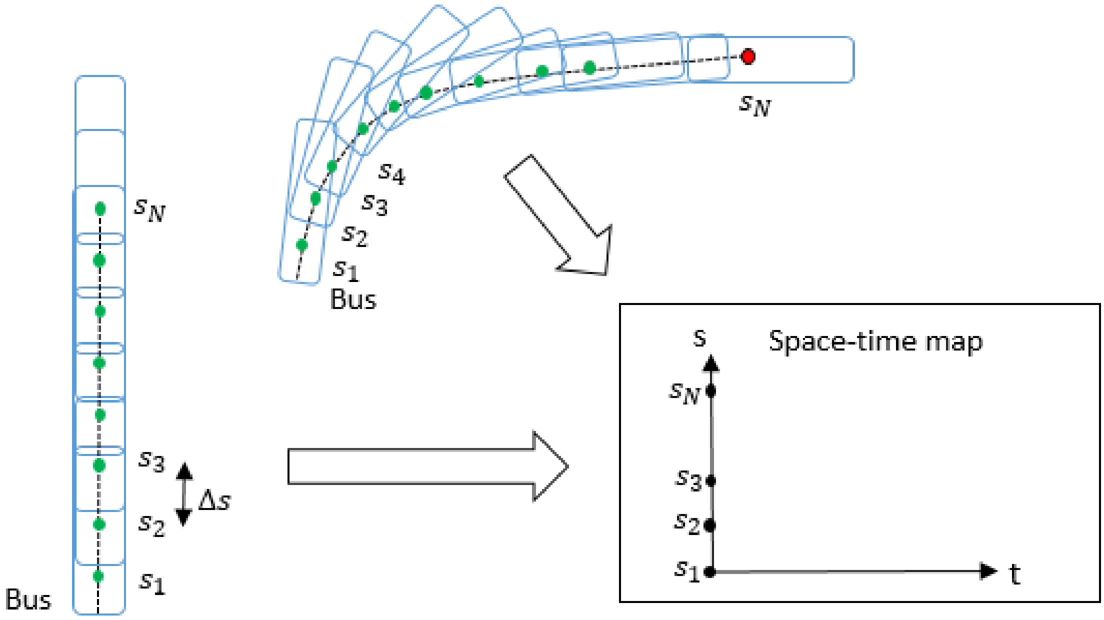


Figure 3.5: Ego vehicle footprints along space

Figure 3.6 shows footprints of ego and other vehicles as discussed above. Generating space-time map would require collision checking between all the footprints of ego vehicle against all the footprints of other vehicles on the street.

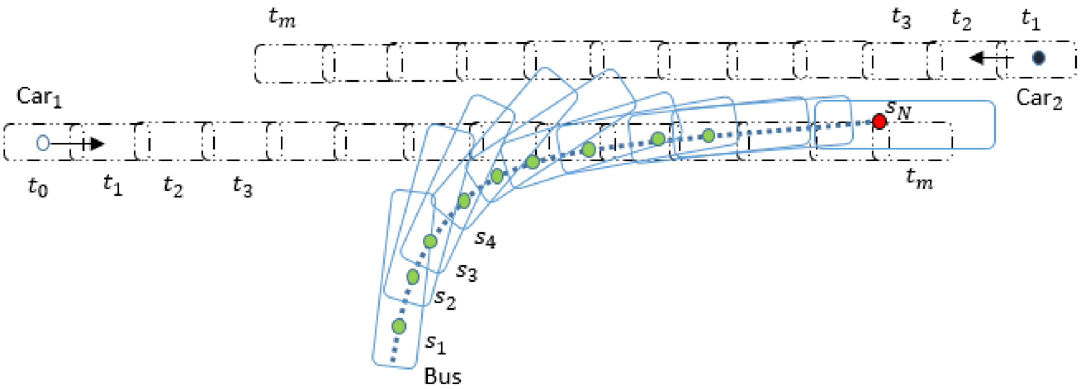


Figure 3.6: Collision Checking: Car footprints Vs Bus footprints

Suppose that we have M vehicle footprints along future time instances and N ego footprints along its path. Figure 3.7 shows a footprint of the Car_1 predicted at the future time instance $t = t_m$.

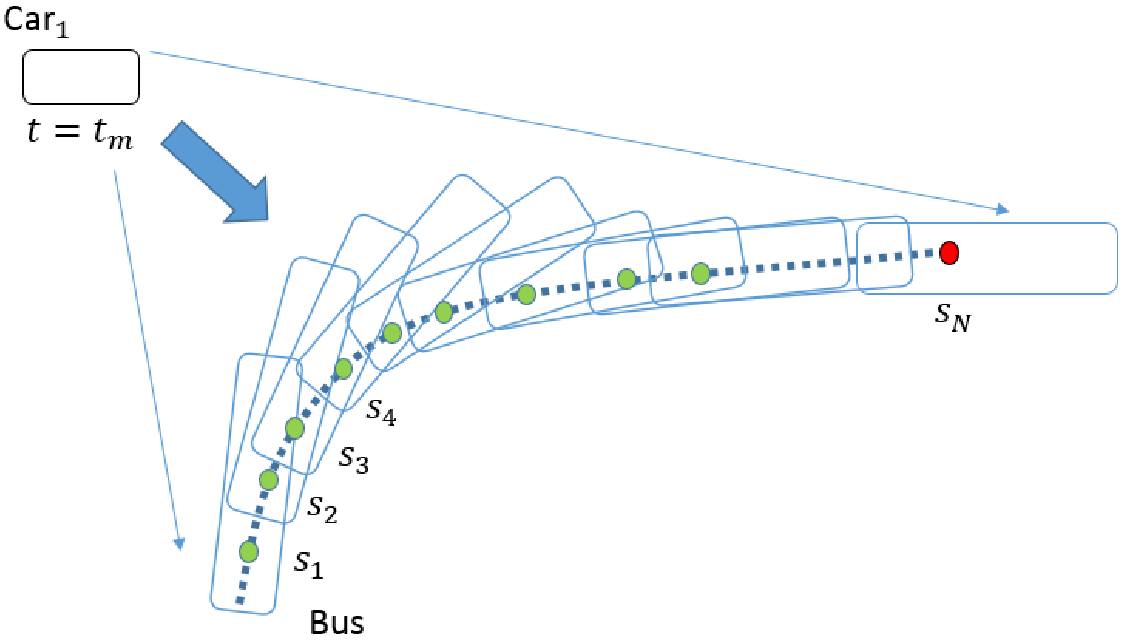


Figure 3.7: Collision Check: $M \times N$ computations

This footprint of Car_1 would have to be checked against all the bus footprints from s_1 to s_N in figure 3.7. For a single vehicle (car), collision checking needs to be done $M \times N$ times. Hence, this step would be highly demanding in terms of computations when algorithm of collision detection stated in section 2.6.3 is used. Furthermore, it would require powerful computers for collision checking with multiple vehicles on the street. Some kind of reduction must be done in order to make this collision check realistic. Section 3.3.6 will focus on reduction in the computations in the collision checking.

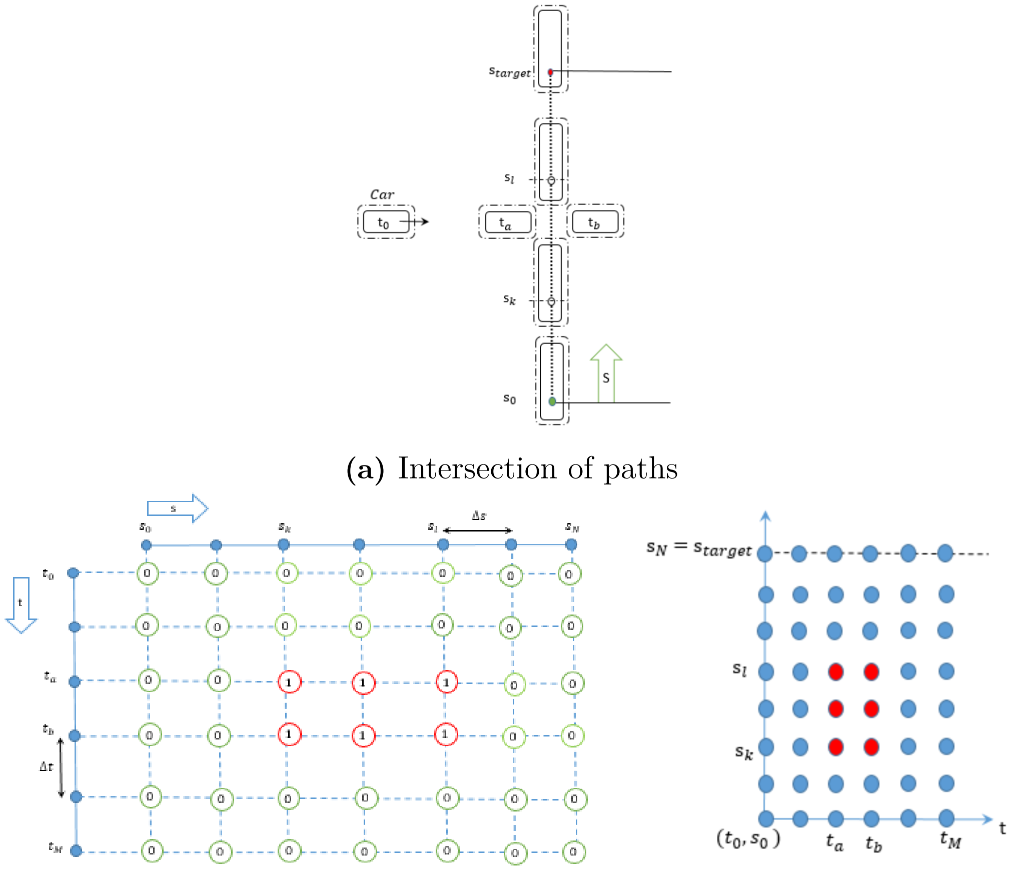
3.3.4 Collision Matrix

Collision checking between ego vehicle and other vehicles can be represented and stored systematically using a single 2D matrix. We will call this matrix the *collision matrix*. In collision matrix, index along rows represent the discretized time (t_0,t_1...t_m) and columns represent the discretized space (s_0,s_1...s_m). For a typical intersection scenario in the figure 3.8a, collision check will result in a 2D Boolean matrix as illustrated in the figure 3.8b. The matrix can be initialised to 0 (false) and then collision can be represented using 1 (true) upon 'detection'. This kind of representation is very useful in saving data of collision checking against all the vehicles in a single matrix. Location of any element in this matrix represents a point in the space-time map $((i, j) \Rightarrow (t_i = i \times \Delta t, s_j = j \times \Delta s))$, where Δs and Δt are the intervals in space and time that are used for the collision checking. This way collision matrix represents entire space-time map along with the collision information.

In order to store collision checking of all the vehicles in a single matrix, it is important to have a same time interval for collision check against all the vehicles. This is important because any element of final collision matrix should correspond to same absolute time for all the vehicles. However, using different Δt for different vehicles based on their speeds (equation 3.1) is computationally cheaper. If two different time intervals $(\Delta t_1, \Delta t_2)$ are used for collision checking against two different vehicles, then single element (i, j) in the final matrix will correspond to different absolute time for different vehicles ($t = \Delta t_1 \times i$ and $t = \Delta t_2 \times i$). Hence, remapping must be done in such case to represent all the collisions in the same space-time map with common time interval (Δt). In this work, time interval (Δt) is kept same for all the vehicles.

3.3.5 Space-time map using collision check

Collision matrix is used to plot the space-time map. There are three different types of collisions that are discussed in figure 2.9. Figure 3.8c shows space-time graph for the scenario in section 3.8a using collision matrix in 3.8b. Same concept of collision matrix is applied for scenarios in figures 3.9 and 3.10 to generate their respective space-time maps.



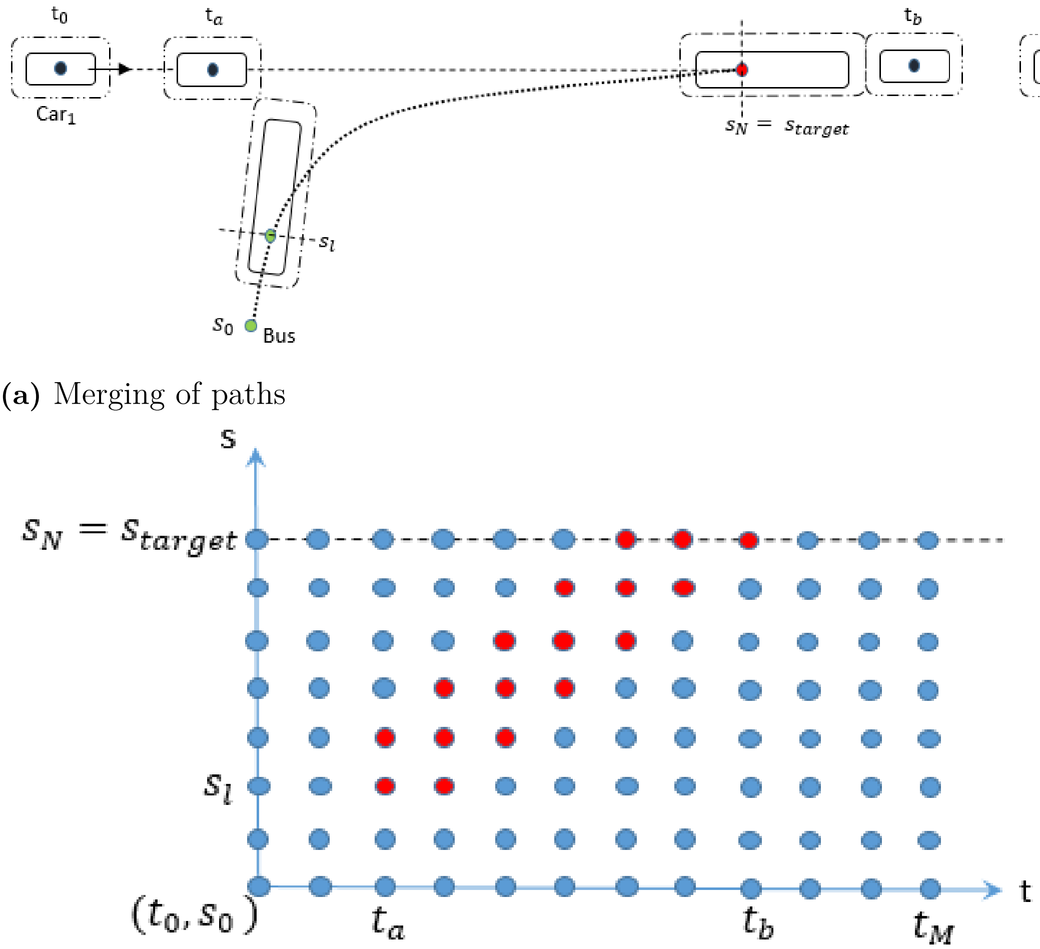
(a) Intersection of paths

	s_0		s_k		s_l		s_N
t_0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
t_a	0	0	1	1	1	0	0
t_b	0	0	1	1	1	0	0
	0	0	0	0	0	0	0
t_M	0	0	0	0	0	0	0

(b) Collision Matrix: ‘1’ represents collision (c) Space-time map

Figure 3.8: Space-time analysis in case of intersecting paths

In figure 3.9, t_a is the *earliest* future time instance of Car_1 when collision is detected and s_l is the corresponding position of the ego vehicle (bus) along its path. Similarly, t_b is the last time instant in the future where collision detected and $s_N = s_{target}$ is the last ego position corresponding to it. Collisions in the figure 3.9b represents merging of the ego vehicle and Car_1 into the same lane.

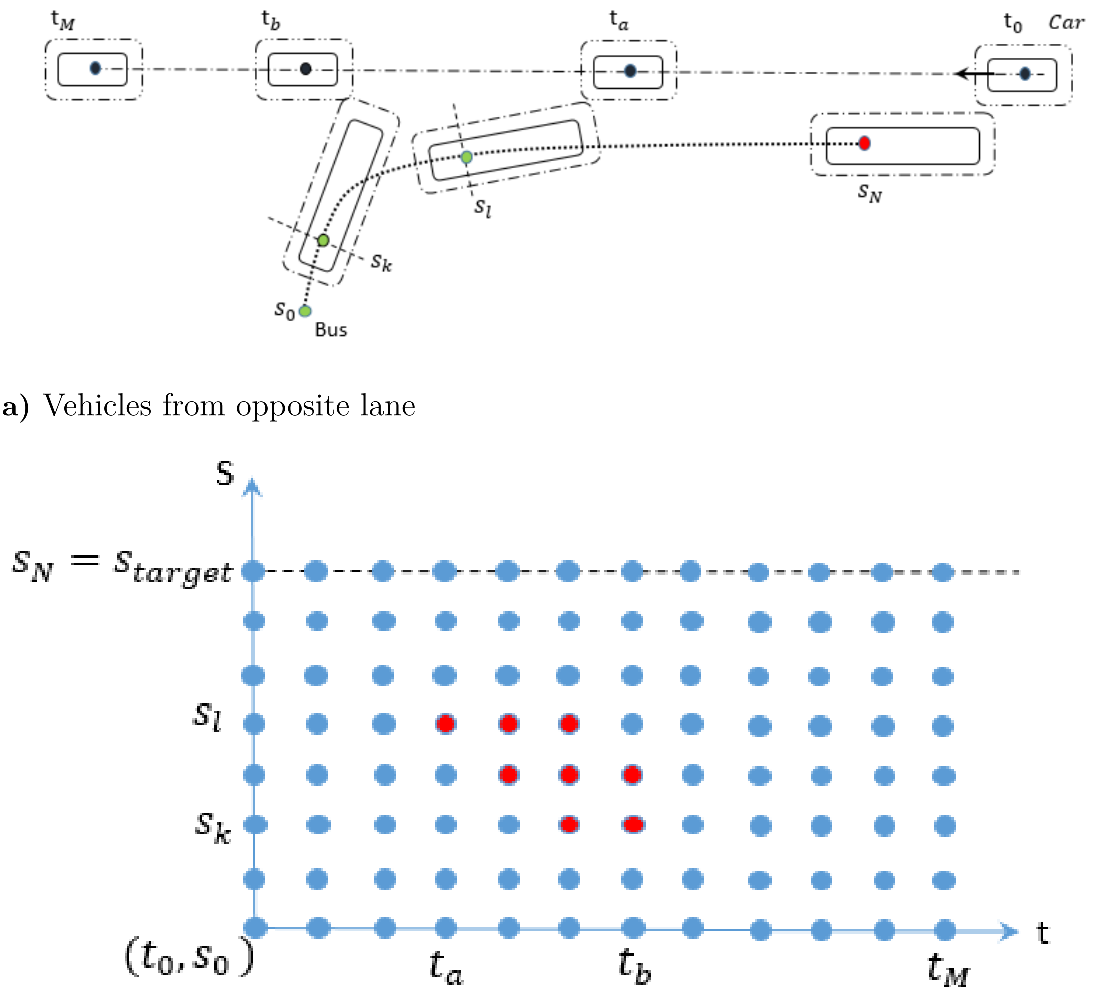


(a) Merging of paths

(b) Space-time map

Figure 3.9: Space-time map in case of merging

In figure 3.10, ego vehicle is going to enter opposite lane while making right turn. Hence, this results in a collision with a vehicle coming from opposite side.



(a) Vehicles from opposite lane

(b) Space-time map

Figure 3.10: Space-time map showing collision of ego vehicle with a vehicle travelling on opposite lane

3.3.6 Collision checking: Scope of reduction in computation

If collision detection between each pair of footprints is achieved using method presented in section 2.6.3, the process of collision checking becomes computationally expensive. Order of computation of 'collision detection' becomes N^2. Hence, it is essential to reduce computations in this process to achieve collision detection against many vehicle footprints.

This can be achieved at several stages in implementation. Perception about the motion of the vehicles in the localised environment can result in the great level of reduction in collision checking computations. For example, information of the lanes of the other vehicles can be used to avoid collision checking against the vehicles that lie in the irrelevant lanes. In this project, use of extensive collision detection method is avoided using three schemes.

1. Using a tree structure to avoid use of extensive collision detection method

2. Using a threshold on the distance between two footprints to avoid use of extensive collision detection method
3. Using the heading direction of a vehicle's footprint to avoid use of extensive collision detection method

3.3.6.1 Use of a Tree structure

Ego vehicle footprints are to be checked for collisions against the footprints of all the vehicles. Space bounded by the ego vehicle's footprints can be divided as shown in the figure 3.11. So, footprints of the vehicles that lie entirely outside this bounded area do not obviously collide with any of the ego footprint and hence do not require extensive collision detection. Footprint of a car at $t = t_m$ lies in a top-left region in the divided space in figure 3.11. Ego footprints that completely lie inside or overlap with the top-left region can possibly collide with this car footprint. All the other ego footprints that lie completely outside this top-left region obviously do not collide with a given car footprint. Hence, a set of potential ego footprints can be generated for extensive collision detection. This concept can be implemented by storing ego footprints in a data structures like Kd tree or quad-tree using the position of the ego footprint in the divided space. A simple tree presented in the figure 3.12 is implemented in this work to improve computational speed.

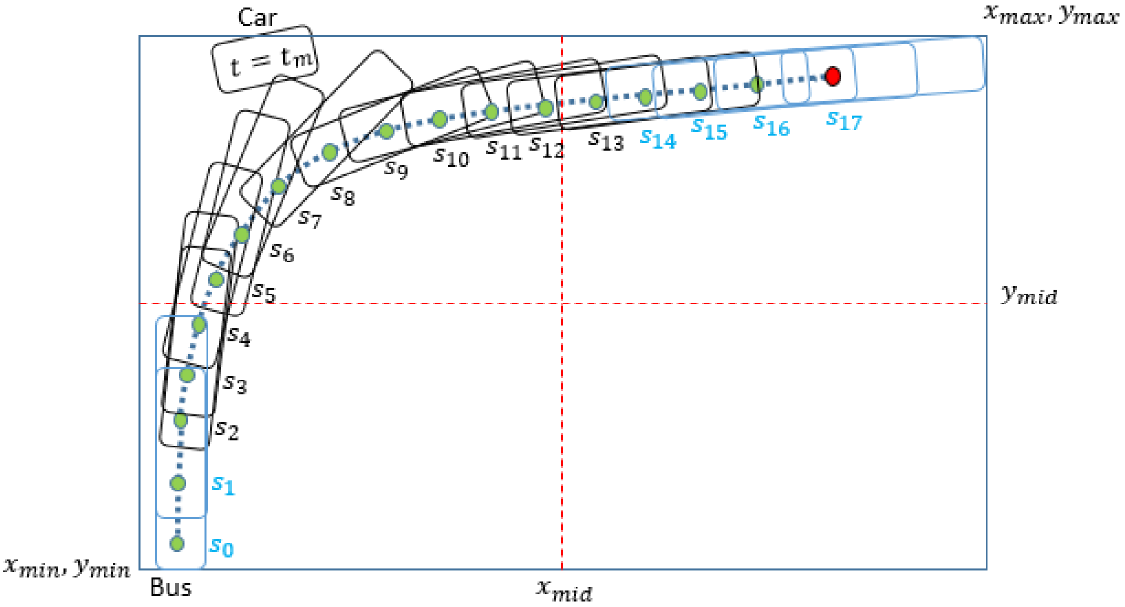


Figure 3.11: Bounding area of ego footprints along with the division of space within it for efficient collision checking. Ego footprints in the black colour lie in the same quadrant as the given vehicle footprint

Each vehicle footprint needs to be checked against all the ego footprints (s_0 to s_{17} in table 3.1) for collision. Goal of this method is to find the set of ego footprints that can potentially collide with a given vehicle footprint. In the figure 3.11, bounding region of area swept by the ego vehicle along its path is represented using (x_{min}, y_{min}) and (x_{max}, y_{max}). This region is divided along x_{mid} and y_{mid}. Based on the position of the ego footprint in the divided space, sets of ego footprints are formed. Division lines x_{mid} and y_{mid} are used to form theses sets of ego footprints (refer to the table 3.1). Pseudo code for generating sets of ego footprints is given in algorithm 1.

Set name	Summary	Footprints from figure 3.11
S_{Total}	Set of all the bus footprints	$\{S_0$ to $S_{17}\}$
S_{Right}	Subset of S_{Total} containing footprints that overlap or lie on the right side of x_{mid}	$\{S_{10}$ to $S_{17}\}$
S_{Left}	Subset of S_{Total} containing footprints that overlap or lie on the left side of x_{mid}	$\{S_0$ to $S_{13}\}$
S_{Top}	Subset of S_{Total} containing footprints that overlap or lie above y_{mid}	$\{S_2$ to $S_{17}\}$
S_{Bottom}	Subset of S_{Total} containing footprints that overlap or lie below y_{mid}	$\{S_0$ to $S_5\}$
$S_{Right/Top}$	Subset of S_{Right} that overlap or lie above partition line y_{mid}	$\{S_{10}$ to $S_{17}\}$
$S_{Right/Bottom}$	Subset of S_{Right} that overlap or lie below partition line y_{mid}	Φ
$S_{Left/Top}$	Subset of S_{Left} that overlap or lie above partition line y_{mid}	$\{S_2$ to $S_{13}\}$
$S_{Left/Bottom}$	Subset of S_{Left} that overlap or lie below partition line y_{mid}	$\{S_0$ to $S_5\}$

Table 3.1: List of ego footprint sets stored at different nodes in the Kd tree shown in the figure 3.11

Division of bounding region (of ego footprints) along x_{mid} and y_{mid} is also used to build a tree. Each node of the tree represents particular sub-region in the bounding region of ego footprints. Therefore each set of ego footprints generated above corresponds to a particular node in the tree from figure 3.12. Tree can be generated using recursive function. However in the current work, tree presented in the figure 3.12 is implemented manually (pseudo code given in algorithms 1 and 2). Such tree is used during collision checking against each footprint of every vehicle.

Position of the given vehicle footprint in the divided space corresponds to a particular node in the tree. This node can be found by doing a search in the tree based on the position of the vehicle footprint with respect to the division lines x_{mid} and y_{mid}. Pseudo code for this implementation is given in the algorithm 2. Set of ego footprints corresponding to this node is used for collision detection against given vehicle footprint. Rest of the ego footprints do not collide and hence do not require collision detection.

Consider the car footprint (at $t = t_m$) shown in the figure 3.11. Car footprint can have any orientation and hence its bounding box (referred as 'Box' in figure 3.12) is used to carry out search through tree. Car footprint (or Box) lies on the left of the line x_{mid} and above the line y_{mid}. This corresponds to the node $S_{Left/Top}$ in the tree from figure 3.12. The set of ego footprints stored at this node is $S_{Left/Top} \Rightarrow \{S_2$ to $S_{13}\}$. Therefore given car footprint requires collision detection against ego footprints $\{S_2$ to $S_{13}\}$.

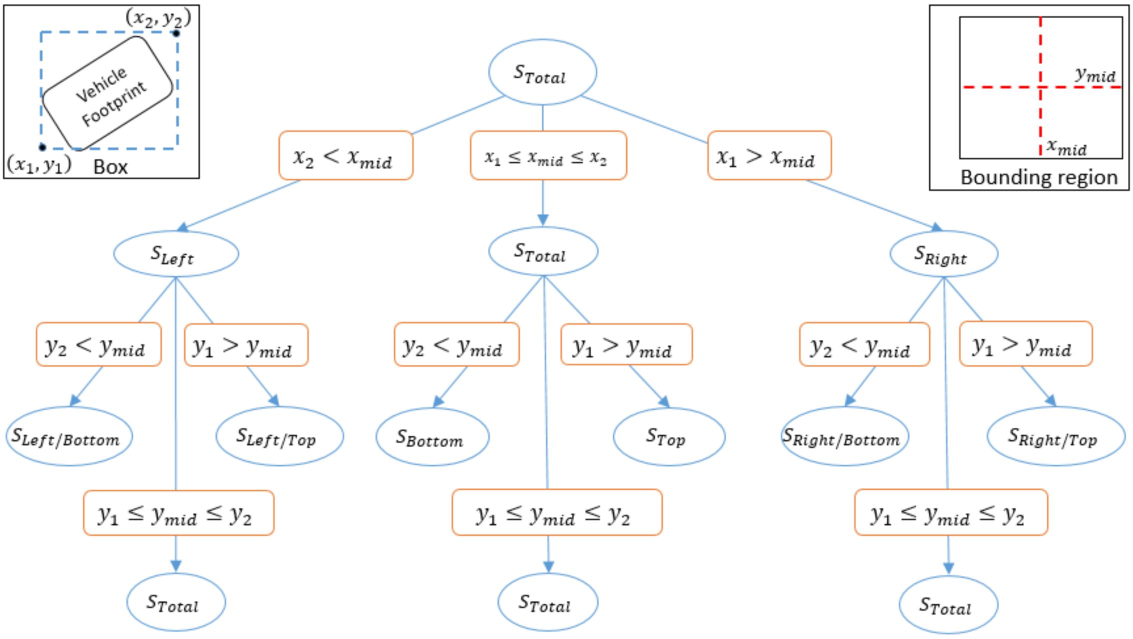


Figure 3.12: Formation of a tree for the collision checking

Depth of the tree is just 2 in the implementation in this work. However, use of kd-tree or quad-tree for collision checking is much more efficient in reducing computations [14, 15]. Kd tree is a binary tree in which nodes of the tree are populated by the objects based on their position in the space [14]. It is a type of a binary tree which is often used in the problems like nearest point search. Recursive function is used to divide the space and populate the tree with different sets of ego footprints. Similar to algorithm 2, a search function is used to search through kd tree for node corresponding to vehicle (car) position. This method can reduce the order of computation of collision checking process up to $NlogN$ [14]. A typical collision checking process of 25 vehicle footprints against 25 ego footprints will be able to achieve the order of computation $25log25 \approx 34.95$ using proper kd tree implementation which would otherwise become $25^2 = 225$. Therefore, this use of kd-tree or quad-tree is recommended for implementing collision checking against large number of vehicles and objects.

```
Data: Efootprints − 3D array storing corners $[x_1, x_2, x_3, x_4; y_1, y_2, y_3, y_4]$ of all the
      ego footprints
Result: Sets of ego footprint in divided space
Result: Tree parameters - $Left_{bounding}$, $Right_{bounding}$, $Top_{bounding}$, $Bottom_{bounding}$,
        $x_{mid}$, $y_{mid}$
  • Set $N$ equal to total number of ego footprints;
  • Set $S_{Total}$ to 1D array from 1 to N;
  • Generate array(size $1 \times N$) storing bounding box of each ego footprint:
    $Left$ - array storing $\min(x_1, x_2, x_3, x_4)$ of each ego footprint,
    $Right$ - array storing $\max(x_1, x_2, x_3, x_4)$ of each ego footprint,
    $Top$ - array storing $\max(y_1, y_2, y_3, y_4)$ of each ego footprint,
    $Bottom$ - array storing $\min(y_1, y_2, y_3, y_4)$ of each ego footprint,
  • Calculate bounding region all ego footprints ($footprints_E$):
    Bounding region - $(x_{min}, y_{min})$ and $(x_{max}, y_{max})$-
    $Left_{bounding}$ - min(Left)
    $Bottom_{bounding}$ - min(Bottom)
    $Right_{bounding}$ - max(Right)
    $Top_{bounding}$ - max(Top)
  • Calculate division line:
    $x_{mid} = (Left_{bounding} + Right_{bounding})/2$ and
    $y_{mid} = (Bottom_{bounding} + Top_{bounding})/2$
  • Find sets of ego footprints in regions formed by $x_{mid}$ and $y_{mid}$:
for $i$ from 1 to $N$ do
    if $(Right(i) \leq x_{mid)}$ or $(Left(i) \leq x_{mid} \leq Right(i))$ then
        $Set_{Left} = Set_{Left} \cup i$ ;                       /* Left half region */
    end
    if $(Left(i) \geq x_{mid)}$ or $(Right(i) \geq x_{mid} \geq Left(i))$ then
        $Set_{Right} = Set_{Right} \cup i$ ;                     /* Right half region */
    end
    if $(Bottom(i) \geq y_{mid)}$ or $(Top(i) \geq y_{mid} \geq Bottom(i))$ then
        $Set_{Top} = Set_{Top} \cup i$ ;                         /* Top half region */
    end
    if $(Top(i) \leq y_{mid)}$ or $(Bottom(i) \leq y_{mid} \geq Top(i))$ then
        $Set_{Bottom} = Set_{Bottom} \cup i$ ;                   /* Bottom half region */
    end
end
$Set_{Left/Top} = Set_{Left} \cap Set_{Top}$
$Set_{Right/Top} = Set_{Right} \cap Set_{Top}$
$Set_{Left/Bottom} = Set_{Left} \cap Set_{Bottom}$
$Set_{Right/Bottom} = Set_{Right} \cap Set_{Bottom}$
  • Return all the calculated sets of ego footprints
  • Return tree parameters - $Left_{bounding}$, $Right_{bounding}$, $Top_{bounding}$,
    $Bottom_{bounding}$, $x_{mid}$, $y_{mid}$
```

Algorithm 1: Generating sets of ego footprints based on division of space

```
Data: Sets of ego footprint in divided space
Data: Tree parameters - $Left_{bounding}$, $Right_{bounding}$, $Top_{bounding}$, $Bottom_{bounding}$,
      $x_{mid}$, $y_{mid}$
Data: Bounding box of Car footprint - $[Car_{Left}, Car_{Bottom}, Car_{Right}, Car_{Top}]$
Result: SET - Set of ego footprints corresponding to given car footprint
if $Car_{Right} < Left_{bounding}$ then
    SET = $\phi$ ;                                   /* Empty set (No collision) */
else if $Car_{Left} > Right_{bounding}$ then
    SET = $\phi$ ;                                   /* Empty set (No collision) */
else if $Car_{Top} < Bottom_{bounding}$ then
    SET = $\phi$ ;                                   /* Empty set (No collision) */
else if $Car_{Bottom} > Top_{bounding}$ then
    SET = $\phi$ ;                                   /* Empty set (No collision) */
else
    if $Car_{Right} < x_{mid}$ then
        if $Car_{Bottom} > y_{mid}$ then
            $SET = Set_{Left/Top}$ ;                        /* Left-Top region */
        else if $Car_{Top} < y_{Bottom}$ then
            $SET = Set_{Left/Bottom}$ ;                  /* Left-Bottom region */
        else
            $SET = Set_{Left}$ ;                           /* Left half region */
        end
    else if $Car_{Left} > x_{mid}$ then
        if $Car_{Bottom} > y_{mid}$ then
            $SET = Set_{Right/Top}$ ;                      /* Right-Top region */
        else if $Car_{Top} < y_{Bottom}$ then
            $SET = Set_{Right/Bottom}$ ;                /* Right-Bottom region */
        else
            $SET = Set_{Right}$ ;                         /* Right half region */
        end
    else
        if $Car_{Bottom} > y_{mid}$ then
            $SET = Set_{Top}$ ;                              /* Top half region */
        else if $Car_{Top} < y_{Bottom}$ then
            $SET = Set_{Bottom}$ ;                        /* Bottom half region */
        else
            $SET = Set_{Total}$ ;              /* Overlapping with all regions */
        end
    end
end
Return SET
```

Algorithm 2: Searching in a tree to find corresponding set of ego (bus) footprints for a given footprint of the vehicle (car)

3.3.6.2 Minimum distance threshold

A sub-set of ego footprints that can possibly collide with the given vehicle footprint is generated using tree structure. It is possible to discard some of the ego footprints from this sub-set by calculating distance between the footprints. Consider one vehicle footprint and one ego footprint at distance 'd' from each other. Distance between two footprints (d) should be less than a threshold for collision to exist between these footprints.

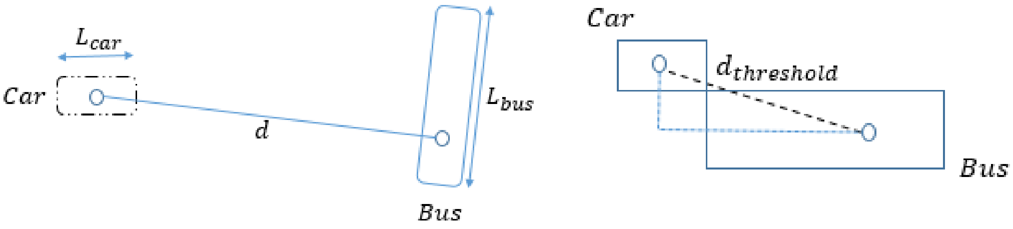


Figure 3.13: Distance checking

Figure 3.13 shows how the distance checking can help to avoid checking extensive overlapping of the rectangles represented by vehicle(car) and bus (ego). If $d > d_{threshold}$, then vehicles do not collide. Formula to find $d_{threshold}$ is given in equation 3.3 where dimensions of the bus and the car are inflated dimensions and d_{raxle} is the distance of the rear axle center from the center of the bus.

$$d_{threshold} = \left[\left(\frac{L_{car} + L_{bus}}{2} + d_{raxle}\right)^2 + \left(\frac{W_{car} + W_{bus}}{2}\right)^2\right]^{\frac{1}{2}} \tag{3.3}$$

Distance check is useful to avoid extensive collision detection between two distant footprints.

3.3.6.3 Heading direction of the vehicle

Refer to the bounding box of ego footprints in figure 3.14. Vehicle footprints that lie outside this bounding box do not obviously collide with any of the ego footprint. Furthermore, if such footprint is headed away from this bounding box, then the predicted footprints in the future also do not collide with any of the ego footprints. Hence, heading direction of vehicle footprint can be used to avoid extensive collision detection to some extent.

Consider the scenario in the figure 3.14 where vehicle footprints are plotted from the current time instance $t = t_0$ to time instance $t = t_{end}$. Vehicle footprints predicted after $t = t_m$ will not have any collision with any of the bus footprints. Therefore, footprints of the vehicle after time $t = t_m$ are not even required to be predicted for the analysis of collision checking. This method contributes to a significant amount of reduction in the computation from collision checking process.

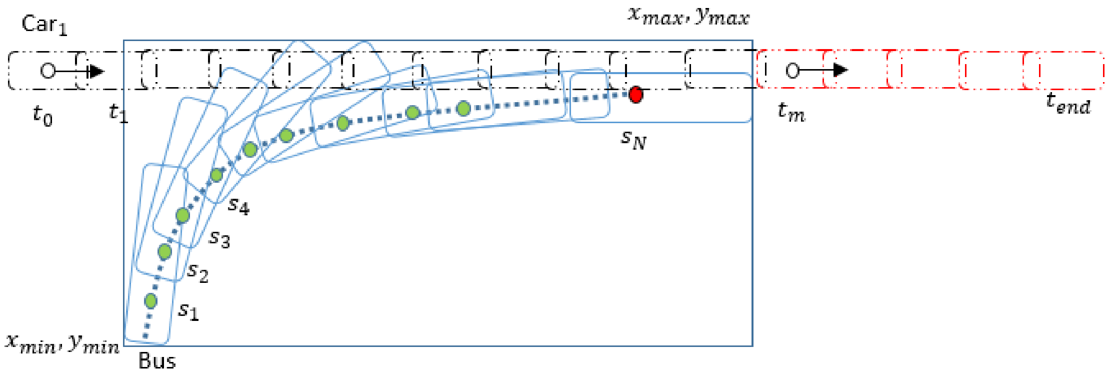


Figure 3.14: Optimization in collision checking operation using vehicle heading direction

3.3.7 Block diagram: Collision check implementation

Implementation of the collision check using three layers of optimisation mechanisms is presented in the figure 3.15. Current state of vehicles and Kd tree of ego footprints are the inputs to this function block of collision checking. This function block is executed in the loop for multiple vehicles (Not presented in this figure).

For a single vehicle at a given planning instance, vehicle positions are predicted at M (M is time horizon) future instances with the time interval of Δt. However, Kd tree is the same for all of the footprints of all the vehicles. Therefore, every successive vehicle footprint in the future is generated in the loop ($i = 0$ to M) and the collision checking is carried out for that vehicle footprint.

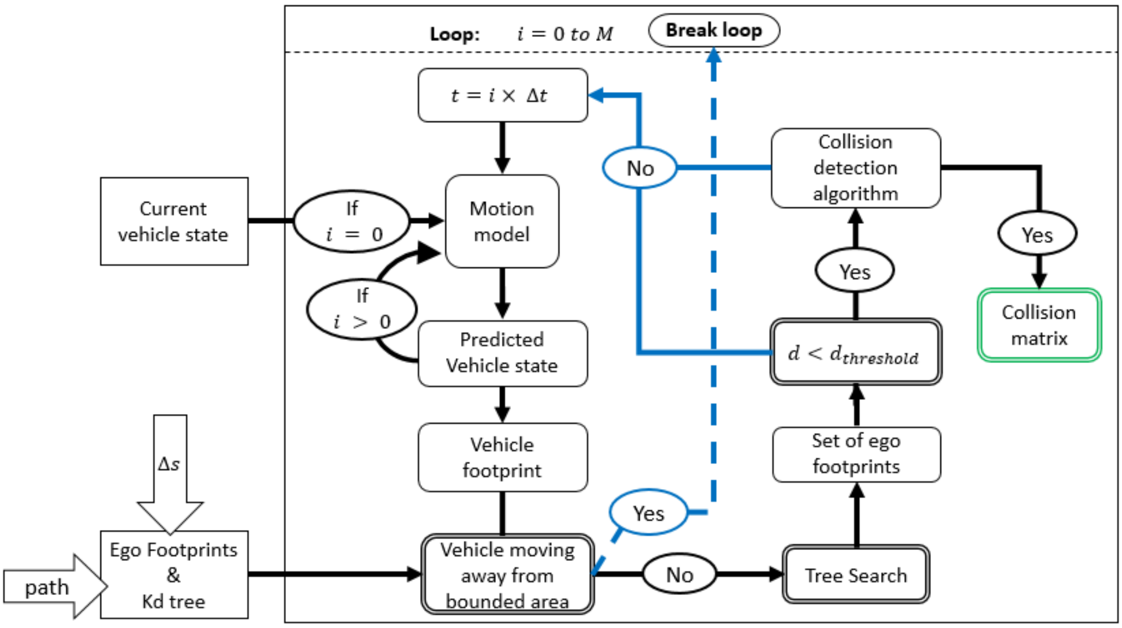


Figure 3.15: Block diagram: Implementation of collision checking

3.4 Velocity planning using space and time analysis

Let us have a look at space-time map in figure 3.16. One way to find the optimal path between the start position s_0 and the end line s_{end} is by making a grid search. It is possible to find the shortest path to reach the end line using search algorithm such as A*. Space-time map represented as a grid contains nodes and some of which represent collisions. More importantly space-time map is a reflection of the collision matrix that is discussed in the section 3.3.5. Acceleration of the ego vehicle is not considered while planning the velocity, instead the scheme for addressing the effect of finite acceleration is presented in the section 3.4.6.

3.4.1 Formation of optimization problem

Consider the following figure 3.16 representing space-time map as a grid. Here, the nodes in red color represent the collision region and those in green color represent the non-collision region. The grid spacing is defined by Δs and Δt. Size of a grid is $M \times N$, M being space horizon of the path of ego vehicle and N being time horizon for motion prediction. Goal is to reach the target line $(s = s_{target})$ using shortest feasible collision free path. Shortest path is associated with the minimum travel time which makes it a time optimal problem. Furthermore, any solution to this problem should also obey the constraints discussed in the section 2.6.1.3.

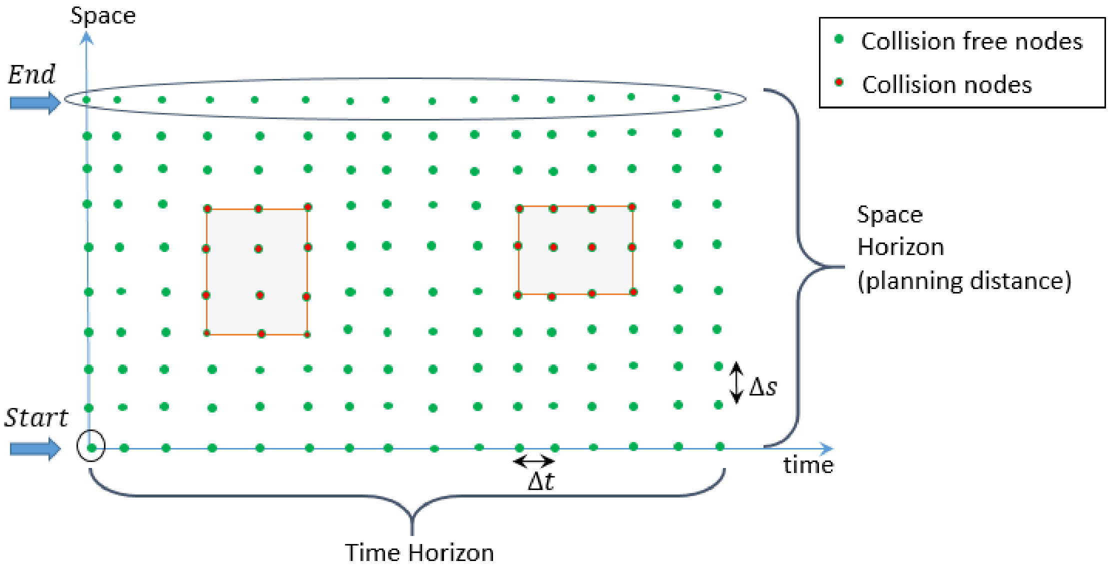


Figure 3.16: Space-time map as a grid

The optimization problem can be formulated as follows-

Objective: Find a shortest sequence of nodes starting from the start node (s_0, t_0) to any node on end-line $s = s_{target}$ subject to constraints -

- Sequence is strictly monotonously increasing along time axis
- Sequence is not decreasing along space axis
- All the nodes labelled as obstacles are to be avoided

3.4.2 A* search

Optimization problem formulated in the section 3.4.1 is solved using an A* algorithm stated in the section 2.6.5. Version of A* path search algorithm available on Mathwork file exchange [16] is used for implementation. This search function is modified in this work to solve above optimization problem. This section includes details on solving optimization problem using A* search.

Constraints stated in the previous section result in the lesser branching factor (refer to the section 2.6.5). Hence, search is always carried out in the forward direction (north-east search as shown in the figure 3.17) making the search process much faster than the original search. The A* algorithm maintains open list and closed list. The node in the open list with the lowest $f(n)$ value is explored. Since nodes in the closed list are not explored, all the nodes representing collision are put into the closed list.

A* search starts from the start node and searches in the direction of the target node with the help of heuristic based function value ($f(n)$ from the equation 2.5). At a typical stage during the search, successive node can be chosen via possible connections in the north-east direction as shown in the figure 3.17.

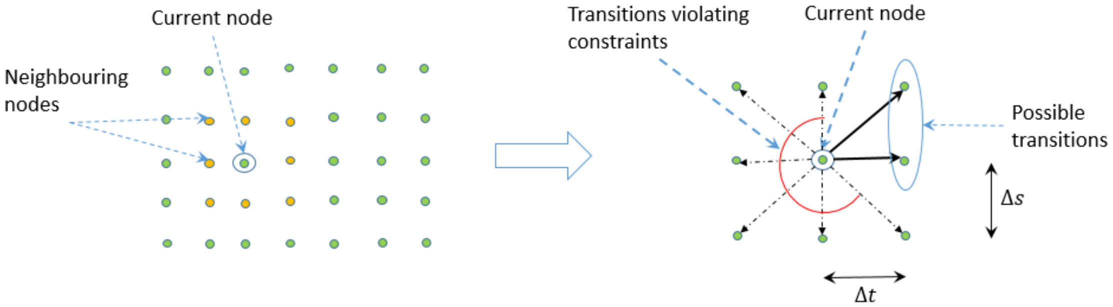


Figure 3.17: Search direction during A* algorithm

A* is an informed search algorithm. Total number of nodes explored by a non-informed search algorithm (such as depth-first) is of order $O(b^d)$, where b is branching factor and d is the depth of the solution (shortest path) [20, chapter 3, p. 74]. With suitable admissible heuristic (discussed in section 2.6.5), A* is much faster search algorithm. Reducing branching factor also increases the speed of the search by reducing computations.

Connection cost

Since this is a time optimal problem, all the costs and function values should also have the same unit as the time. Therefore, the cost of a connection from one node to the next is equal to the time required for that connection (refer figure 3.18).

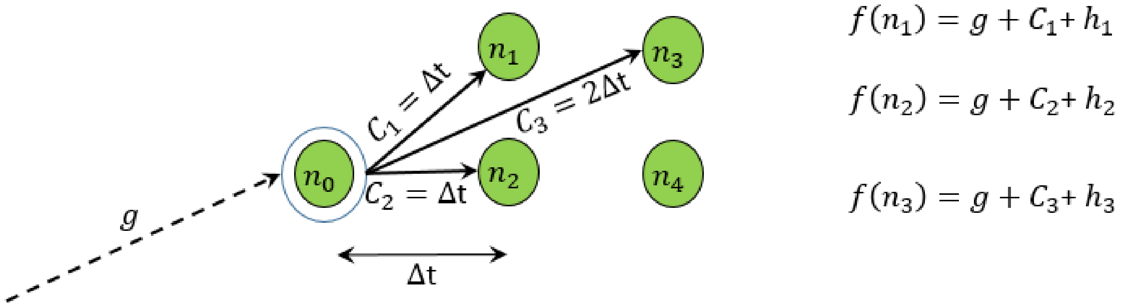


Figure 3.18: Calculating costs for different connections

Here, $f(n_i)$ is the function value at a given node n_i, g is the cost to reach the parent node, C_i is the connection cost from parent node and h_i is the heuristic as explained in the section 2.6.5.

Target node and heuristic

This search problem only demands to reach destination along the space (s_{target}) dimension and there is no hard constraint on the final time (t_{target}). Also, not all the nodes can reach the same target node because of maximum velocity constraint. Hence, there cannot be a single target node for all the explored nodes in the search.

In this problem, the heuristic is the estimated time to reach the final node. As discussed earlier in the section 2.6.5, heuristic should be admissible to ensure optimality. The admissible heuristic in this case should not be greater than the minimum time to reach the destination (t_{target}). Minimum time to reach destination is basically the time taken to reach destination with maximum speed. Hence, the admissible heuristic for any node ($n(t_n, s_n)$) in the search space is given by

$$H(n) \leq \frac{s_{target} - s_n}{V_{max}} \tag{3.4}$$

V_{max} corresponds to the direction of the search along maximum speed in the space-time map, using which one can also find out an individual target node for any particular node during the search. Refer to the figure 3.19 which shows how the heuristic is calculated for three given nodes using above equation. Maximum speed in this figure corresponds to the connection along diagonal ($V_{max} = \frac{\Delta s}{\Delta t}$). Implementation becomes simple by using integers (unit spacing) instead of actual spacing (with standard SI units) between nodes. All functions and costs can be calculated using integers to ease the calculations. In this figure 3.19, yellow nodes correspond to the destination (s_{target}).

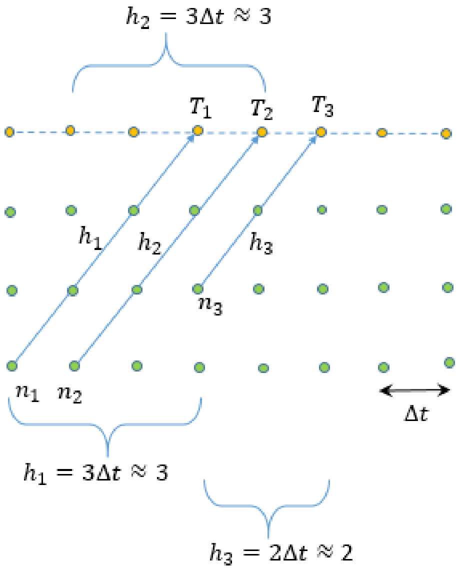
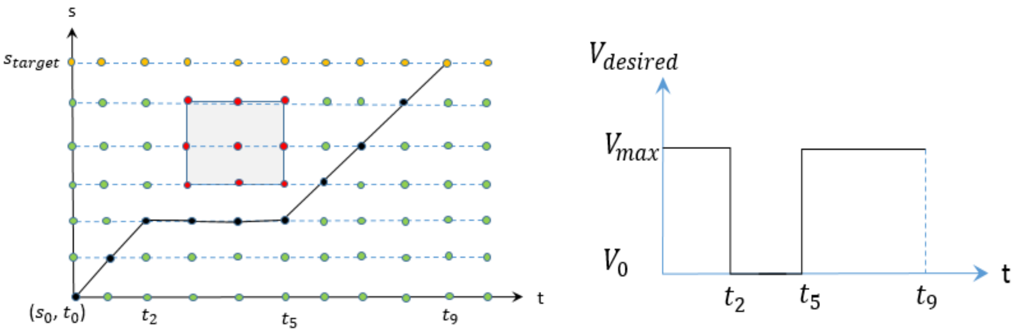

$h_2 = 3\Delta t \approx 3$
T_1
T_2
T_3
h_1
h_2
h_3
n_3
n_1
n_2
Δt
$h_1 = 3\Delta t \approx 3$
$h_3 = 2\Delta t \approx 2$

Figure 3.19: Heuristic calculation for nodes with corresponding target nodes

3.4.3 Discontinuity in speed and connectivity of nodes

Search algorithm is implemented using above method results in the optimal path (in space-time map) as shown in the figure 3.20a. Such optimal path results in the velocity profile as shown in 3.20b. One can observe that the speed profile provided by the optimal path is varying from zero to maximum (and vice versa). Such speed profile is a result of only 2 possible connections in the grid search where the vehicle can either go with finite speed or stop. Sharp transitions in the speed are unrealisable. Hence, optimal path should also ensure step by step changes in the speed profile. By having higher connectivity in the search direction based on the current speed, gradual changes in the speed can be obtained.



(a) Optimal path: Search with two types of node connections

(b) Sharp change in the speed levels

Figure 3.20: Optimal path with only two speed levels

This is where formulating the problem as a grid helps. In the center of the figure 3.21, all possible connections from a particular node are shown. During the search, based on the previous connection to reach the given node, only specific set of connections is allowed in order to allow small changes in the speed. Refer to the figure 3.21.

1. C_0 represents connection with speed, $v = 0$
2. C_1 represents connection with speed, $v = \frac{\Delta s}{3\Delta t}$
3. C_2 represents connection with speed, $v = \frac{\Delta s}{2\Delta t}$
4. C_3 represents connection with speed, $v = \frac{\Delta s}{\Delta t}$
5. C_4 represents connection with speed, $v = \frac{2\Delta s}{\Delta t}$

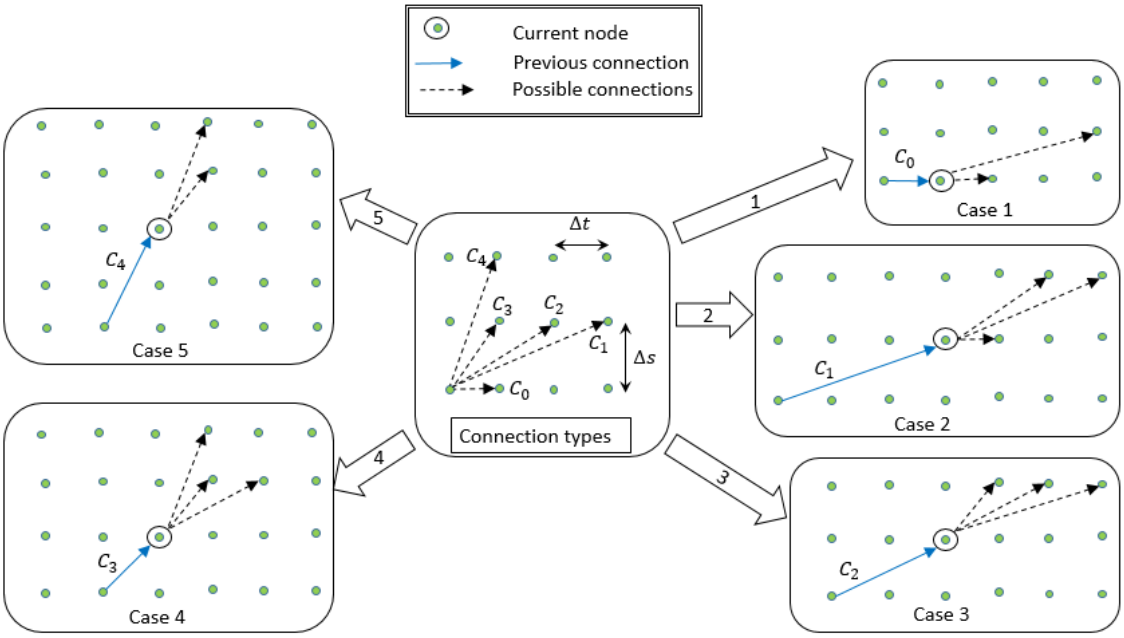
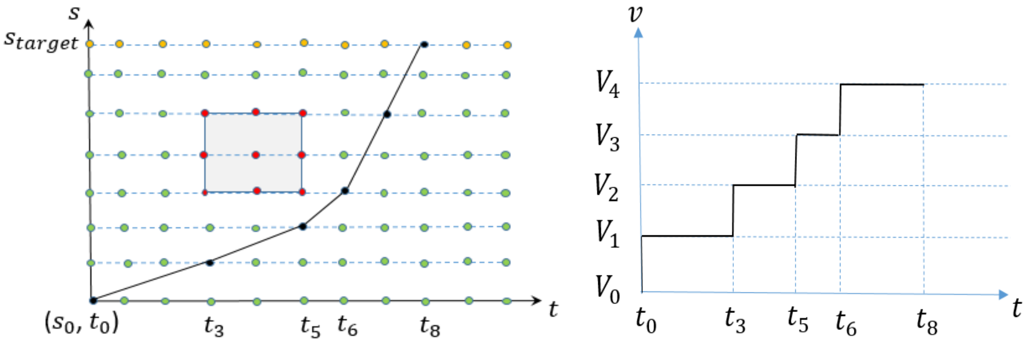


Figure 3.21: Search direction based on the previous connection

Therefore, there are five cases resulting in five different sets of possible connections from a *current* node as shown in figure 3.21. Such modification will result in the relatively smoother speed profile along the optimal path which is more preferred from the comfort point of view. Figure 3.22 shows optimal solution to the same problem in 3.20 with relatively smooth speed profile. Even though the change in the speed levels is gradual, it is still sharp which implies infinite acceleration. This issue is addressed in the section 3.4.6.



(a) Optimal path: Search with four types of node connections

(b) Gradual change in the speed levels

Figure 3.22: Optimal path with gradual change in the speed levels

This concept can be extended in the future work to include more speed levels. Also, search can be modified to have at least two consecutive node connections with same speed to achieve slower transitions between different speed levels. Therefore, there is a good scope of study of grid connectivity in the A* search to improve performance of this approach.

3.4.4 Motion at the end of the path

As the destination approaches, the vehicle needs to plan speed in order to stop at the destination while avoiding collisions. This can be realised by defining grid connectivity in A* to achieve gradual decrease in the speed levels. Following figure 3.23 shows how deceleration at the end of the path can be defined.

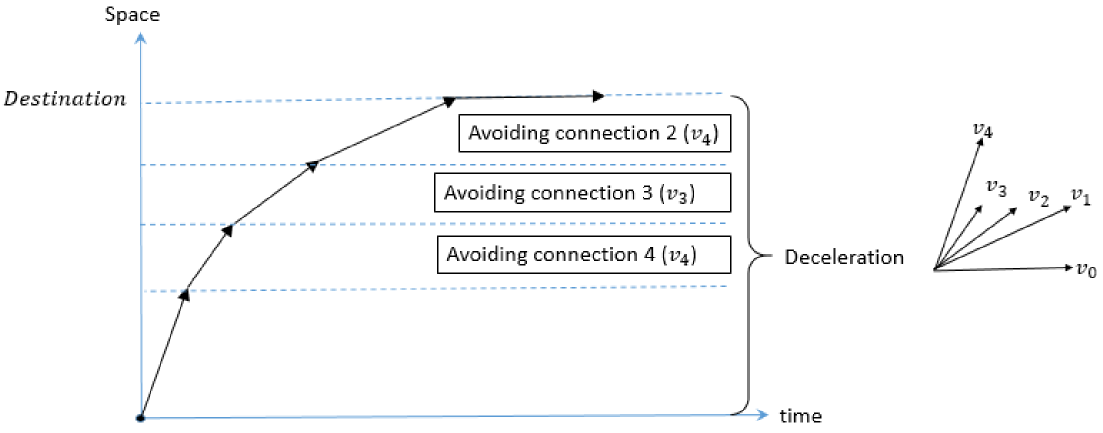

Space
Destination
Avoiding connection 2 (v_4)
Avoiding connection 3 (v_3)
Avoiding connection 4 (v_4)
Deceleration
time
v_4
v_3
v_2
v_1
v_0

Figure 3.23: Illustration to show how the deceleration at the end of the path is defined by not allowing 'higher speed connections' in the search at the nodes closer to the destination

3.4.5 Real time planning and Desired speed

Once the speed plan is available, next step is to execute it. Speed controller is a part of an inbuilt path following controller in the simulation environment of software *PreScan* (discussed in section 2.4.5). Figure 3.24 shows that the velocity planning is carried out at lesser sampling rate ($T_{sampling} = 0.5$ sec) than the sampling rate of rest of the system ($T_{sampling} = 1/50$ sec). Speed input to the controller is constant for 0.5 sec after which it receives re-planned reference speed (further comments in section 4.1.2).

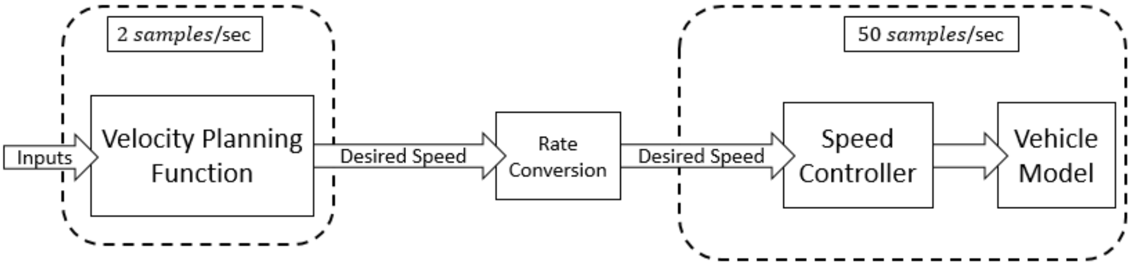


Figure 3.24: Block diagram showing interface between the velocity planning function and the speed controller

Desired speed input (or reference speed) for the speed controller needs to be calculated from the generated velocity plan. At the end of A* search, there could be two outcomes: Optimal solution or no solution. In the case of no solution, vehicle may take following actions:

1. Re-calculating the speed in order to stop before nearest collision in the space time map or at a stop line the road such as zebra crossing
2. Slowing down or stopping immediately by applying hard brakes

In this project, hard brakes are applied in order to stop vehicle when the solution to optimization problem seize to exist. So, absence of the optimal solution is one of the challenges that needs to be addressed in the future.

Desired speed is calculated using two methods. Consider figure 3.25, showing speed profile generated using velocity planning.

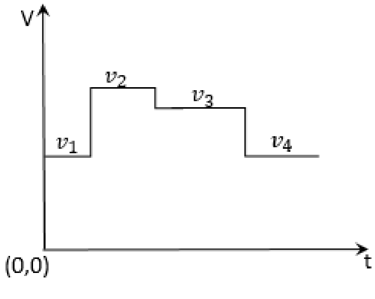


Figure 3.25: Time optimal speed profile generated by velocity planning function.

In the experiment 1 from section 4.2.1, $v_{desired} = v_1$. It is quite possible to have different time optimal speed profiles in two consecutive re-planning instances because of -

1. Discrete speed levels: Speed plan consists of only few speed levels
2. Finite set of feasible solutions due to discrete nature of search space
3. Continuously changing positions, speeds and directions of all the vehicles
4. Rapid changes in traffic and corresponding collision predictions

Speed levels defined via A* are finite. If the traffic situation requires ego vehicle to travel with the intermediate speed between the predefined speed levels, then desired speed input to the controller will switch between these two speed levels. Such rapid changes in optimal speed profile may result in flickering of reference input to the speed controller (as shown in the figure 3.26).

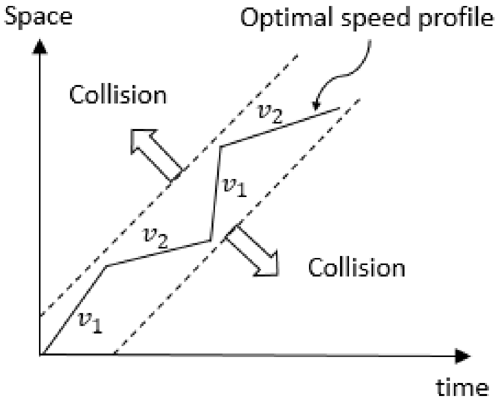


Figure 3.26: Discrete set of speed levels in the A* search may cause switching between two speed levels in certain cases due to lack of appropriate speed level

In experiments 2 and 3 from sections 4.2.1 and 4.2.2, moving average technique is used to calculate the reference input for the controller. Moving average is a type of a low pass filter that can be used to reduce effect of short term fluctuations while keeping long term trends [21, chapter 15, p . 277]. Moving average of previous, current and three future speed values from the generated speed vector is given in equation 3.5.

$$v_{desired} = (v_{k-1} + v_k + v_{k+1} + v_{k+2} + v_{k+3})/5 \tag{3.5}$$

Weighted moving average can also be used to have specific type of performance. Figure 5.2 shows how moving average can have a stable speed input to the controller.

3.4.6 Effect of finite acceleration

Speed profile generated here assumes instant change in the reference speed to the speed controller which cannot be achieved by real vehicles with finite and continuous acceleration. The actual speed profile of the vehicle looks similar to the one shown in figure 3.27. Let us say that vehicle 1 has infinite acceleration and vehicle 2 has finite acceleration. If vehicle 1 follows desired speed profile exactly, then it will cover more distance than vehicle 2 which follows the desired speed profile with finite acceleration. This difference in the distance is represented by the area of shaded region the figure 3.27.

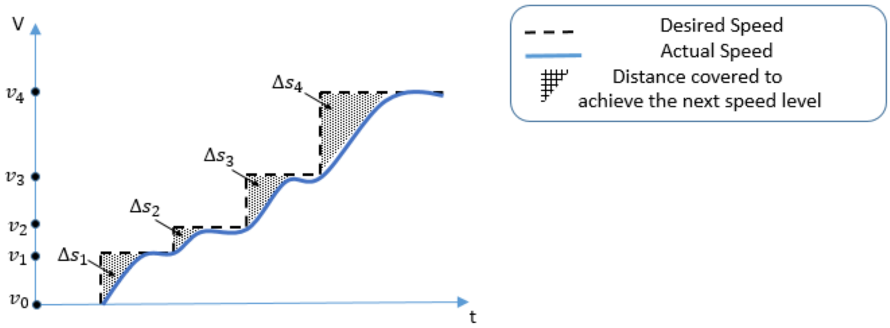


Figure 3.27: Velocity vs time plot for vehicle with finite acceleration. Area of shaded region between two adjacent velocity levels is showing the lag in the distance covered in actual motion from the distance covered in desired motion

The time required to cover a particular distance is also different in the case of finite and infinite acceleration. Figure 3.28 shows deviation of the actual motion from desired motion in space and time (Δs_i and Δt_i). In this figure, Δs_1 is the distance lag caused by the speed change from v_0 to v_1. Similarly, Δs_2 is the distance lag caused by the speed change from v_1 to v_2 and so on. Δt_1 is the time lag caused by the speed change from v_0 to v_1. Similarly, Δt_2 is the time deviation caused by the speed change from v_1 to v_2 and so on. Similarly, deviation can be expected in case of deceleration where actual motion (s(t)) leads the desired motion in space-time map.

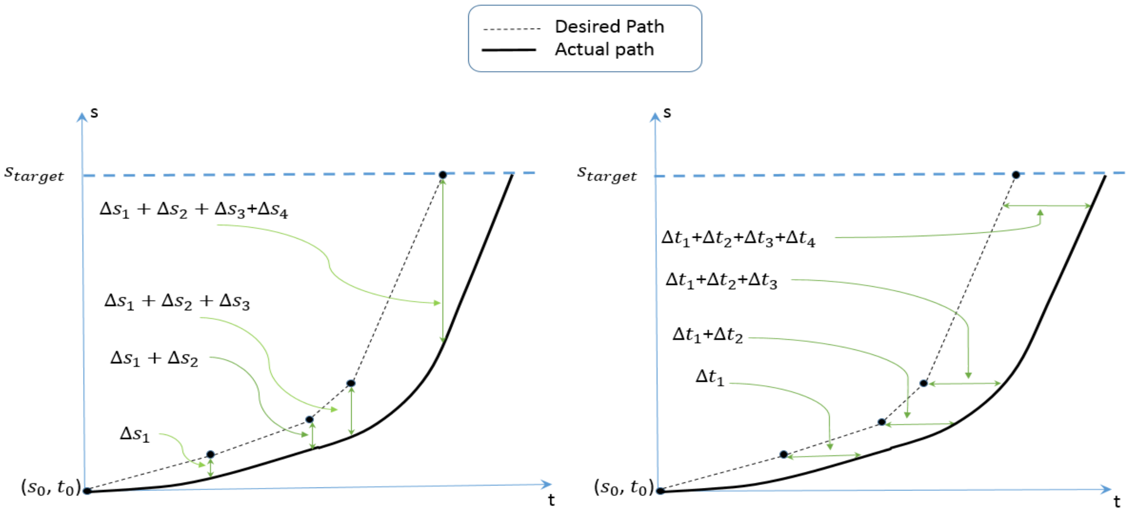

Desired Path
Actual path
s
s_{target}
$\Delta s_1 + \Delta s_2 + \Delta s_3 + \Delta s_4$
$\Delta s_1 + \Delta s_2 + \Delta s_3$
$\Delta s_1 + \Delta s_2$
Δs_1
(s_0, t_0)
t
s
s_{target}
$\Delta t_1 + \Delta t_2 + \Delta t_3 + \Delta t_4$
$\Delta t_1 + \Delta t_2 + \Delta t_3$
$\Delta t_1 + \Delta t_2$
Δt_1
(s_0, t_0)
t

Figure 3.28: Figure showing desired motion and actual motion of the ego vehicle in space-time map. Δs_i and Δt_i is distance lag and time lag respectively from the desired motion plan caused by finite acceleration of the vehicle

Finally, figure 3.29 shows the implications of not considering such effect of finite acceleration in the motion planning. The planned motion (shown by dotted line) seems collision free however the actual motion (shown by bold line) results in situation of potential collision in the future.

Real time velocity planning overcomes this limitation by re-planning speed over the time. Since other vehicles on the street also respond to the motion of ego vehicle, such collision may not occur in reality. However, it is still important to consider this deviation during velocity planning in order to achieve better performance and higher margin of safety.

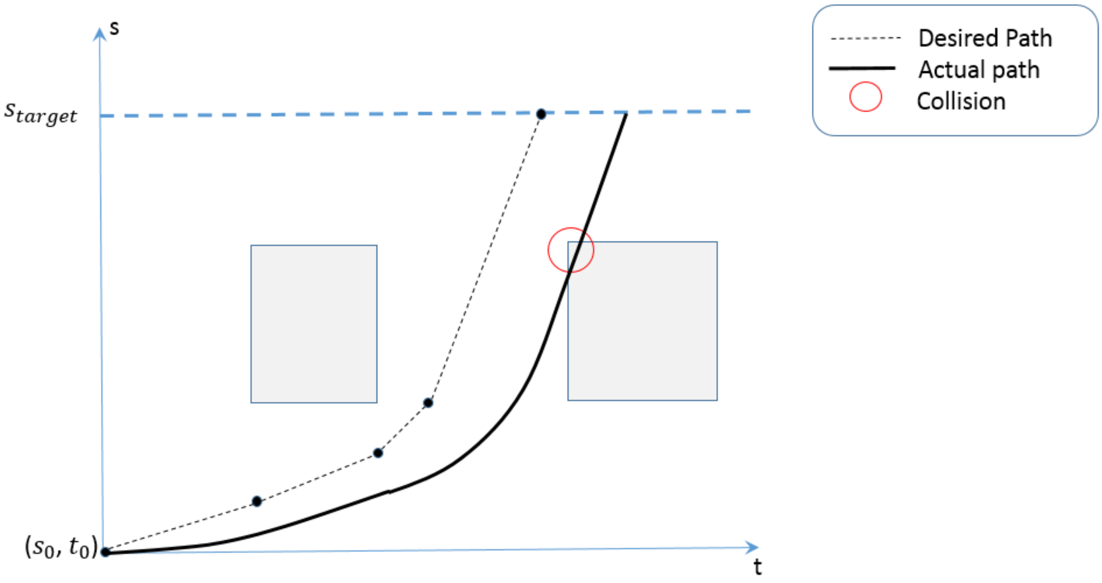


Figure 3.29: Desired motion Vs actual motion in space-time map. Deviation from the planned motion (desired motion) resulting in the potential collision in the future

In order to compensate for the deviation from desired motion, one can add a safety margin while searching for the optimal path in space-time map. Margin can be derived by studying the deviations in the space and time (Δs_i and Δt_i) when speed changes from one level to another level (as shown in figure 3.28). Figure 3.30 shows an example of how the otherwise optimal path can be disregarded as the collision is detected at the marginal node.

It is worth noting that the method proposed in this section is not implemented in this thesis work due to lack of acceleration data of simulation model. This method is proposed for overcoming the limitation of this approach in the future work by including the vehicle dynamics in the grid connectivity. With the help of vehicle dynamics and acceleration values for a particular vehicle model, search connectivity can be designed to overcome the limitations of finite acceleration.

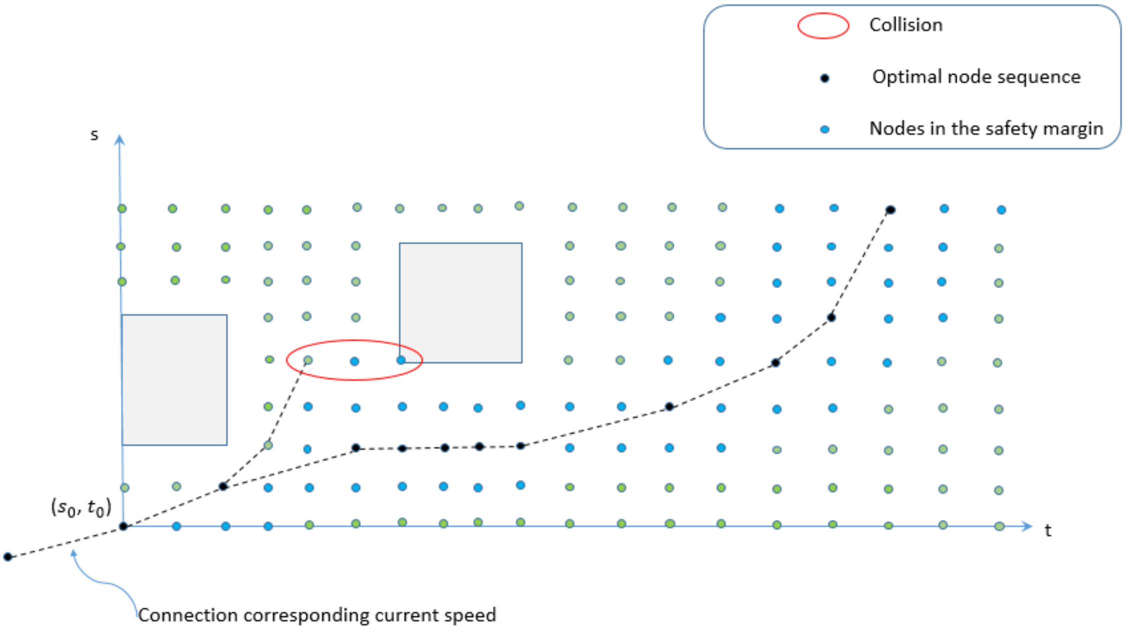

Collision
Optimal node sequence
Nodes in the safety margin
s
t
(s_0, t_0)
Connection corresponding current speed

Figure 3.30: Adding safety margin can result in the velocity plan which is more conservative than the one without safety margin while ensuring a collision free motion for the vehicle with finite acceleration

3.5 Planning distance and planning frequency

It is important to investigate the relationship between the planning distance, braking distance and planning frequency for safer motion planning. Let say that $B\ meters$ is the braking distance (at $v = V_{max}$), $T_p\ sec$ is the time interval for re-planning speed and $V_{max}\ m/s$ is the maximum speed of the vehicle. Vehicle travelling with V_{max} will move a distance of $V_{max} \times T_p\ m$ in time $T_p\ sec$. Planning distance P should be high enough to stop the vehicle within its limits due to the fact that the anything beyond planning distance is not known to the vehicle as far as space time analysis is concerned. Therefore, $P > B$. Since planning input to the speed controller (or brakes) arrives after every $T_p\ sec$ and distance travelled by the ego vehicle (with maximum speed) during this time interval should also be considered along with the braking distance. Hence, the relationship becomes

$$P > V_{max} \times T_p + B \tag{3.6}$$

3.6 Velocity planning function - System architecture

Entire approach developed in the previous sections is summarised in this section. Velocity planning function takes in following three inputs -

1. Sensor information: State of ego vehicle and all other vehicles on the street
2. Path of the ego vehicle
3. Invariants of the system: Vehicle dimensions, grid parameters, planning distance, etc.

Block diagram of velocity planning function is shown in the figure 3.31. Function block for collision checking is explained in the figure 3.15.

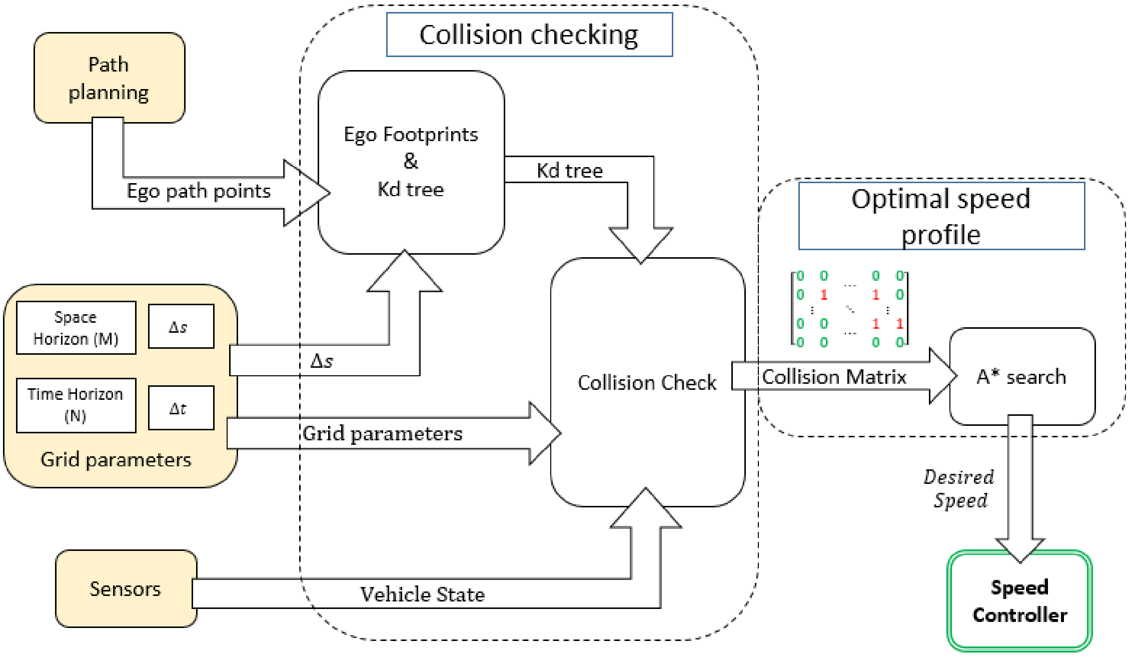


Figure 3.31: System architecture: Velocity planning function

4
Results

Objective of the thesis is to develop a 'velocity planning approach'. Therefore, there is no concrete base against which the algorithm performance can be compared other than the performance of a vehicle with a human driver. Simulation results presented in this chapter are understood by observing how the function requirements and characteristics presented in the section 2.2 fulfilled.

Velocity plan generated at the very first instance of the motion is used to compare the performance of the real time velocity plan (figure 4.8). If the speed of the other vehicles (apart from ego vehicle) on the street is constant then ideally the velocity plan should not change over the time due to deterministic nature of this approach. Hence, the velocity plan (initial plan) evaluated in the first instant can be used as a bench mark to compare the actual real time plan that is being followed. However, such type of comparison only makes sense when the surrounding vehicles do not change their speeds for entire maneuver of the ego vehicle. If the traffic is changing continuously, then there is no meaning to the comparison with the initial plan.

For this purpose, typical traffic scenarios are created and simulations are performed. This approach is very general in the sense of road geometry and hence following two road geometries are used for modelling the traffic scenario.

1. Traffic Junction
2. Traffic Circle

Both traffic junction and traffic circle demand high level of negotiation capabilities as compared to most other road geometries. At these places, vehicles also experience rapid changes in the speed based on real time perception of the changing traffic. Therefore, simulations are performed at these two road geometries and corresponding results are presented in this chapter.

4.1 Simulation platform - PreScan

Development and testing of velocity planning algorithm is carried out in the simulation software *PreScan*. This software has an interface with the *MATLAB-Simulink* through which all the calculations related to planning and vehicle dynamics are carried out. Entire traffic scenario is modelled in *PreScan* and corresponding *Simulink* model (also called as compilation sheet) is generated.

4.1.1 Ego vehicle Model

Bicycle model available in *PreScan* is used for the dynamics of the bus in this simulation. This model is able to simulate the vehicles longitudinal, lateral and roll motion. Bus model [17] used for this simulation is 'Scania Omni Bus' that is shown in the simulation results.

4.1.2 Simulation parameters

There are several parameters that are invariant for the simulation. Some of these parameters are co-related and hence they are to be evaluated before simulation. Following are the main invariant parameters that directly affect performance of the function.

Actual and increased sizes of the vehicle footprints Following table 4.1 shows sizes of the cars and the bus along with the increased sizes (length and width) of their footprints used in the collision checking. Change in height of the vehicle does not affect the size of footprint.

parameter	actual value (m)	inflated value (for footprint) (m)
Bus length	14	16.5
Bus width	2.55	3.55
Bus height	3.32	-
car length	4.7	6.2
car width	2	2.5
car height	1.5	-

Table 4.1: Sizes of the vehicles used in the simulation along with the corresponding inflated footprint dimensions

Maximum ego speed: One of the parameters on which most of the other simulation parameters are dependent is maximum speed of the ego vehicle for the given motion. The inbuilt velocity controller in the *PreScan* is a PID controller which generates a steady state error for a given step input. This steady state error is significant at higher speed inputs and hence maximum speed in all simulations is kept at $4m/s$ ($14.4Km/hr$).

Grid Spacing: Analysis on suitable values of Δs based on the size and speed of ego vehicle along with the performance specifications is out of the scope of this work. Space interval (Δs) of 1 meter is used for the simulations. A* search is implemented in a way to facilitate five different speed levels through five different connections (T_0 to T_4) as shown in figure 4.1.

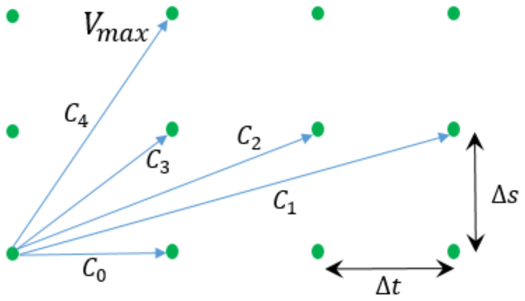


Figure 4.1: Different connections corresponding to different speed levels

Using V_{max} from the last paragraph, time interval for the collision checking, Δt is calculated as $\Delta t = \frac{2\Delta s}{V_{max}} = \frac{2\times1}{4} = 0.5\ sec.$ Please note that, this value of Δt also satisfies inequality derived in the equation 3.2 during the simulations. Therefore, the rest of the speed levels can be calculated as shown in the table 4.2.

Transition	Speed expression	value (m/s)
T_0	$v_0 = 0/\Delta t$	0
T_1	$v_1 = \Delta s/3\Delta t$	0.66
T_2	$v_2 = \Delta s/2\Delta t$	1
T_3	$v_3 = \Delta s/\Delta t$	2
T_4	$v_4 = 2\Delta s/\Delta t$	4

Table 4.2: List of the speed levels associated with the different connections defined in the A* search algorithm

Simulation frequency and planning frequency: In order to save the processing power, velocity planning is carried out with lower rate than actual sampling rate of rest of the functions in the simulation platform. Minimum time required for receiving new speed input in the space-time map is Δt. Hence, re-planning should be done no longer than Δt in order to remain consistent with the velocity plan from the previous planning instance (refer to the figure 4.2).

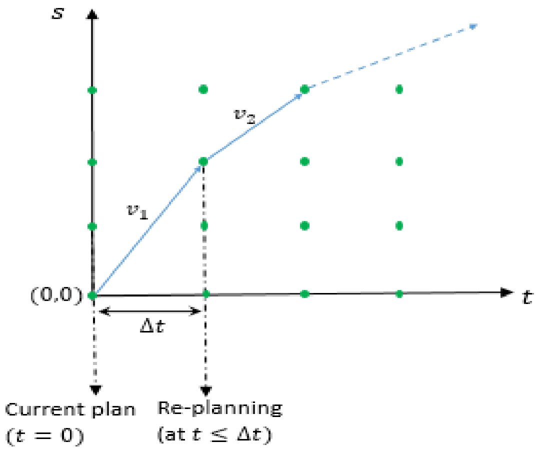


Figure 4.2: Time interval for re-planning is $T \leq \Delta t$

Therefore, re-planning is carried out after every $\Delta t = 0.5sec$ (sampling rate = $1/\Delta t$). These parameter values are summarized in the table 4.3.

Parameter	value	unit
V_{max}	4	m/s
Simulation - sampling rate	50	Hz
Δs	1	m
Velocity planning function - sampling rate	2	Hz
Initial ego speed	0	m/s

Table 4.3: Simulation parameters

4.2 Simulation Results

Critical instances in the simulation are captured and presented in following format:

1. *PreScan* screen shot
   - Traffic scenario in *PreScan* (ex. Left plot in figure 4.3)
2. Space-time search space showing collision nodes and optimal node sequence (ex. Right plot in figure 4.3)
   - X axis is representing the planning time (in seconds) starting from $t = 0$ ('current time instant') until time horizon. Hence, current position of the ego vehicle in this plot is always lying along the Y axis.
   - Y axis is representing the 'distance travelled by ego' along its path
   - 'Start' and 'End' on the left of the plot represent the starting point and end point (destination) of the entire ego path. Ego position advances along Y axis from 'start' to 'end' of the path.
   - 'Current' on the right of the plot denotes current ego position along Y axis.
   - 'Horizon' denotes distance up to which the planning is carried out.
   - Search space for a given space-time plot is a rectangular region confined between 'current' ego position and 'horizon'.
3. Collision checking plots showing intersection of vehicle footprints and ego footprints (ex. figure 4.7)

Planning frequency of 2 Hz (sampling time = 0.5 sec) had resulted in initial delay of 0.5 seconds. Hence, initial capturing instant is $t = 0.5$ seconds. Post simulation plots include:

1. Real time plan Vs actual speed
2. Executed plan (real time) Vs initial plan at $t = 0$

4.2.1 Experiment 1: Straight line motion at traffic junction

Here, the ego vehicle (Scania Omni bus) is trying to cross the junction while negotiating with the other two vehicles. Speeds of other vehicles are constant as given below.

1. Vehicle 1 (red vehicle) with speed = 4 m/s
2. Vehicle 2 (blue vehicle) with speed = 3.5 m/s

Search space is defined by space and time horizon as given below.

- Space horizon = 25 m
- Time horizon = 20 sec

Desired speed is calculated as the first element the velocity plan $(\{V_1, V_2, V_3, ..., V_N\})$ as given in the equation 4.1.

$$V_{desired} = V_1 \tag{4.1}$$

In the space-time map of figure 4.3 at $t = 6s$, the optimal path (blue line) goes around the corner of set of nodes that represent collision (red circles) while keeping some gap. At a later instant in the simulation shown by figure 4.4, current ego position and the collision node are next to each other with some gap between them. This distance-gap is the same as the one that is observed in the figure 4.3 at $t = 6$ seconds. This is a good illustration of how connectivity of the grid can be used to achieve distance keeping in this approach of velocity planning.

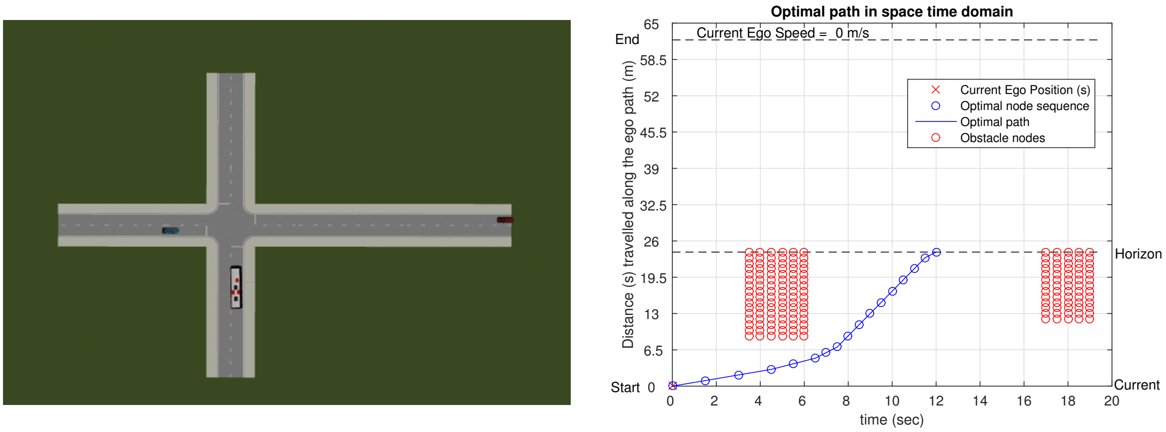


Figure 4.3: First iteration, $t = 0.5sec$

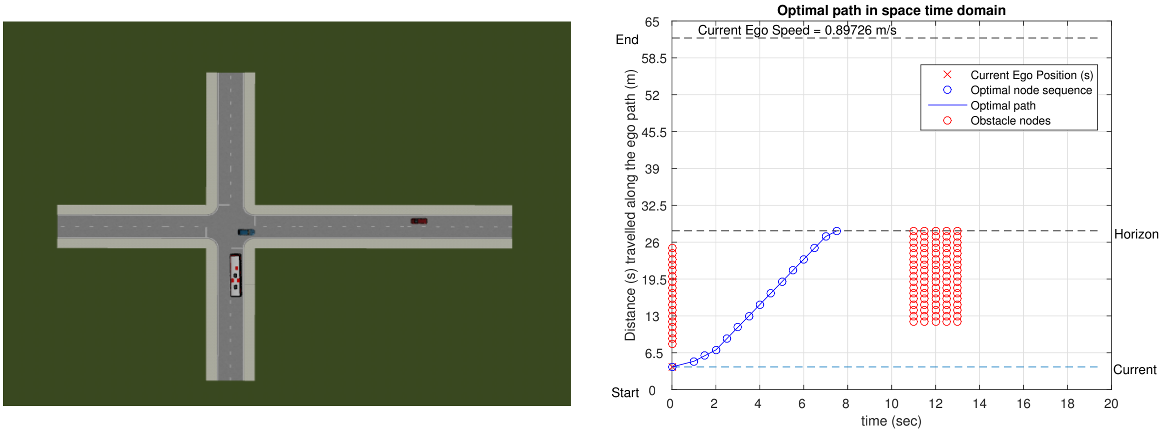


Figure 4.4: Intermediate iteration, $t = 6.5sec$

Space-time map in figure 4.5 shows another similar instance where time gap can be seen between ego vehicle and future collision. Hence, grid connectivity can be used effectively not only to plan collision free velocity profile but also to plan speed in a particular fashion to maintain certain distance and time margins while avoiding collisions.

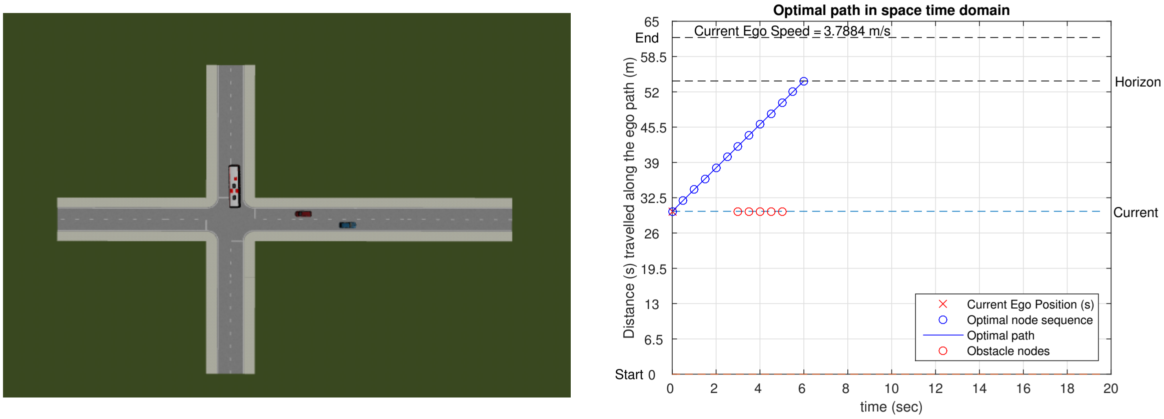


Figure 4.5: Intermediate iteration, $t = 14.5sec$

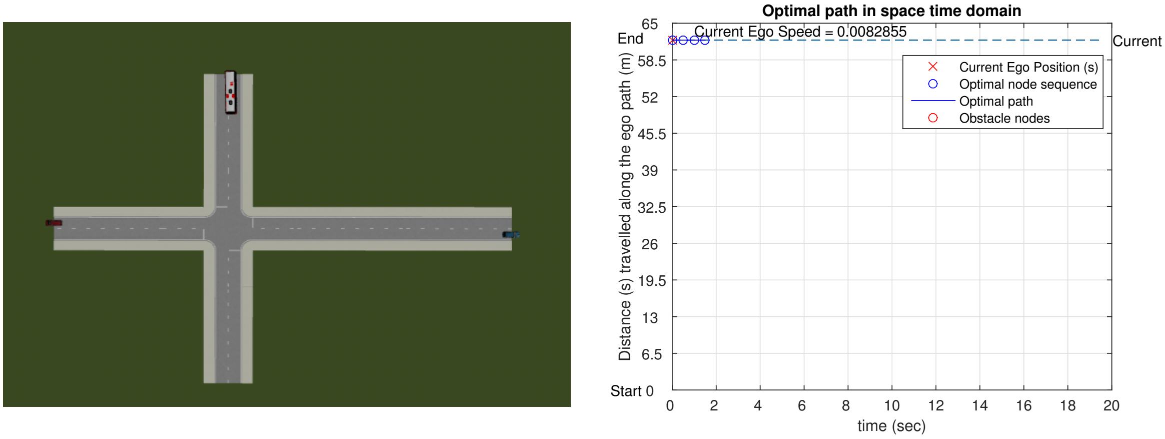


Figure 4.6: Last iteration, $t = 35.5sec$

Following figure 4.7 shows calculation of collision checking at above two instances. Footprints of ego vehicle intersecting with the footprints of moving vehicles results in a region represented by collision nodes (red circle) in the space-time map. Footprints of the ego vehicle are bounded in a bounding box. This bounding box is divided in the four quadrants to implement the collision checking using division of space as explained in the section 3.3.6.1.

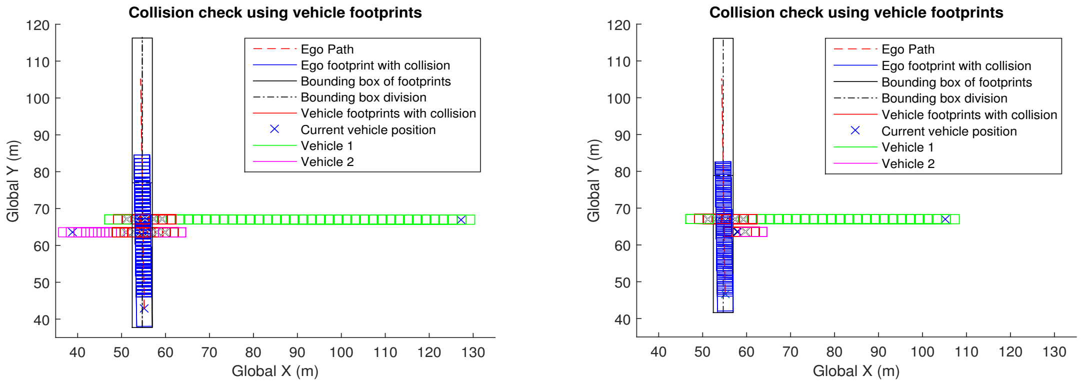


Figure 4.7: Matlab plot showing collision checking of vehicle and ego footprints at $t = 0sec$ (left plot) and $t = 6.5sec$(left plot)

Post simulation analysis can be done by observing certain variables such as ego speed $(v(t))$, distance of ego vehicle from other moving vehicles and also distance travelled by ego vehicle over time $(s(t))$. Performance of the velocity planning can be understood by looking at the actual speed of ego vehicle against desired speed input to the speed controller.

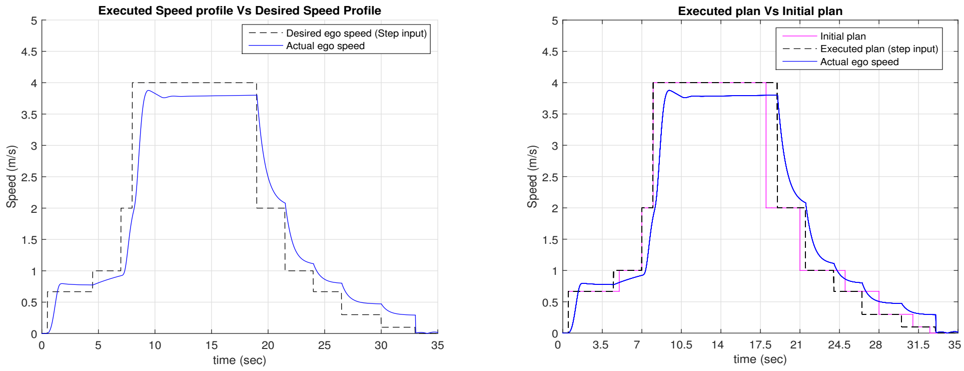


Figure 4.8: Executed plan Vs initial plan at $(t = 0.5sec)$

It is important to note that there are in total 7 reference speed levels that can be seen in the figures 4.8, 4.16 and 4.16. However, there are only five speed levels that are defined by the A* search for the simulation. The two extra speed levels $(v = 0.3m/s,\ v = 0.1m/s)$ at the end of path are implemented in order to achieve smoother braking via speed controller in the *PreScan*. These speed levels should be ignored while analysing the results since they occur after the negotiation with other vehicles has happened.

Left plot in figure 4.8 is showing actual ego speed vs desired speed over the time of simulation. Vehicle gradually escalates speed levels from 0 to 4 m/s and decreases gradually to 0 while avoiding the collision with moving vehicles. Vehicle speed increases and decreases with finite acceleration and deceleration creating a deviation from desired plan. Speed controller results in a significant steady state offsets at different speed levels adding more deviation from desired speed profile. Deviation caused by all these issues is effectively addressed by real time planning. This can be understood from the right plot in the figure 4.8. Right figure includes an extra curve which is referred as 'initial plan'. This initial plan is the velocity profile generated in the first iteration of the simulation. In this simulation, vehicles are moving with the constant speed along a straight line which means that the collision checking predictions will not change significantly over time. Hence, initial plan should not differ significantly from the actual plan. However, deviations caused by 'actual speed profile (blue curve)' are compensated by real time planning. This compensation can be seen as offsets between 'executed plan' and 'initial plan' at several instances as shown in the right hand side plot.

Figure 4.9 shows distance covered by ego vehicle as a function of time. This plot is basically the actual motion of bus in the space-time map.

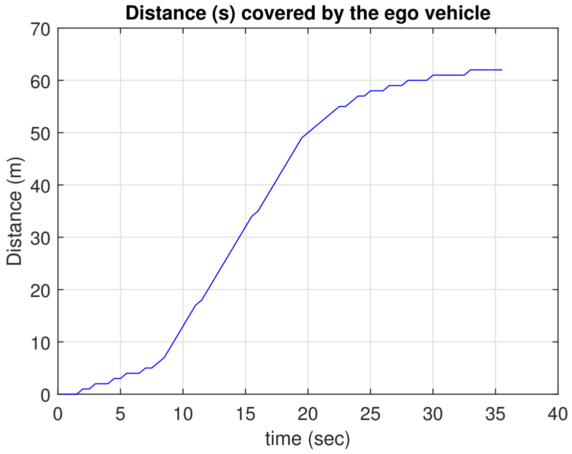


Figure 4.9: Distance covered by the ego vehicle over time ($s(t)$)

4.2.2 Experiment 2: Right hand turn at traffic junction

Here, the ego vehicle (Scania Omni bus) is going make a right hand turn at the junction while negotiating with the other two vehicles. Vehicle speeds are given below:

1. Vehicle 1 (red vehicle) with speed = 4 m/s
2. Vehicle 2 (blue vehicle) with speed = 3.5 m/s

Search space is defined by space and time horizon as follows:

- Space horizon = 25 m
- Time horizon = 20 sec

As mentioned earlier in the section 3.4.5, reference speed input to the conotroller is calculated as weighted moving average of the velocity plan ($\{V_1, V_2, V_3, ..., V_N\}$) as shown in equation 4.2.

$$V_{desired} = \frac{V_{old} + V_{current} + 1.5V_1 + V_2 + 0.5V_3}{5} \tag{4.2}$$

Simulation results are as given below. Figures 4.10 to 4.14 shows development of space-time map in real time motion of ego vehicle.

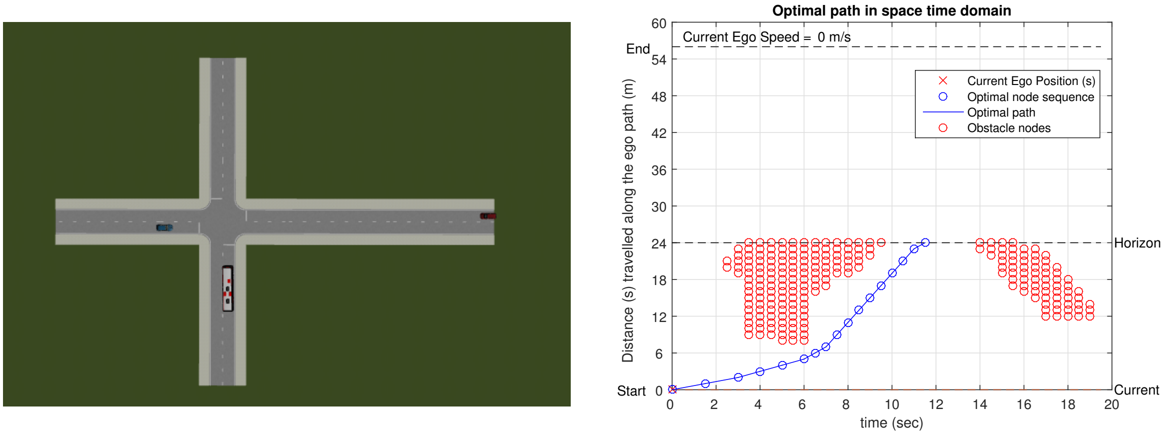


Figure 4.10: First iteration, $t = 0.5sec$

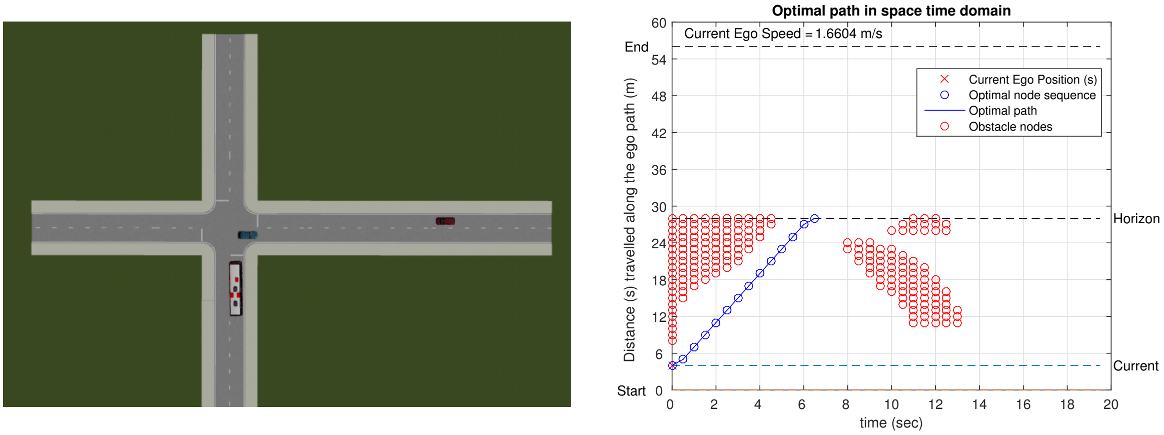


Figure 4.11: Intermediate iteration, $t = 6.5sec$

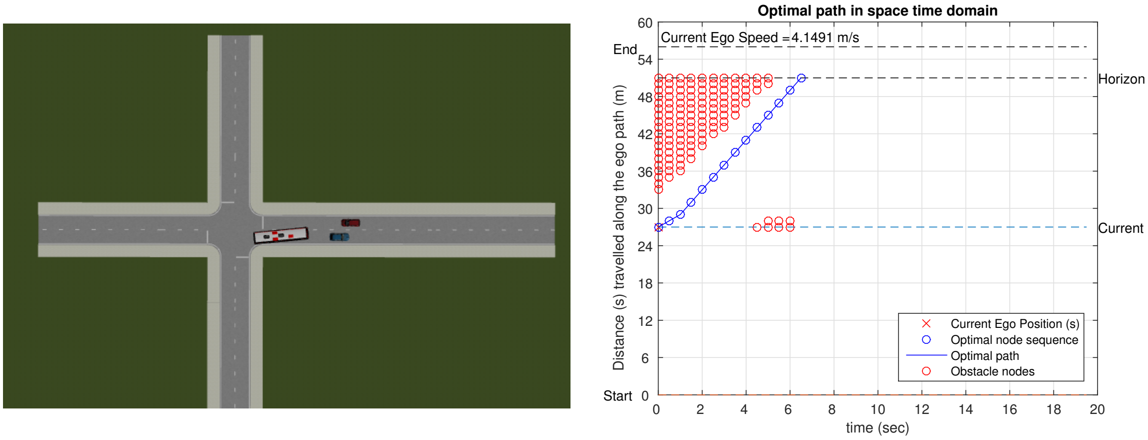


Figure 4.12: Intermediate iteration, $t = 12.5sec$

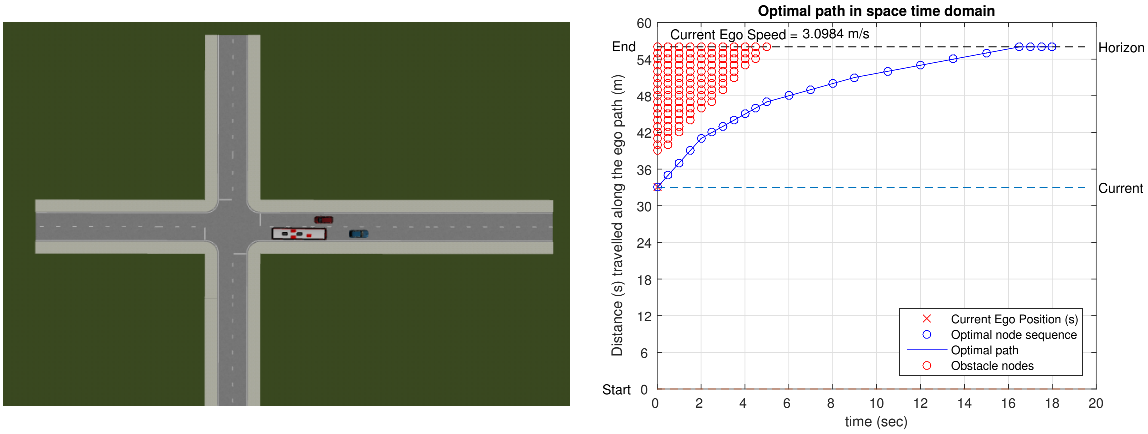


Figure 4.13: Intermediate iteration, $t = 14sec$

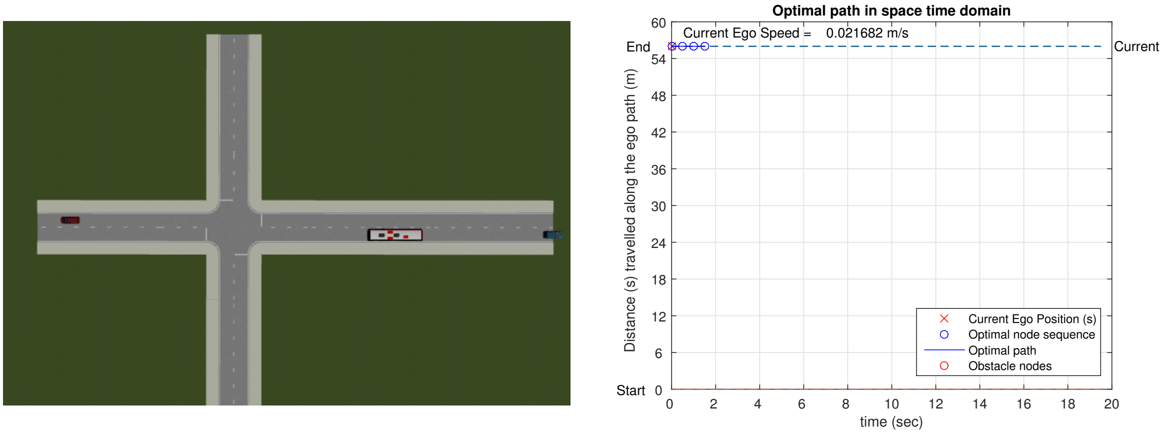


Figure 4.14: Last iteration, $t = 28sec$

Following figure 4.15 shows calculation of collision checking at time instances $t = 0.5sec$ and $t = 6.5sec$. It is possible to relate collision checking in figure 4.15 to the corresponding space-time maps.

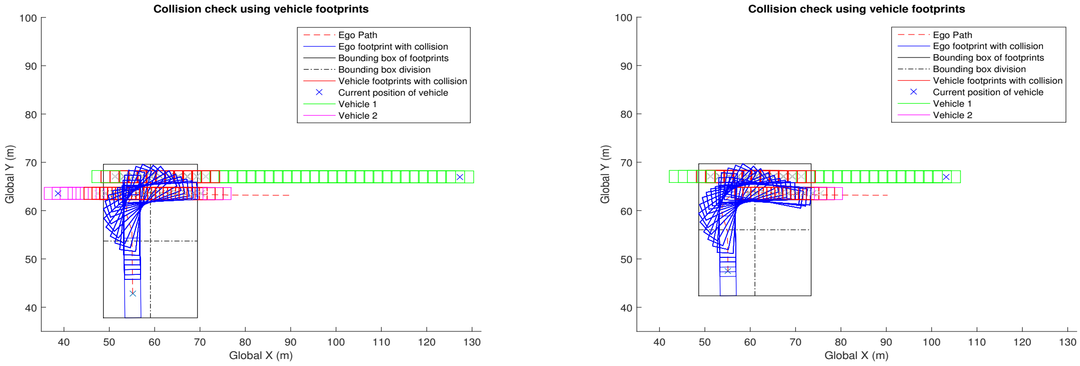


Figure 4.15: Matlab plot showing collision checking of vehicle and ego footprints at $t = 0.5sec$ (left plot) and $t = 6.5sec$ (right plot)

Left plot in the figure 4.16 shows the actual speed of the ego vehicle vs the desired speed (v_1 in the generated speed profile). In the other plot, initial velocity plan is also plotted. Since the speed of the other vehicles on the streets are constant, the executed plan should be same as the initial plan for the deterministic approach (no uncertainties in the model). However, due to finite acceleration of the ego vehicle and steady state errors of the speed controller, ego vehicle is not able to follow its initial plan. Executed plan however is result of re-planning after every $0.5sec$. This is why, the executed plan has offsets with the initial plan. This also proves that the real time planning is an essential feature of velocity planning function to overcome limitations of model as well as the plan itself.

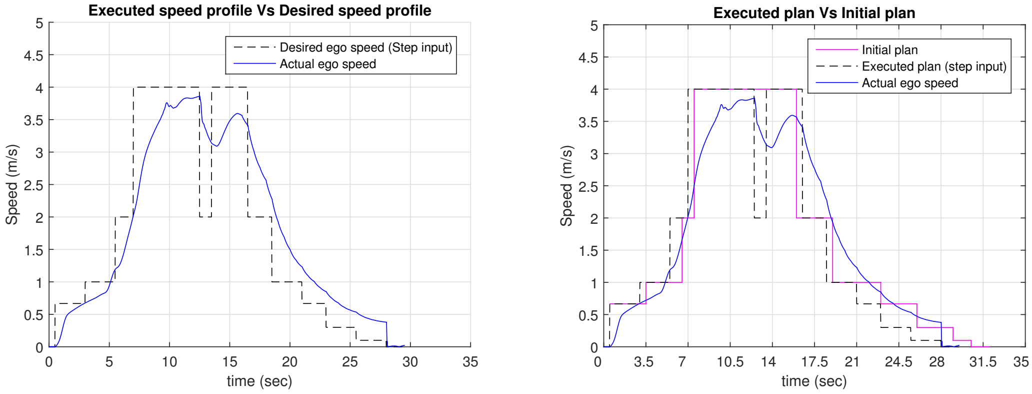


Figure 4.16: Executed plan Vs initial plan (at $t = 0.5sec$)

Figure 4.17 shows actual motion of the bus in space-time map.

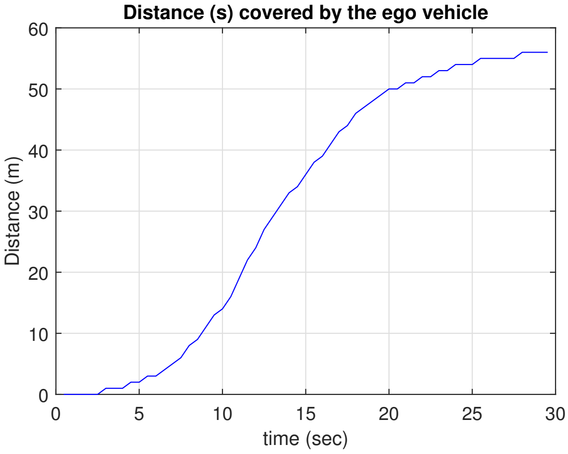


Figure 4.17: Distance travelled by ego vehicle over time ($s(t)$)

4.2.3 Experiment 3: Motion at traffic circle

Here, ego vehicle (Scania Omni bus) is going to cross the traffic circle while merging into the traffic. Vehicle speeds are given below:

1. Vehicle 1 (green vehicle) with speed = 4.5 m/s
2. Vehicle 2 (red vehicle) with speed = 3.9 m/s

Search space is defined by space and time horizon as follows:

- Space horizon $= 30\ m$
- Time horizon $= 20\ sec$

Reference speed input to the speed controller is calculated using moving average of the velocity plan ($\{V_1, V_2, V_3, ..., V_N\}$) as shown in equation 4.3.

$$V_{desired} = \frac{V_{old} + V_{current} + V_1 + V_2 + V_3}{5} \tag{4.3}$$

Simulation results are as given below. Figures 4.18 to 4.24 shows development of space-time map in real time motion of ego vehicle.

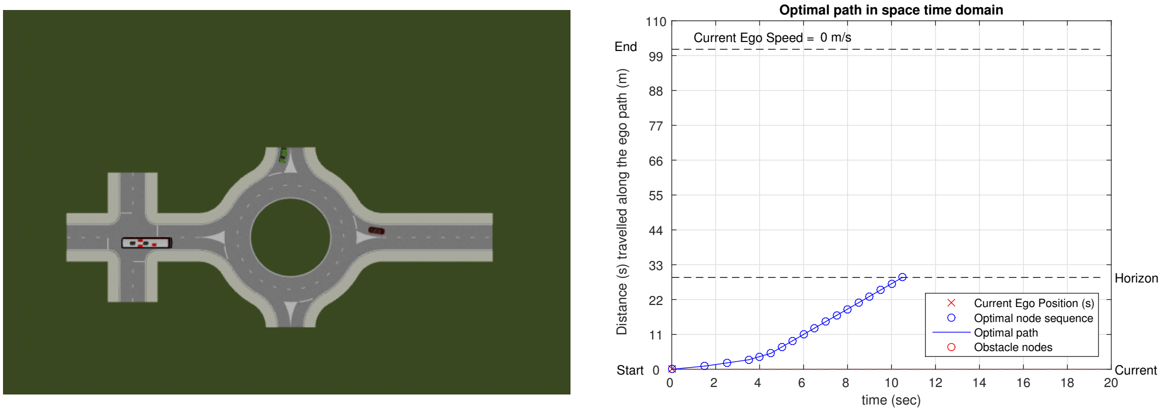


Figure 4.18: First iteration, $t = 0.5sec$

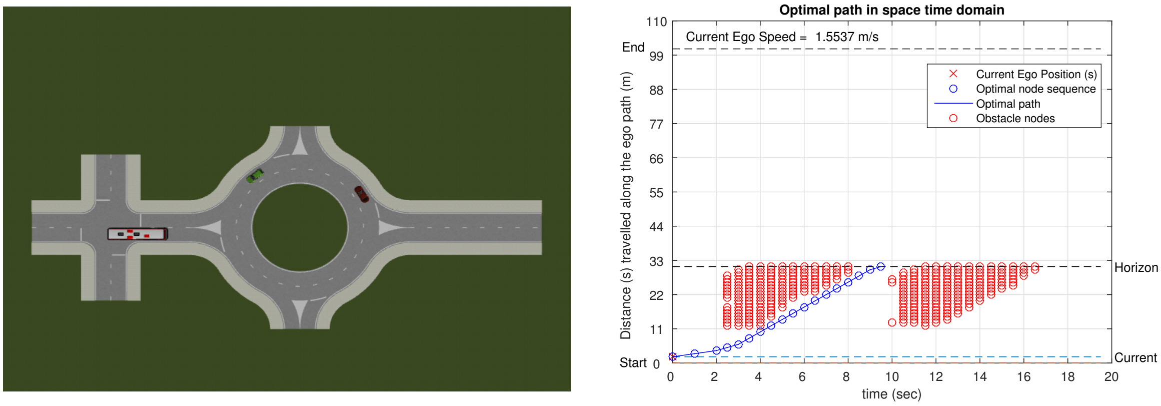


Figure 4.19: Intermediate iteration, $t = 3.5sec$

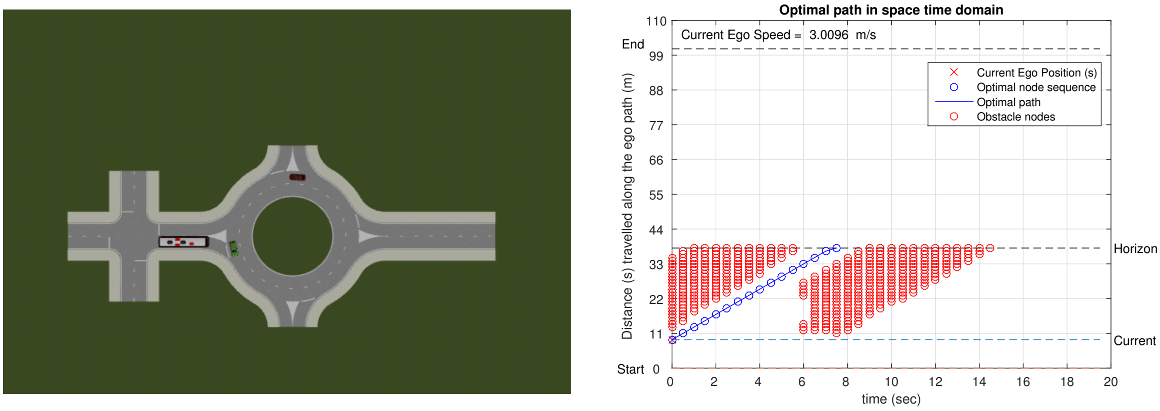


Figure 4.20: Intermediate iteration, $t = 7.5sec$

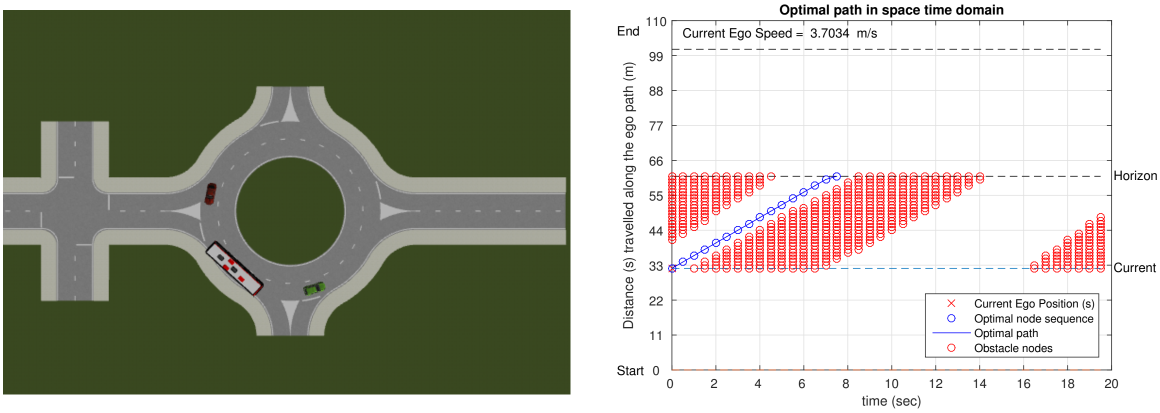


Figure 4.21: Intermediate iteration, $t = 13.5sec$

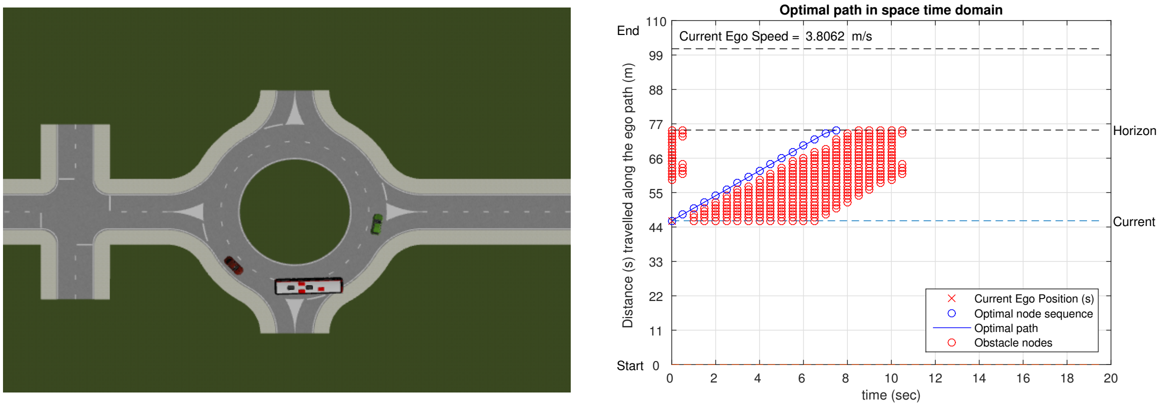


Figure 4.22: Intermediate iteration, $t = 17.5sec$

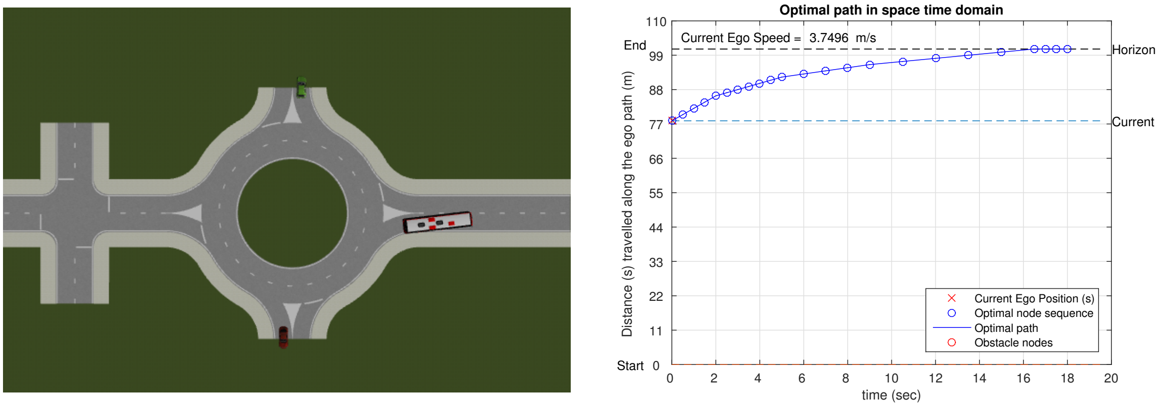


Figure 4.23: Intermediate iteration, $t = 25.5 sec$

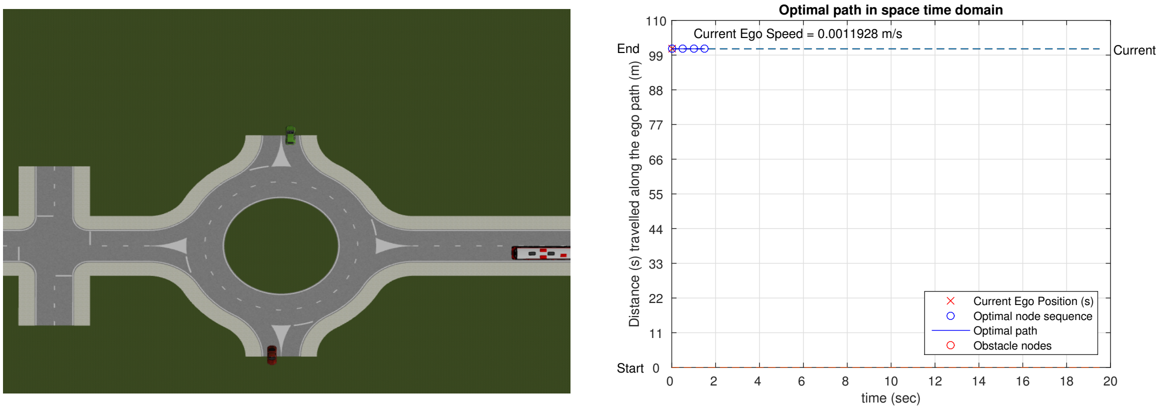


Figure 4.24: Last iteration, $t = 40 sec$

Following figure 4.25 shows calculation of collision checking using vehicle footprints at above two simulation instances. Collision checking is done by predicting the motion of the fellow vehicles using motion model as discussed in section 3.3. Left plot in the figure 4.25 shows example of poor motion prediction. These two plots differ by only 3.5 seconds but the parameter that makes a big difference is yaw rate. Yaw rate has opposite signs in these two instances making totally different perception of the situation. Such drastic change in the motion prediction affects the corresponding space-time plot which results in different speed profiles (refer space-time plot in 4.18 and 4.19). This is expected since the knowledge of road structure is not considered in the prediction step. Limited knowledge of the environment and lack of motion prediction of fellow vehicles put limits on its perception. This issue is discussed in details in the section 5.1.6.

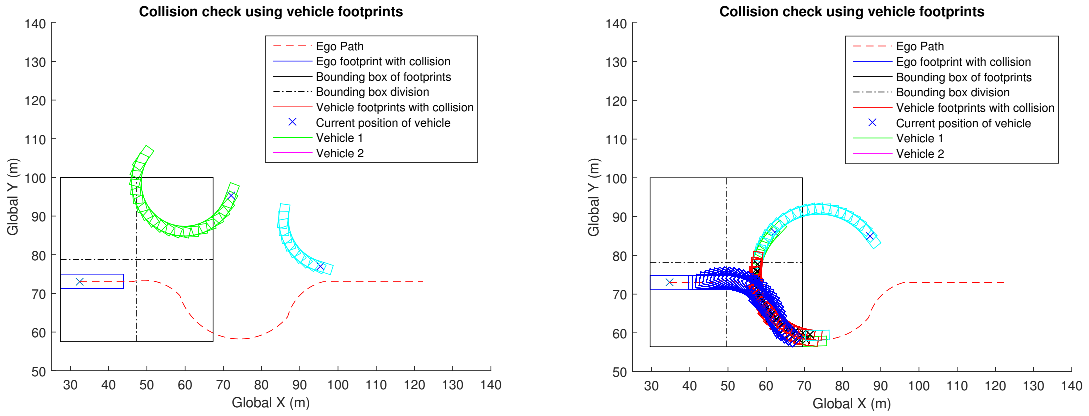


Figure 4.25: Matlab plot showing collision checking of vehicle and ego footprints at $t = 0.5sec$ (left plot) and $t = 3.5sec$ (right plot). Bounding box of ego footprints (black colour) and the corresponding partition shown by dotted lines are used to form Kd tree (for optimization in collision checking) at this planning instance

Left plot in the figure 4.26 shows actual ego speed Vs desired speed (v_1 in the generated speed vector). Green line in this plot corresponds to the moving average speed input to the speed controller (calculated using equation 4.3). It is interesting to see that the collision which is detected at time $t = 3.5sec$ is reflected in the corresponding speed plan. This 'change in plan' is happening due to lack of perception of motion of the vehicle 1. Having a comparison with the initial plan is not going to make any sense in this case since initial plan has already changed at $t = 3.5sec$ as discussed in earlier paragraph. Therefore, executed velocity plan is compared with the velocity plan at time $t = 33.5sec$.

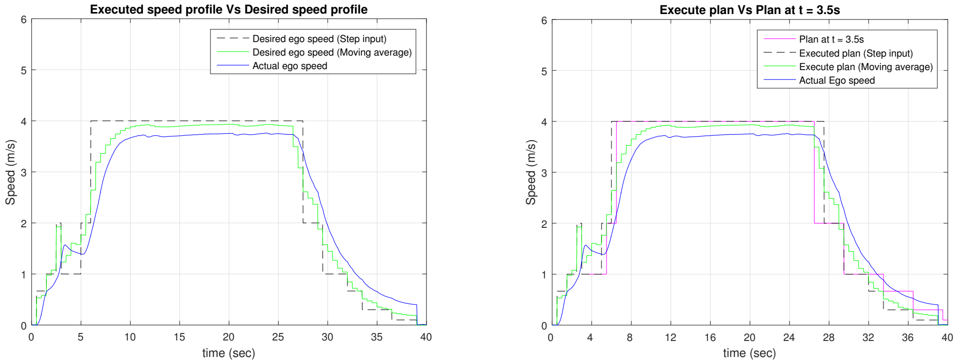


Figure 4.26: Executed plan (dotted line) Vs plan at $t = 3.5sec$ (pink line)

Following figure 4.27 belongs to the same time instance ($t = 3.5sec$) in the simulation as it is of figure 4.19 but with the bigger search space (or planning distance). This figure shows bigger search space with number of potential possibilities to merge into the traffic. Multiple collisions with the same vehicle are generated in the space-time map due to limitations of the motion prediction.

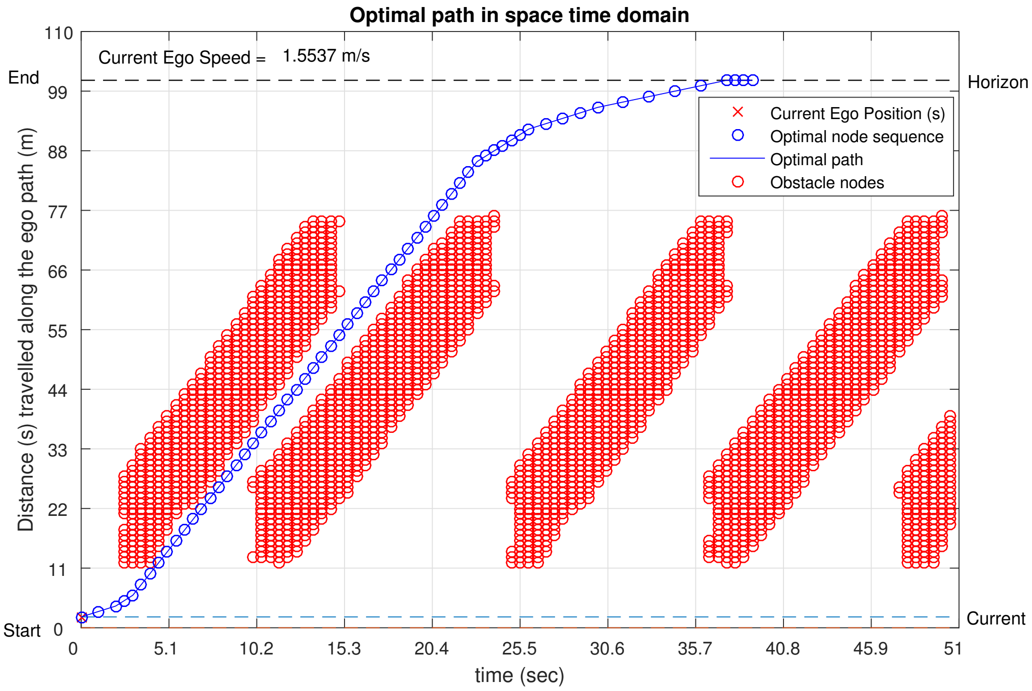


Figure 4.27: Complete path with bigger search space at $t = 3.5$ sec

Figure 4.28 shows the position of the bus at discrete positions on the space-time map as a function of time. Hence, this figure fails to capture the dip in speed due to change in plan at time $t = 3.5sec$.

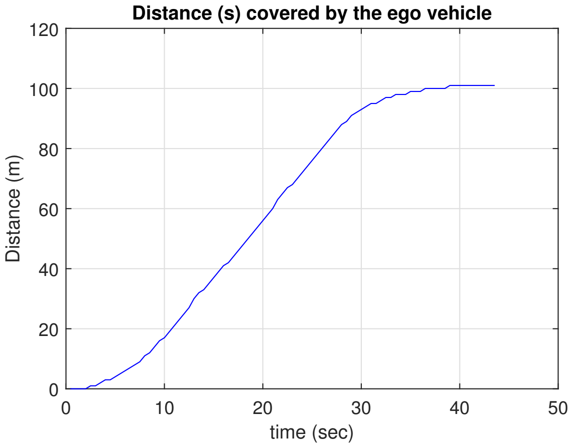


Figure 4.28: Distance travelled by ego vehicle over time $(s(t))$

5
Discussions

Key issues causing limitations to this approach are observed during the simulation. These issues and corresponding limitations of this approach are discussed in this chapter. These limitations also provide the direction for the future study and development associated with this approach. Therefore, possible future work directions are also proposed in this chapter.

5.1 Limitations: Key issues

Final outcome in this approach is based on series of computations from different sub-functions along with the values of certain parameters. Hence, it is important to understand how these functions and parameters result in the limitations to the overall performance of this velocity planning approach.

5.1.1 Discretization of the search space

Discretization of the search space allow us to find the solution efficiently but also introduces various limitations to this approach.

1. Discrete search space results in a finite set of feasible solutions
2. Absence of the solution through the search in the discrete space-time map does not necessarily imply absence of solution in continuous space -time map.
3. In case of lack of a feasible solution, re-planning is required to make an informed choice.
4. Discrete nature of problem formulation allows only fixed speed levels for ego vehicle. In general, velocity planning function should be able to deal with any speed up to the maximum speed of the ego vehicle. Having fixed speed levels may also result in the lack of comfort due to undesirable changes in the speed.

5.1.2 Flickering in optimal speed profile

Discrete nature of space time grid allows only specific combinations of node connections in search of optimal path. This may result in different optimal speed profiles for consecutive iterations despite of no or little change in the traffic situations. Refer to the figure 5.1 where four consecutive iterations with four different velocity profiles. Initial connection in the optimal path decides desired reference speed for the controller. Therefore, reference input to the controller fluctuates which we are calling as flickering.

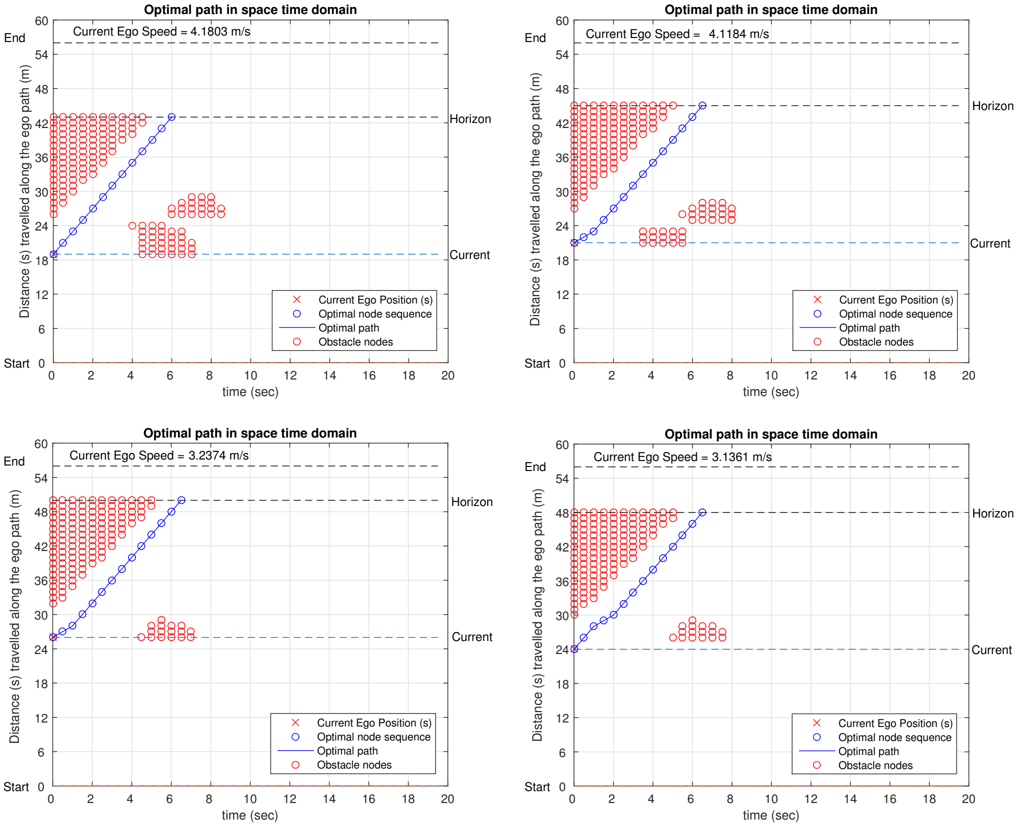


Figure 5.1: Four consecutive iterations (clockwise from top left) showing how different optimal solutions are generated due to little changes in the space time grid

Left plot in the figure 5.2 shows the flickering in the speed profile. This is not desirable behaviour and hence needs to be solved either by ensuring unchanged speed profile over time or by employing some kind of low pass filter at the input to the speed controller. Moving average is a type of low pass filter that not only eliminates the short term fluctuations by keeping long term trends but also can be used as a weighted mean. This allows us to tune it for desirable performance.

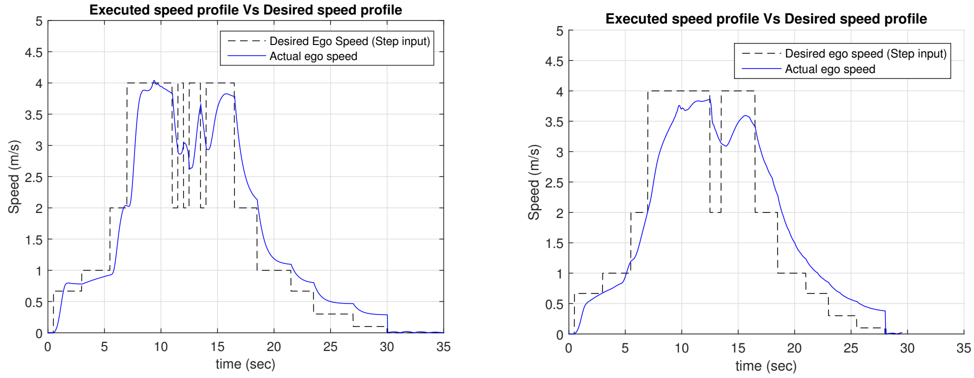
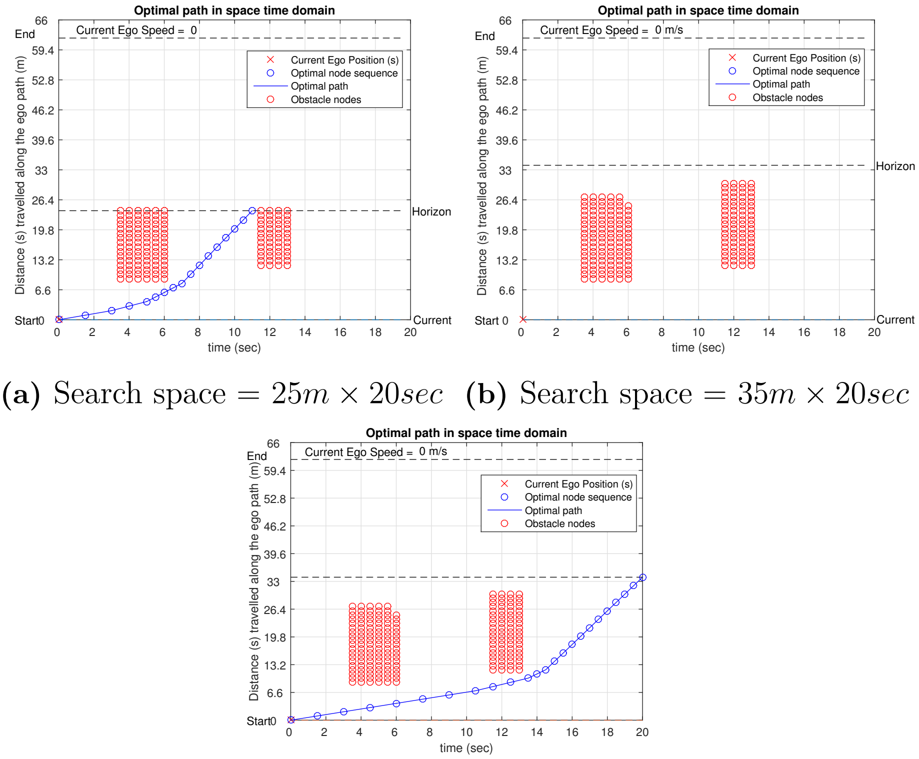


Figure 5.2: Left figure showing the flickering in the desired speed and right figure showing filtering using moving average of desired speed

In the experiments from sections 4.2.2 and 4.2.3, the moving average of the first three terms in the velocity profile ($\{v_1, v_2, v_3, ..., v_{N-1}, v_N\}$) is used together with the current and previous speed values. Right plot in the figure 5.2 shows effect of moving average input to the speed controller by reducing the fluctuations in the actual ego speed. Weighted moving average that is used in this simulation (right hand side plot) is calculated according to the equation 4.2.

5.1.3 Limited search space

Size of the search space is defined by the space and time horizons. This factor directly affects the availability of the feasible solutions. Larger search can help in better planning but it is computationally expensive. Following figure 5.3 shows how different search space results in solutions for the space-time plot of same traffic situation.



(a) Search space = $25m \times 20sec$ **(b)** Search space = $35m \times 20sec$

(c) Search space = $35m \times 22.5sec$

Figure 5.3: Optimal solution for a given traffic scenario with different search space sizes. No feasible solution exist for the example in figure (b)

In the figure 5.3a, solution (optimal path) does not realise the collision in the subsequent instances. In the second figure 5.3b, search space is big enough to capture the entire collision region but small enough to plan the speed. This is of course a more mature solution than the first one. In such case, vehicle has to take a conservative stance by slowing down or stopping. In the third case shown in the figure 5.3c, search space is even bigger allowing the search algorithm to find the most suitable solution for this situation.

5.1.4 Inconsistent footprints

Footprints of the ego vehicle are calculated along its desired path. Path following controller makes sure that the ego vehicle is travelling on the desired path with a good accuracy which is also one of the key requirements in this work. However, if the deviation of the actual path and the desired path of ego vehicle is significant, then the ego vehicle travels away from the footprint which is used for the collision checking. This inconsistency may result in the collision which may not be detected or predicted by the velocity planning function especially for the system with predefined path (or offline path planning). Consider the figure 5.4 and 5.5 as shown given below.

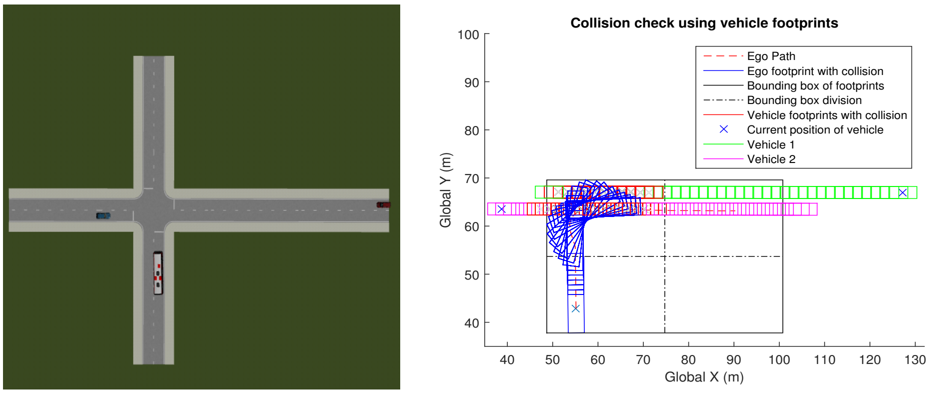


Figure 5.4: Ego vehicle following footprints with precision

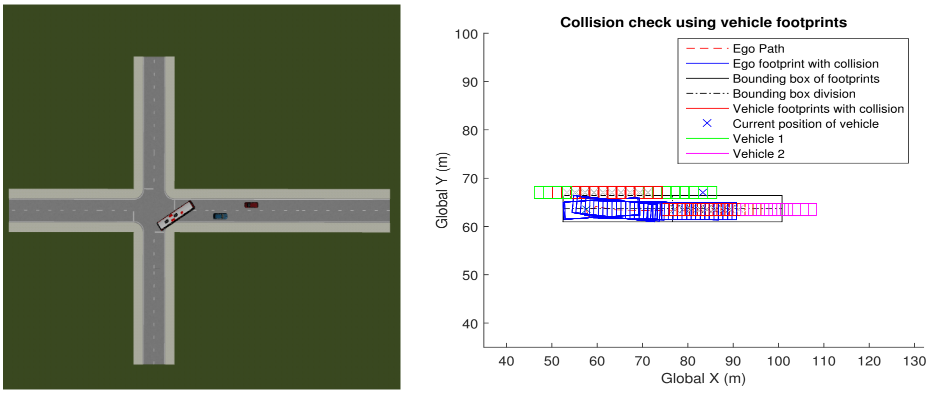


Figure 5.5: Ego vehicle travelling with deviation from the footprints

5.1.5 Lack of feasible solution

Referring back to the figure 5.3 where three different sizes of search space results in three different outcomes of A* search. Second plot in this figure is showing no optimal path implying that no solution exists for this case. Such situations may happen quite often on typical traffic scenarios. In case ego vehicle is made to slow down or stop completely, new solutions are obtained over successive re-planning instances. Lack of solution also takes velocity planning problem into higher level of motion planning where perception of the environment and traffic situation can play important role in generating proper solution. Consider a traffic junction in which ego vehicle is yet to arrive at junction as shown in the figure 4.3. If ego vehicle fails to find optimal solution due to occupied junction, it can re-plan the speed profile in order to stop at the zebra crossing (shown in figure 5.6 by red line). Such planning or re-planning can be carried out with better perception through understanding of environment, traffic scenario, traffic rules, priorities, etc.

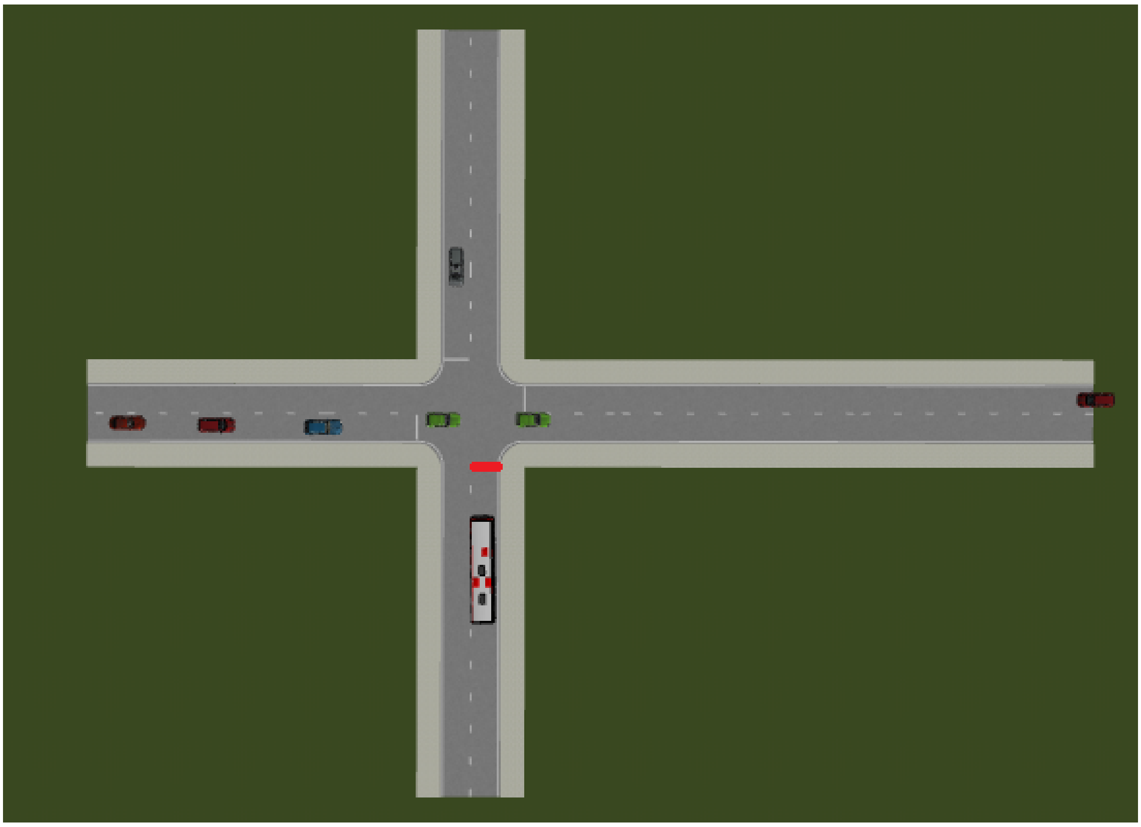


Figure 5.6: Concepts like stop line can be used to plan speed for short distance if long term planning is not possible

There is one more case that needs to be looked at. Refer to the figure 5.7 where optimal path in the space time plot is the only feasible path. Hence, this is a solution that exists on the edge of the set of feasible solutions. Following the speed profile which exists on the edge of feasibility may also create flickering in the successive iterations. For manually driven vehicle, this is a typical situation of a confusion in the traffic negotiation. Usually conservative approach is employed in such scenarios by slowing down before nearest collision. Hence, velocity planning in space time diagram should be influenced by the inputs from the perception in such scenario.

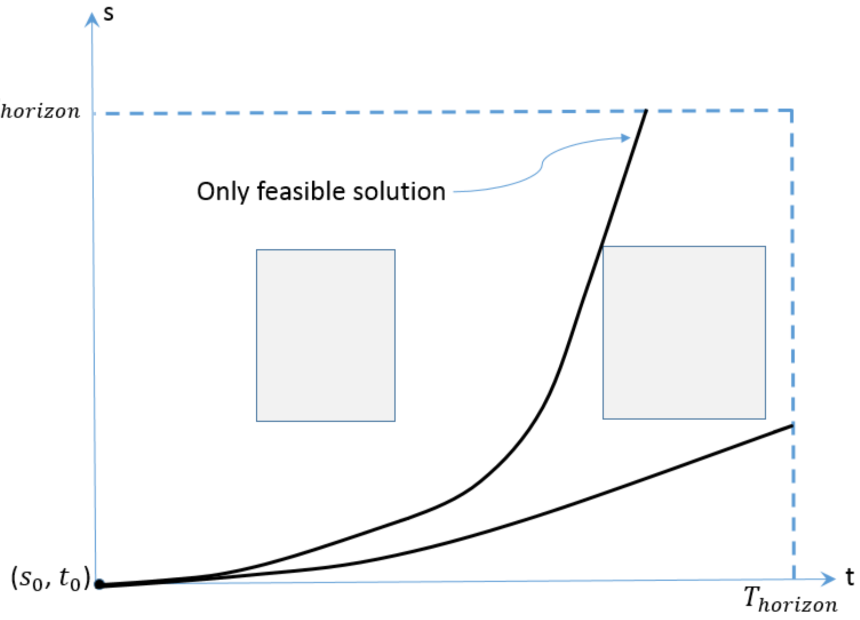


Figure 5.7: In case of only feasible solution, conservative velocity plan should be preferred

5.1.6 Limitations in motion predictions

At this stage, motion prediction of the fellow vehicles is solely based on the CT model. This much of information is clearly not good enough for making a right choice. As already encountered in the figure 4.25, information such as lanes of the other vehicles, road geometry and lane exits can be very helpful in making an informed decision in velocity planning. Hence, performance of velocity planning function will improve over the time with improvement in motion prediction with better sensor fusion and perception of the environment. One optimization problem with the fixed criteria of optimality is not sufficient enough plan motion in all the traffic scenarios. Hence, perception plays very important role not only in understanding the traffic situation but also in deciding right motion planning strategy for a particular motion. Perception can be useful in many ways. Some of the examples are given below:

- Including perception to predict most likely lanelet in which another vehicle may travel
- Branching the other vehicles along all the possible lanelets in the prediction model
- Identifying vehicles irrelevant for motion planning analysis so that the information from such vehicles can be disregard for further processing.

Analysing the behaviour of other vehicles to anticipate their actions and predict their motion is very important part of perception that can improve performance of motion planning in general for autonomous vehicles.

5.1.7 Computational complexity

Even though this approach seems promising in terms of mathematical complexity, there are still factors that significantly affect the computational cost. Complexity analysis of the this velocity planning approach is out of the scope of this project. However, it is possible to identify the factors influencing the computational cost. Computational complexity of this velocity planning function is dependent on -

1. Number of vehicles on the street
2. Size of the search space in the space-time map
3. Constraints on the optimal speed profile

In order to scale up this approach for negotiation against many vehicles, computational cost of collision checking needs to be optimised using a tree structure such as kd tree or quad tree [15]. Use of a tree structure along with various other methods (such as distance checking) together have a good possibility to bring down computational cost for collision checking operation. As far as the search operation for optimal path is concerned, it is not influenced by number of vehicles in the environment since size of search space remains the same.

Availability of the information can also influence the amount of computations system has to do to make an informed choice about speed planning. For example, knowledge of lane layout can be used to easily discard the vehicles travelling in the irrelevant lanes without needing to carry out extensive collision checking by predicting their motion.

5.2 Future work

There are several aspects with the scope of future development and improvements in this work. Apart from the optimisation in computations using a complete Kd tree (or quad-tree) following are the few aspects that need an attention from the future developers.

5.2.1 Discrete speed levels to continuous speed levels

Current way of finding the optimal speed profile in a discrete search space results in some limitations that need to be overcome in the future time. It is possible to increase grid connectivity to achieve large number of speed levels in the speed planning function.

Another possible way to achieve any speed profile could be to replace the speed controller with the acceleration controller. Such controller would require an acceleration reference input. This acceleration input can be generated with the help of acceleration planning instead of velocity planning in the same discrete search space. This kind of acceleration planning requires formulation of the path finding problem and search algorithm accordingly.

Discrete acceleration levels will overcome two important limitations of this approach:

1. Allowing all possible speeds for the ego vehicle
2. Deviation caused by the finite acceleration will not exist any more

5.2.2 Optimisation problem formulation

After generating a space-time map, optimisation (used for speed planning) can be achieved using various methods (ex. A* search) and criteria of optimality (such as travel time, jerk and collision margin from the obstacle). Formation of objective function as well as the optimisation algorithm have a good scope of modification to overcome the current challenges as well as to achieve enhanced performance.

5.2.3 Variable speed limit along the path

Maximum speed limit(to avoid the rolling over and skidding) along a curved path changes along the length of the path. In this work, this speed limit has been kept same at $4m/s$ for entire motion. Therefore if the information of the maximum allowable speed along the path is available in the database then there is a scope of improvement in the current path finding algorithm to generate a speed profile with variable speed limit along the path. This will allow ego vehicle to travel with the maximum possible speed to improve travel time.

5.2.4 Deterministic to probabilistic approach

Even though estimated trajectories may be different from the real trajectories of the obstacles, it will provide reasonable information at a given instance to plan the future motion. As the estimation is also repeated every instant, the estimate will adjust itself in the response to the real time changes in traffic. However, uncertainty in the motion prediction should be included in the analysis while planning the speed. Consider the following figure 5.8. Optimisation problem in such approach can be modified to include the margin to the collision (shown by the arrow in the figure) along with the time as the criterion for the optimality.

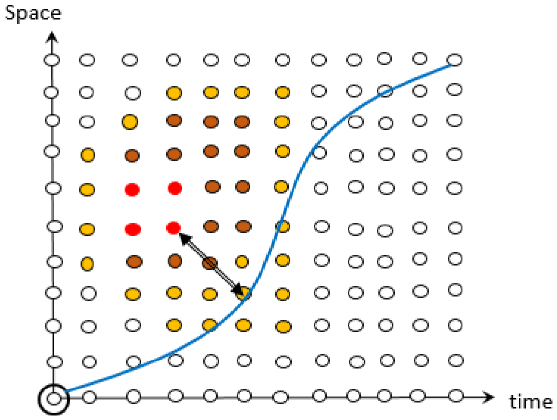


Figure 5.8: Space-time map with the nodes having lesser probability of collisions (lighter colours). Collisions represented by the red colour represent the nodes with highest probability of the collision

6 Conclusion

In this chapter, a brief summary of this master thesis work is presented by revisiting the assumptions and the fundamental constraints. Key achievements and advantages of this approach are highlighted. Conclusions based on various observations that are made during the study are also presented in this chapter. In final remarks, an attempt is made to describe the importance and significance of this approach in the current research related to the autonomous vehicles.

6.1 Summary

This work started with the motivation of trajectory planning for the autonomous vehicles in the urban environment to achieve collision free motion. Trajectory planning for the autonomous vehicle is a very popular and challenging topic in the research community. Approach used in this work is inspired from the work carried out in [5] where the mathematical complexity of the problem is given a significant consideration while achieving general solution to the trajectory planning problem. Proposed approach in [5] is based on decomposition of trajectory planning problem into path planning and velocity planning problem.

Velocity planning approach developed in this work is based on the same idea of decomposition of the trajectory planning problem proposed in [5]. Hence, to start with this approach, the knowledge of the path of the ego vehicle is essential. In longer terms, there should be a separate path planning functionality in the autonomous driving system that will feed pre-calculated path to this velocity planning function to achieve complete online trajectory planning. Furthermore, planning velocity on the known path implies the need of a path following controller. Path following controller has to make sure the desired path is being followed with the speed calculated by the velocity planning function. These are the two main requirements upon which this approach is based on.

Space-time analysis used in this thesis was originally proposed for achieving collision free motion of a point object moving along a known path ([5]). Moreover, the true trajectories of the moving obstacles were assumed to be known. Through this project, We have managed to extend this space-time approach for motion of a 2 dimensional object moving along a known path in a plane while avoiding collision with other 2 dimensional objects moving in the same plane. Since the true trajectories of other vehicles on the street were unknown, corresponding trajectories were estimated using coordinated turn (CT) motion model. Collision check was used in a specific fashion to obtain space-time map. Reduction in the collision checking computations was achieved using various schemes particularly by the use of tree date structure. Collision checking at discrete path positions and future time instances resulted in a discrete space-time map (in the form of a grid).

Velocity planning was achieved with the help of A* search algorithm to find the collision free optimal path in the space-time grid. Time was the only criteria of optimality. It was observed that the grid connectivity in the space-time grid is associated with the speed of the ego vehicle. This property was used in the search algorithm to allow gradual changes in the speed levels while searching for the optimal path. As a proof of concept, five different node connections corresponding to five different speed levels were implemented in the A* search and corresponding optimal speed profile as a function of time was obtained. Sharp transitions in the optimal speed profile are not achievable by the vehicle with finite acceleration. Possible solution to this limitation is proposed in the section 3.4.6 through the modification of grid connectivity.

Simulations were carried out on the 'Scania Omni bus' model in the simulation platform called *PreScan* along with the computations in *Simulink - MATLAB*. This velocity planning function was implemented and simulated for two different types of road geometries i.e. traffic junction and traffic circle. At a traffic junction, right hand turn and straight line motion of the bus (ego vehicle) in the presence of two other vehicles was tested. These two scenarios included merging and intersection maneuvers for the ego vehicle with respect to the other moving vehicles. Also, these experiments were designed to create a '*window of opportunity*' for the ego vehicle (to pass through) which was '*captured and ceased*' by the velocity planning function. Similarly at the traffic circle, ego vehicle managed to merge into the traffic.

Specific limitations and challenges that were encountered during the simulation are presented in the discussion chapter. Future work possibilities related to this approach are also presented in this chapter.

6.2 Key highlights

Proposed approach has been developed by considering some features and requirements (section 2.2). Now in this section, key features of this velocity planning approach will be highlighted to understand how promising this approach is for the future development. Conclusions based on certain observations will be presented as well.

6.2.1 Space-time map for the motion of 2 dimensional object

Approach of velocity planning using space-time map was originally proposed for the velocity planning of a point [5]. In this master thesis project, the method of generating space-time map for 2 dimensional object (ego footprint) is developed.

In [5], true trajectories of the moving obstacles were considered to be known. Proposed method in this project uses motion model for motion prediction along with the real time velocity planning to overcome this limitation.

6.2.2 Reduction in computational cost for collision checking

Proposed method for generation of space-time map uses collision checking in a particular fashion which is computationally very expensive. Method of using tree structure for optimizing computational cost of collision checking makes this approach realisable for real hardware platforms.

6.2.3 Collision avoidance

Collisions on all sides and corners of ego vehicle can be avoided through the proposed approach. Through this velocity planning approach, collision avoidance with the multiple vehicles is achieved which is another essential feature for the autonomous vehicle on the street.

6.2.4 Generalised approach

This approach is very generalised as far as the traffic situation is concerned. Generality of this approach can be understood through following three aspects:

1. Path type of the ego vehicle:
   This approach works well on both the straight line path as well as the curved path
2. Road geometry:
   The Same approach can be used for the velocity planning at various road geometries such as traffic junction(T-junction, single lane junction, multi-lane junction), traffic circle and lane entrance, etc. These are also the most common road geometries in the urban environment.
3. Maneuver:
   This velocity planning approach works well with some of the most common maneuvers for any vehicle in the traffic such as merging, yielding and intersection.

6.2.5 Real time planning

Real time planning is one of the essential features of the autonomous vehicle. Through the simulations results presented in the previous chapter, conclusion can be made about necessity of real time motion planning. The real time motion planning is not only necessary to address rapid changes in the traffic scenarios but also required to overcome the limitations of ‘controller performance’ as well as the ’velocity plan’ itself.

6.2.6 Formulation of optimization problem

Formulation and solving of optimization problem (time optimal) is another challenging step of velocity planning problem after modelling the problem. Time optimal speed profile with the finite speed levels is generated in this work using A* search algorithm. This serves as a proof of a concept of the fact that space-time framework developed in this thesis can be utilised to formulate and solve optimization problems with different optimality criteria using different optimization methods. A* search algorithm is just one of the many methods to formulate and solve this problem.

6.2.7 Grid connectivity

Grid connectivity in the search algorithm has allowed us to generate a time optimal velocity profile with the gradual increase or decrease in speed to achieve feasible solution to the optimization problem. Moreover, limitation of finite acceleration is also addressed with modified grid connectivity. These two examples are good enough to state that despite of limitations of discretized approach, the grid connectivity holds very high potential for addressing and solving many practical issues and challenging problems in such framework.

6.2.8 Possibility of extension

One of the most important features of this approach is the possibility of the extension to the complex traffic scenarios and road geometries. This approach is also very helpful in modelling and analysing complex and critical traffic scenarios through measurable parameters of space and time. This framework itself can serve as tool for development and simulation of velocity planning algorithms by visualizing the performance in space-time map.

Knowing the limitations of current deterministic approach, it should be possible to use a probabilistic motion model to include uncertainty in the approach during the future development as discussed in section 5.2.4. There is also a possibility to formulate the objective function for the optimal speed profile with the criteria of optimality such as acceleration, jerk, margin from the collision along with time. Such possibilities of expansion and modification exist at different stages in this approach making it suitable for a long time development process.

6.3 Final remarks

Through this project, we have successfully put together a framework that can generate a velocity profile to navigate an autonomous vehicle through a traffic junction or a roundabout, or even follow another vehicle. Within this framework, it would be possible to make it better and better over time as issues are resolved. From this stage, other sub-issues can be focused on in the further projects.

In the final remarks, we will try to see this velocity planning function as a subsystem in the bigger picture i.e. autonomous driving system. In order to achieve the full autonomous behaviour, vehicle needs to gather and process lot of information and make informed decision based on it. Since the planning is being done on the several levels based on the information from the localisation in the environment and the perception of the surrounding, there has to be a good scope of interaction between the various sub-functions at different levels. Various subsystems in the autonomous driving system are currently under development and rely on the exchange of information with the other blocks. Therefore it is not just the interaction between the subsystems but also the collective development over significant period of time that should affect the choice of approach. Generality and the possibility of the extension are two very important aspects for any approach to become a part of long term development process. Hence, velocity planning approach developed and proposed in this master thesis is very promising to fit and interact well with the other subsystems and evolve over the time with the entire autonomous driving system.

References

[1] Junqing Wei, John M. Dolan and Bakhtiar Litkouhi; "Autonomous Vehicle Social Behavior for Highway Entrance Ramp Management", "IEEE Intelligent Vehicles Symposium", 2013, pages 201-207

[2] Julius Ziegler, Philipp Bender, Markus Schreiber, Henning Lategahn, Tobias Strauss, Christoph Stiller, Thao Dang, Uwe Franke, Nils Appenrodt, Christoph G. Keller, Eberhard Kaus, Ralf G. Herrtwich, Clemens Rabe, David Pfeiffer, Frank Lindner, Fridtjof Stein, Friedrich Erbs, Markus Enzweiler, Carsten Knöppel, Jochen Hipp, Martin Haueis, Maximilian Trepte, Carsten Brenk, Andreas Tamke, Mohammad Ghanaat, Markus Braun, Armin Joos, Hans Fritz, Horst Mock, Martin Hein, Eberhard Zeeb; "Making Bertha Drive—An Autonomous Journey on a Historic Route", "IEEE Intelligent transportation systems magazinem", 2014, pages 8-20

[3] Junqing Wei, Jarrod M. Snider, Tianyu Gu, John M. Dolan and Bakhtiar Litkouhi; "A Behavioral Planning Framework for Autonomous Driving", "IEEE Intelligent Vehicles Symposium", 2014, pages 458-464

[4] Dong Hun Shin and Sanjiv Singh; "Path generation for the robotic vehicles using composite clothoid segments", "Robotics Institute, Carnegie Mellon University", December-1990

[5] Kamal Kant, Steven W. Zucker; "Towards Efficient Trajectory Planning: The Path-Velocity Decomposition", "The International Journal of Robotics Research", volume 5, 1986, pages 72-89

[6] Peter E. Hart, Nils J. Nilsson, Bertram Raphael; "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", "IEEE Transactions on Systems Science and Cybernetics", Volume 4, Issue 2, July 1968, pages 100-107

[7] Jesse Levinson, Jake Askeland ; Jan Becker; Jennifer Dolson; David Held; Soeren Kammel; J. Zico Kolter; Dirk Langer; Oliver Pink; Vaughan Pratt; Michael Sokolsky; Ganymed Stanek; David Stavens; Alex Teichman; Moritz Werling; Sebastian Thrun; "Towards fully autonomous driving: Systems and algorithms", "Intelligent Vehicles Symposium , IEEE", volume 4, 2011, pages 163-168

[8] "Robotics Institute: NavLab", `Ri.cmu.edu`, 2016. [Online]. Available: `http://www.ri.cmu.edu/research_lab_group_detail.html?lab_id=28`. [Accessed: 02- Jun- 2016].

[9] A. Widodo, T. Hasegawa; "A new inter-vehicle communication system for intelligent transport systems and an autonomous traffic flow simulator", "The International Journal of Robotics Research", Sept. 1998, pages 82-86,

[10] Scania Group, "Innovative Scania: Automatic driving systems pave the way to safer roads - Scania Group", Scania Group, 2016. [Online]. Available: http://www.scania.com/group/en/automatic-driving-systems-pave-the-way-to-safer-roads/. [Accessed: 02- Jun- 2016].

[11] Standford Team, "Welcome | Stanford Autonomous Driving Team", Driving.stanford.edu, 2016. [Online]. Available: http://driving.stanford.edu/. [Accessed: 02- Jun- 2016].

[12] "Google Self-Driving Car Project", 2016. [Online]. Available: http://www.google.com/selfdrivingcar/. [Accessed: 02- Jun- 2016].

[13] "Volvo IntelliSafe - Innovations | Volvo Cars", Volvocars.com, 2016. [Online]. Available: http://www.volvocars.com/us/about/our-innovations/intellisafe. [Accessed: 02- Jun- 2016].

[14] Russell A. Brown, "Journal of Computer Graphics Techniques", "Building a Balanced k-d Tree in $O(kn \log n)$ Time", vol. 4, No. 1, 2015

[15] S. Lambert, "Quick Tip: Use Quad-trees to Detect Likely Collisions in 2D Space", Game Development Envato Tuts+, 2012. [Online]. Available: https://gamedevelopment.tutsplus.com/tutorials/quick-tip-use-quadtrees-to-detect-likely-collisions-in-2d-space\--gamedev-374. [Accessed: 10- May- 2016].

[16] Paul Premakumar; "A* (A Star) search for path planning tutorial - File Exchange - MATLAB Central", Se.mathworks.com, 2016. [Online]. Available: https://se.mathworks.com/matlabcentral/fileexchange/26248-a---a-star--search-for-path-planning-tutorial. [Accessed: 16- May- 2016].

[17] "Scania Omni bus model, 3dwarehouse.sketchup.com, 2016. [Online]. Available: https://3dwarehouse.sketchup.com/model.html?id=u1da436c7-7479-4f10-845d-ade4f2f80753. [Accessed: 22- Feb- 2016].

[18] "Stopping distances on wet and dry roads (Department of Transport and Main Roads)", Tmr.qld.gov.au, 2016. [Online]. Available: http://www.tmr.qld.gov.au/Safety/Driver-guide/Speeding/Stopping-distances/Stopping-distances-on-wet-and-dry-roads.aspx. [Accessed: 22- Aug- 2016].

[19] Stephen Boyd and Lieven Vandenberghe, "Convex Optimization", Cambridge University Press, New York, Edition 7, 2009

[20] Stuart J. Russell and Peter Norvig, "Artificial Intelligence - A Modern Approach", Prentice Hall, New Jersey, 1995

[21] Steven W. Smith, "The Scientist and Engineer's Guide to Digital Signal Processing", California Technical Publishing, San Diego, California, 1998

A

Appendix A

Bus parameters are presented in the table A.1.

Parameter	value	Unit	Description
M	12783	Kg	Unloaded bus mass
J_{zz}	200000	Kgm^2	Inertia of vehicle along Z (along height)
J_{yy}	175010	Kg	Inertia of vehicle along Y (along width)
J_{xx}	11212	Kg	Inertia of vehicle along X (along length)
Length	13.19	m	Length of Bus
Width	2.550	m	Width of Bus
Height	3.310	m	Height of Bus
Wheelbase	7	m	Distance from front to rear wheel
$Ntyre_1$	2	-	Number of tyres on 1^{st} Axle
$Ntyre_2$	4	-	Number of tyres on 2^{nd} Axle
F_{11}	2.795	m	Distance of front wheel to front edge of vehicle
a	4.3786	m	Distance of COG to front wheel
b	2.6214	m	Distance of COG to rear wheel
h_{COG}	0.460	m	COG height from road
R_1	0.507	m	Radius of wheel on second axle
R_2	0.507	m	Radius of wheel on second axle
P_{break}	150	bar	Maximum brake pressure
C_w	0.7	-	Air resistance coefficient
R_2	0.507	m	Radius of wheel on second axle
Cs_{rear}	41400	N/m	Rear suspension system stiffness
Cs_{front}	52151	N/m	Front suspension system stiffness
Ds_{rear}	3624	Ns/m	Rear suspension damping rate
Ds_{front}	4980	Ns/m	Front suspension damping rate
Steering ratio	17.0	-	Steering ratio
δ_{max}	750	degree	Maximum steering wheel angle

Table A.1: Model parameters used for the simulation: Scania Omni Bus