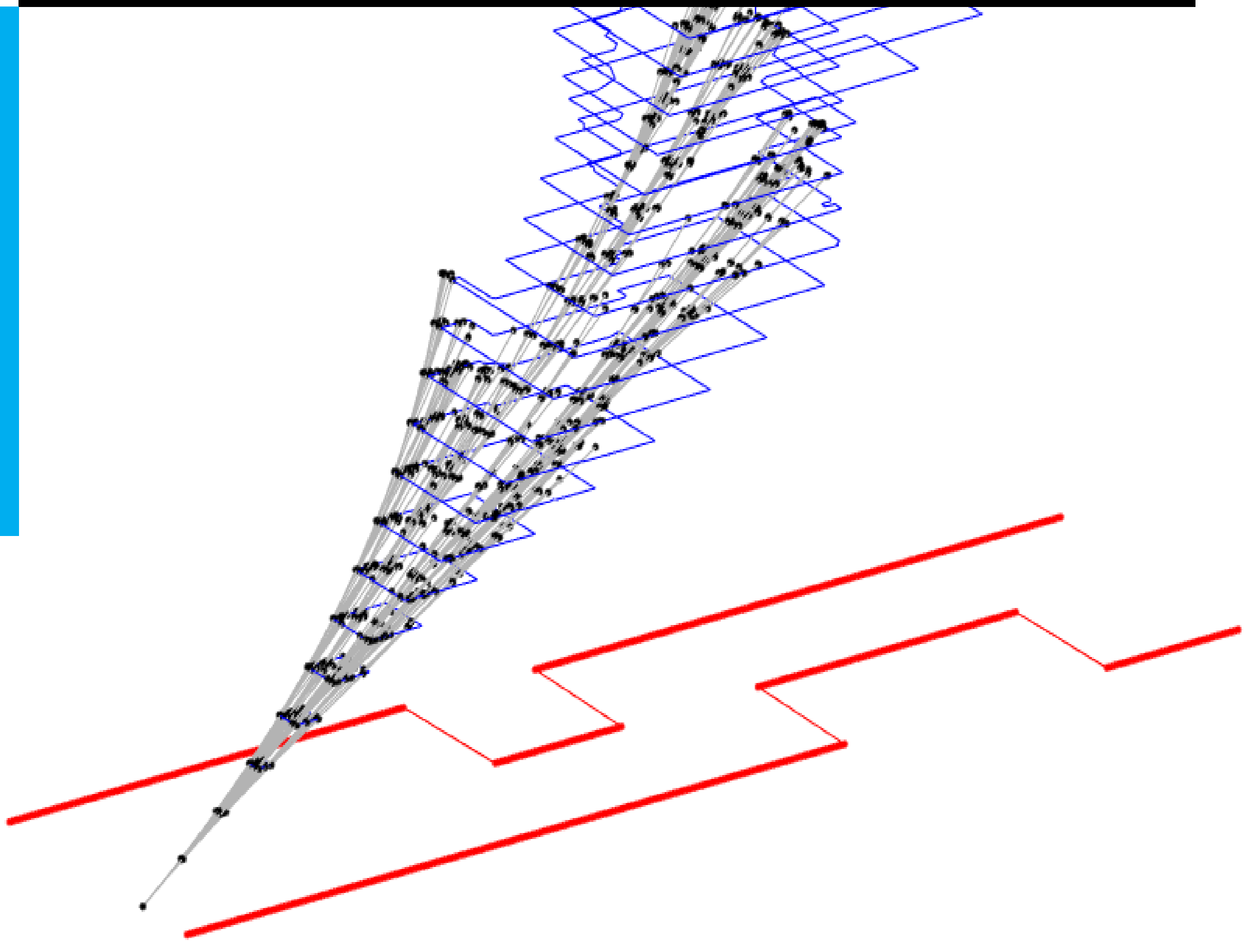
A Metric to Quantify the Hazard Avoidance Capability of Vehicles

Anoosh Anjaneya Hegde

Master of Science Thesis



Delft University of Technology

Cognitive Robotics (CoR)

A Metric to Quantify the Hazard Avoidance Capability of Vehicles

Master of Science Thesis

For the degree of Master of Science in Mechanical Engineering at Delft University of Technology

Anoosh Anjaneya Hegde

August 23, 2019

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Delft University of Technology
Department of
Cognitive Robotics (CoR)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

A Metric to Quantify the Hazard Avoidance Capability of Vehicles

by

Anoosh Anjaneya Hegde

in partial fulfillment of the requirements for the degree of

Master of Science Mechanical Engineering

Dated: August 23, 2019

Supervisor(s): dr. Arturo Tejada Ruiz

Reader(s): dr.ir. Joost C.F. de Winter (Chair)

ir. F.A. Mullakkal-Babu

Abstract

Safety is an important parameter considered during the design of Advanced Driver Assistance Systems and fully autonomous vehicles. One of the ways to assess the road vehicle's safety is by estimating the likelihood with which the vehicle can react to prevent the danger. In the presence of an impending collision (hazard), the trajectory planning module in the autonomous vehicle would generate few escape trajectories to avoid the collision. The escape trajectory is chosen such that it maximises the safety of the vehicle based on certain criteria. One of these criteria is the vehicle's avoidance capability throughout the trajectory.

This thesis presents an avoidance metric that is constructed using a computational procedure to quantify the avoidance capability of the vehicle in both pure longitudinal (1-D) and, combination of both lateral and longitudinal (2-D) scenario. The key idea is (a) Propagate forward in time the current state of the host (and the world) using a vehicle model to estimate the host's reachable set of states. (b) Carefully select a set of samples from the reachable set and repeat the propagation. (c) At every step, the trajectories that lead to collisions are eliminated. The ratio of the size of the region spanned by the remaining trajectories to the size of the region spanned by all the trajectories (including those that lead to collision) then constitute the estimate of the host's avoidance capability.

Through simulations on specific use-cases, for a pure longitudinal motion, on comparison with Brake Threat Number (BTN), it was observed that the metric (from the proposed computational procedure) performs very similar to BTN and also takes a low computational time of 380 [ms] for a time horizon of 2.5 [s]. However, in the presence of dynamic obstacles, major differences in performance (such as discontinuities, step-like variation), were observed between the metric and BTN. In the case of the combination of both lateral and longitudinal motion, the computation time for the proposed procedure was found to be independent of the number of obstacles. To reason about the accuracy of the approximation of the reachable set for a double integrator model obtained from the proposed procedure, it was compared with the nodes obtained from Rapidly-exploring Random Trees (RRT), an under-approximation, and found that the nodes lie either very close or well within the boundary of the approximation. However, the computation time for the proposed procedure took around 10.87 [s] which comes as a major drawback.

Table of Contents

List of Figures

List of Tables

Acknowledgements

The past two years has been an arduous journey filled with ups and downs. Although it was challenging at times, I have had a great time in Delft. This was mainly possible because of people who are thanked here:

1. My supervisor dr. Arturo Tejada Ruiz for his positive attitude and assistance throughout the duration of this thesis.
2. To dr.ir. Joost C.F. de Winter (Chair) and ir. F.A. Mullakkal-Babu for agreeing to be a part of thesis committee.
3. My ammijaan and pithaji for their continued technical support and choosing me as the best one to pursue masters.
4. My gade Anuraag for being the kind soul he is.
5. My cousins, uncles and aunts for their continued support.
6. My PES dosts Mijar, Biradar, Sujai, Darshan, Chiraag, Venkat, Hobo, Swathi for their continued belief and listening to me during tough times.
7. My jigris Vishrut and Arvind with whom I have collaborated and worked on several assignments and also constantly helping me throughout this 2-year journey.
8. My classmates Karan, Nishant, Bhoraskar, Achin, Neel, Arsel, Vishant, Yannick, Lars for helping out whenever needed.
9. My friend Ewoud for all the time spent discussing about motion planning techniques, jobs and life.
10. My nova fam Sanne, Ted, Berend, Koen, Itamar for the time spent in building a bike together.
11. My MavLab fam Nilay, Jelle and Fede for all the fun we had while building algorithms for the autonomous drone.
12. The Rh gang, Akhilesh, Badri, Pandu, Sid, Sushanth, Sharabh, Arya, Vroon, Anurag-Sam, Kishan for all the friday night scenes and also being a part of this 2-year arduous journey.

13. My housemates Jesil, Swagatam, Anton, Flavien, Clement, Shirin, Muhammad, Shreyas for all the good times we had together.

Delft, University of Technology
August 23, 2019

Anoosh Anjaneya Hegde

“Mandiyelo”
— *Anuraag*

Chapter 1

Introduction

According to figures from the European Road Safety Observatory, 30,687 people were killed in road accidents in EU countries in 2011 [3]. That means, around 84 people were killed per day, on European roads. Among these accidents, some could have been prevented by vehicles equipped with state-of-the-art vehicle safety systems.

The European Parliament and the Council passed a regulation [4] concerning road safety, that made it mandatory for all the new vehicles to be equipped with active safety systems (or Advanced Driver Assistance Systems [ADAS]) such as Electronic Stability Control (ESC) and Advanced Emergency Braking Systems [1](AEB). This regulation was greatly responsible for bringing down road fatalities in EU countries from 30,687 in 2011 to 26,132 in 2015 [3].

Autonomous vehicles promise increased safety for passengers, thereby potentially reducing the number of road fatalities. Safety is improved by enforcing that the risk of hazardous event(s) in traffic is kept at a reasonably low level. By requiring minimal human intervention, autonomous vehicles are an extension to the partially automated systems (ADAS). Several challenges arise while making automated driving a reality. One of them is the decision-making process to ensure safety, in every layer of the autonomous vehicle architecture [5].

For instance, the DARPA Urban challenge in 2007 witnessed the autonomous vehicles compete against each other in simple traffic environments. With 11 participants in the final, only six managed to complete the course. One of the interesting things that happened at the challenge was the low speed collision between MIT's Talos and Cornell's Skynet. According to [6], one of the reasons for the collision was an over-importance on lane constraints (boundaries) versus vehicle proximity, in motion planning.

In the DARPA collision scenario, both exiting the lane boundaries and closing the distance between the vehicles can be seen as manoeuvres that lead to hazardous events. However, on assessing the risk for both the manoeuvres, the autonomous vehicles (both Talos and Skynet) decided to collide as it had a lower risk of hazard, compared to the manoeuvre exiting the lane boundaries.

[1]Vehicle safety systems such as AEB assist the driver by reacting early to 'risky' situations to reduce the severity of the damage, the vehicle would potentially suffer in the future.

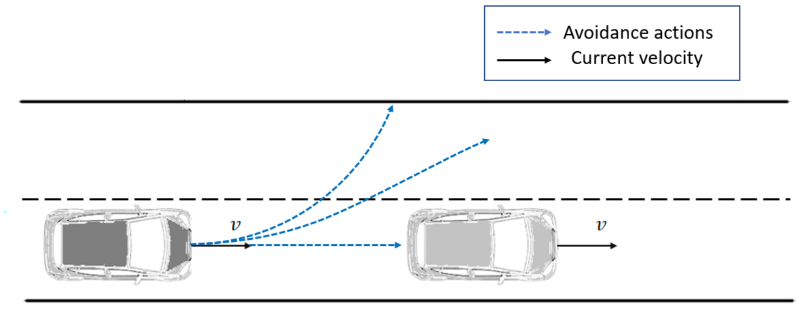


Figure 1-1: Forward collision scenario where both the host (left) and the obstacle (right) vehicle are moving at the same velocity from left to right. Blue dashed lines: Various avoidance actions that can be executed by the host vehicle to avoid the hazard.

The manoeuvre exiting the lane boundaries has higher risk as the damage that the vehicle might endure in the future is higher (user-defined in case of exiting the boundaries) than the manoeuvre closing the distance, implying an over-importance placed on lane constraints (boundaries) versus vehicle proximity.

Therefore, in the case of autonomous vehicles one of the necessary requirements in the decision making process is a better assessment of the risk of a hazardous event (unlike the aforementioned scenario where the assessment is based only on the damage that the vehicle might endure in the future).

1-1 Motivation

One of the better ways to assess the risk of hazardous event(s) is by estimating the likelihood and severity of the damage that the vehicle might endure in the future, and the likelihood with which the vehicle can act to prevent the damage [7].

For example, consider a hazard of forward collision of an autonomous host vehicle[2] (left) with the obstacle vehicle (right) as illustrated in Figure 1-1. Both the vehicles are travelling at the same speed from left to right. Although this situation does not seem dangerous at first there is always a non-zero chance that the obstacle vehicle might trail brake, potentially leading to a rear-end collision. In such a case, the trajectory planner in the autonomous host vehicle would be expected to generate few escape trajectories (avoidance actions) to prevent the occurrence of the collision and select the best one based on certain criteria.

As remarked in [8], one of the ways to select the best escape trajectory would be to add a safety margin ($\approx$ 1 [m] from [9]) to the escape trajectories and check for the ones that avoid collision with the obstacles present in the road scenarios. However, this way to evaluate trajectories become overly conservative in typical road scenarios.

Alternatively, an escape trajectory with the least cost can be chosen based on a cost function. The cost function incorporates elements such as distance between the predicted states of the

[2] Vehicle under consideration.

host and obstacle vehicle in the future, comfort based on the maximum accelerations, jerk experienced along the trajectory, distance from the lane centre [10].

In order to execute the escape trajectory with the least cost, a set of control commands need to be applied to the host vehicle. However, if the chosen escape trajectory is very sensitive i.e. if the control commands are not tracked accurately, the resulting trajectory will deviate from the escape trajectory potentially ending up in collision region. In spite of accurate tracking, the uncertainties in the prediction of the states of the obstacle vehicle or actuator saturation can also lead to a collision. Therefore, the vehicle's capability to execute the trajectory such that it avoids the collision needs to be developed.

However, in order to incorporate the avoidance capability of the host vehicle in the cost function, a measurable (computable) notion of vehicle avoidance capability needs to be developed. This is the goal of this thesis.

The computation of the avoidance capability of the autonomous host vehicle, however, depends on factors such as the type of environment (static or dynamic obstacles), host and obstacle vehicle acceleration limits and the system dynamics. Loosely speaking, in pure longitudinal scenarios, threat measures such as Brake Threat Number (BTN) can be used to quantify the avoidance capability of the vehicle since it measures the fraction of the total vehicle potential. In the presence of obstacles, however, it assumes a perfect knowledge about the future state of the obstacle vehicle which is seldom true. Also, in a scenario considering both lateral and longitudinal dynamics, there does not exist a computation procedure to compute the avoidance capability of the host vehicle. The possibility of collision implies very low avoidance capability of the host vehicle along the escape trajectory with the least cost. Therefore, avoidance capability of the vehicle is an imperative element that needs to be included in the assessment of the escape trajectories. Developing one such method of avoidance capability estimation is the goal of this thesis.

To summarise, the main question can be posed as follows:

Research Question:

1. Is it possible to describe the ability of a vehicle to avoid a collision?
2. If so, how should the avoidance capability be estimated for both longitudinal (1-D) and lateral and longitudinal (2-D) manoeuvres?

1-2 Contributions

The main contributions of this thesis is an avoidance metric to quantify the avoidance capability of the vehicle by means of computational procedures for both pure longitudinal (1-D) manoeuvres and, lateral and longitudinal (2-D) manoeuvres.

The key contributions of this thesis are 1) Development of a computational method to approximate the host's avoidance capability using reachability analysis. 2) Systematic comparison of the results with the estimators (and approximation of the estimators) found in the literature.

1-3 Outline

The rest of the thesis is organised as follows. Chapter 2 begins by listing the ideal properties of the avoidance capability estimator and reviews existing methods in the literature that could be used to approximate such estimator. Among these methods are Brake Threat Number (BTN), constant control for fixed horizon and motion primitives.

Chapter 3 introduces the concept of reachability and proposes computation procedures to approximate the reachable set of a vehicle model with double integrator system dynamics for both pure longitudinal manoeuvres and, combination of lateral and longitudinal manoeuvres. The approximation of the reachable sets are then used to develop an avoidance capability estimator.

Chapter 4 elaborates on the implementation of the computational procedures described in Chapter 3 for certain scenarios. The avoidance metric for the pure longitudinal manoeuvres, based on the proposed computational procedure is compared with BTN via simulations, in both static and dynamic scenarios. In case of the combination of lateral and longitudinal manoeuvres, the avoidance metric is computed and the accuracy of the computation procedure is compared with Rapidly-exploring Random Trees (RRT) via simulations in a static scenario.

Chapter 5 draws upon this thesis by making some conclusions and also remarks about future extensions of this work.

Chapter 2

Preliminaries

An intuitive way to estimate the capability of the vehicle to avoid a hazard is to estimate 'how many' escape manoeuvres it can possibly execute.

This chapter begins by giving a set of ideal properties for a metric and then proceeds to list candidate metrics for longitudinal manoeuvres only. Later, for a combination of lateral and longitudinal manoeuvres, metrics based on existing methods from the literature are derived.

2-1 Ideal Properties of an Avoidance capability metric:

Ideally, an estimator of a vehicle's avoidance capabilities should have the following properties:

1. *Bounded Range*: The lower limit of the range means that the vehicle is about to crash whereas the higher limit means that the vehicle has a maximum number of options (trajectories to follow) to avoid hazards.

2. *Continuity*: The metric should be continuous and a function of the vehicle properties such as velocity, acceleration, etc,. and also time.

3. *Constraint-dependent*: The metric should also account for the road boundaries and obstacle vehicles throughout the collision-free path.

4. *Low computational complexity*: In order for the metric to be implemented real-time, the number of storage spaces and/or the amount of time required to compute the metric must not be significant.

5. *Repeatability:* For a given scenario (or environment) and an initial state of the vehicle, the metric should always return the same value from its computation procedure.

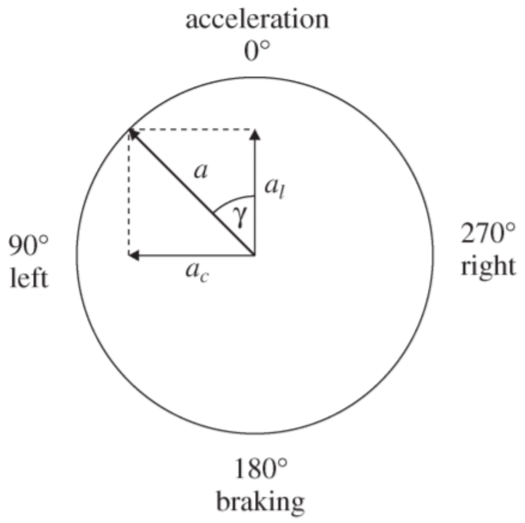


Figure 2-1: Kamm's circle [1]

2-2 Avoidance capability metrics in the Literature

In this section, methods from the literature are reviewed and shown how they can be adapted to derive metrics to quantify the avoidance capability.

2-2-1 Risk Metric - BTN

Currently available vehicular emergency support systems, such as automated emergency braking (AEB) systems are generally activated when a particular 'threat' metric exceeds a prescribed limit. Two popular such 'threat' metrics are Brake Threat Number (BTN) and Steering Threat Number (STN).

Such metrics measure the threat levels by computing the fraction of the total vehicle potential (acceleration limits of the vehicle) based on the minimum acceleration required to avoid the collision hazard. Loosely speaking, this fraction translates to a sort of avoidance capability of the vehicle. Therefore, BTN, STN can be treated as suitable candidates to quantify the avoidance capability of the vehicle in their respective dimensions (longitudinal and lateral respectively).

Furthermore, for 1D manoeuvres, only longitudinal motion (BTN) is studied in this thesis since the same analysis can be adapted to the lateral motion (STN)[1].

In order to compute BTN, the minimum acceleration required to avoid the hazard needs to be obtained. These accelerations, however, must always be within the vehicle limits, which are given by the Kamm's circle as illustrated in the Figure 2-1.

[1]Refer [11], [12] for the computation procedure of STN.

The maximum acceleration (a_{max}) as shown in Figure 2-1, is given by :

$$a_{max} = \mu_0 g \tag{2-1}$$

$$(a_l)^2 + (a_c)^2 \leq \mu_0 g \tag{2-2}$$

$$\gamma = atan\left(\frac{a_c}{a_l}\right) \tag{2-3}$$

where, μ_0 is the static friction coefficient, a_l, a_c are the longitudinal and lateral components of acceleration, respectively, and g is the acceleration due to gravity.

Brake Threat Number (BTN)

BTN describes the fraction of the maximum braking potential[2] required by the host vehicle to avoid a collision hazard. BTN can be calculated from equation 2-4.

$$BTN = \frac{a_{l,req}}{a_{l,max}} \tag{2-4}$$

where, $a_{l,req}$, is the required acceleration to avoid the collision and $a_{l,max}$ is the maximum braking acceleration of the host vehicle.

The following subsections show how to compute $a_{l,req}$ under various cases based on certain assumptions on the host and the obstacle vehicles. This acceleration required is then substituted in equation 2-4 to obtain BTN.

Static obstacle

The first use case is a road scenario where a static obstacle (for example road-works, stationary obstacle vehicle) is in the way of a host vehicle. This use case is a typical road-scene encountered by vehicles in traffic. Consider the scenario shown in Figure 2-2. The following assumptions are made for this scenario:

1. The obstacle is considered to be static (non-moving).
2. The required acceleration is calculated by considering a point mass vehicle model.
3. The required acceleration to avoid the hazard for the host vehicle can be generated instantaneously.

In this scenario, a static obstacle is in the way of the host vehicle. In order to avoid a collision, the host vehicle needs to decelerate such that it stops before the obstacle. Also, due to bounds on the maximum deceleration there exists a critical distance (between the host and obstacle), after which the collision cannot be avoided. This critical distance, x_o for braking can be found based on the maximum deceleration available to the vehicle as given in equation 2-5.

$$x_o = -\frac{v_h^2}{2a_{l,max}} \tag{2-5}$$

[2]maximum deceleration of the host vehicle.

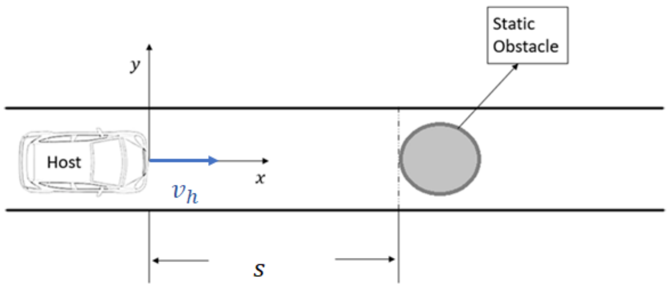


Figure 2-2: Static obstacle scenario used to compute BTN. A static obstacle is present in front of a host vehicle, which is travelling at an initial speed, v_h and s, is the distance between the host vehicle and the static obstacle.

where, v_h is the initial velocity of the host, and $a_{l,max}$, is the maximum longitudinal acceleration of the vehicle with, $a_{l,max} < 0$.

If the distance between host and obstacle, s is greater than critical distance x_o, then the acceleration ($a_{l,req}$) required to stop *just before* the vehicle is given by equation 2-6.

$$a_{l,req} = -\frac{v_h^2}{2s} \tag{2-6}$$

where, s is the distance between the host and obstacle.

Therefore, if the critical distance is not reached, the host vehicle can apply a longitudinal acceleration in the set $a_{l,h} \in [a_{l,max}\,, a_{l,req}]$ to evade the collision with the static obstacle. But, if the critical distance is reached, the required acceleration to avoid the collision can still be obtained from equation 2-6. However, this required acceleration would be greater than the maximum acceleration limits of the host vehicle, indicating that the host vehicle can no longer evade the collision.

Dynamic obstacle with prediction

The second use case is a road scenario where a dynamic obstacle (moving obstacle vehicles) is in the way of a host vehicle. This use case is a typical road-scene encountered by (host) vehicles in traffic.

Consider the scenario shown in Figure 2-3. The figure shows the scenario at two different time steps, t_0, t_1. The following assumptions are made for this scenario:

1. The obstacle is assumed to move with constant acceleration.
2. The obstacle always has a non-negative speed. If it comes to rest, it remains at rest.
3. The required acceleration for the host vehicle to evade the hazard is calculated by considering a point mass vehicle model.
4. The required acceleration for the host vehicle to avoid the hazard can be generated instantaneously and is also of a constant value.

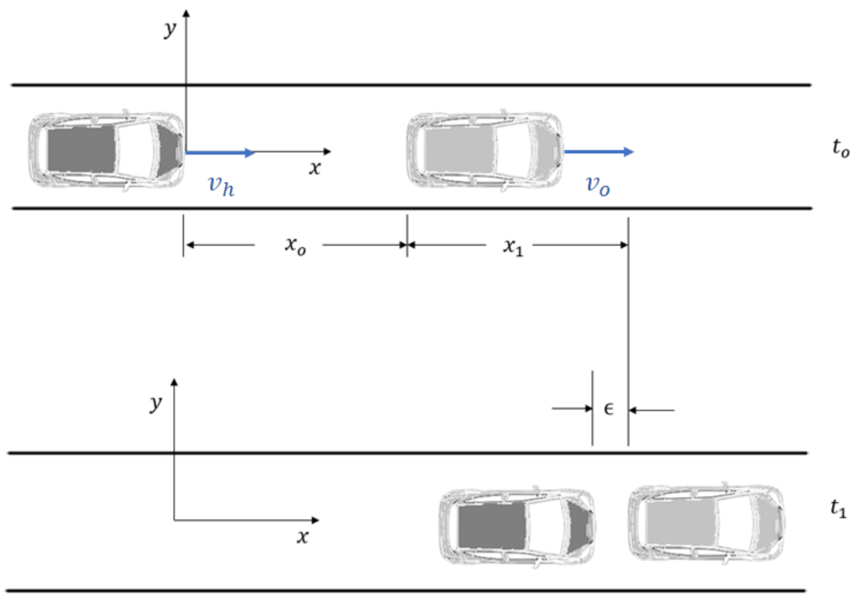


Figure 2-3: Dynamic obstacle scenario used to compute BTN. The top scene shows the configuration of the host and obstacle vehicle at time, t_o while the bottom scene shows the configuration at time, t_1.

In this thesis, it is assumed that, there exists a safety margin, ϵ between the obstacle and host vehicle when they are in contact. Therefore, a collision is declared when the distance between the host and obstacle vehicle is lower than ϵ.

In the dynamic scenario, *only* two cases needed to be considered [13] to compute the required accelerations assuming the initial conditions of the scenario and motion assumptions of both the vehicles. The first case considers the case where the obstacle vehicle comes to rest before coming in contact with the host vehicle. The second case considers the case where the obstacle vehicle comes in touch with the host vehicle before the obstacle vehicle comes to rest. These two case are described subsequently.

Case1: The obstacle vehicle stops before coming in contact with the host vehicle.

At time t_o, the obstacle vehicle begins to brake with acceleration $a_{l,o}$ (from the prediction of the obstacle's future motion). Then, just like in the static obstacle scenario there exists a critical distance, x_o after which the host vehicle cannot avoid the collision with the obstacle.

The distance travelled by the obstacle vehicle during braking is given by equation 2-7

$$x_1 = -\frac{v_o^2}{2a_{l,o}} \tag{2-7}$$

whereas, the distance travelled by the host is given by equation 2-8.

$$x_1 + x_o - \epsilon = -\frac{v_h^2}{2a_{l,max,h}} \tag{2-8}$$

where, v_o is the initial velocity of the obstacle vehicle, v_h is the initial velocity of the host vehicle , $a_{l,max,h}$ is the maximum acceleration of the host vehicle with $a_{l,max,h} < 0$, and $a_{l,o}$ is the acceleration of the obstacle vehicle with $a_{l,o} < 0$.

Then, the critical distance (x_0) can be found out by rearranging equations 2-7, 2-8 as shown in equation 2-9.

$$x_o = -\frac{v_h^2}{2a_{l,max,h}} + \frac{v_o^2}{2a_{l,o}} + \epsilon \tag{2-9}$$

Distance between host and obstacle is greater than critical distance - In case the host is well above the critical distance, the acceleration required to stop the host before the obstacle vehicle with some safety margin (ϵ) is given by equation 2-10 (also obtained in [11]).

$$a_{l,req,h} = -\frac{v_h^2}{2(s - \epsilon - \frac{v_o^2}{2a_{l,o}})} \tag{2-10}$$

where, s is the distance between the host and obstacle.

Therefore, if the critical distance is not reached the host vehicle can apply a longitudinal acceleration in the set, $a_{l,h} \in [a_{l,max,h}\, , \, a_{l,req,h}]$, to evade the collision with the dynamic obstacle in the event, when the obstacle vehicle stops *before* coming in contact with the host vehicle.

The time taken for the obstacle vehicle to stop (TTS) is given in equation 2-11, whereas the time taken for the host vehicle to come in contact (TTT[3]) is given in equation 2-12.

$$TTS = -\frac{v_o}{a_{l,o}} \tag{2-11}$$

$$TTT = \frac{2(s - \epsilon - \frac{v_o^2}{2a_{l,0}})}{v_h} \tag{2-12}$$

where, TTS < TTT and $a_{l,o} < 0$.

But, if the critical distance is reached, the required acceleration to avoid the collision can still be obtained from equation 2-10. However, this required acceleration would be greater than the maximum acceleration limits of the host vehicle, indicating that the host vehicle can no longer evade the collision.

Case2: The obstacle vehicle either does not stop or stops after coming in contact with the host vehicle. Just like the previous case, there exits a critical distance (x_o) after which the host vehicle cannot avoid the collision with the obstacle. This critical distance can be obtained based on the concept of relative velocity. In such a case,

$$x_{ho} = -\frac{v_{ho}^2}{2a_{ho}} \tag{2-13}$$

where, $x_{ho}\, (= s - \epsilon)$ is the distance between the host and the obstacle vehicle, v_{ho} is the initial velocity of the host vehicle with respect to the obstacle vehicle and a_{ho} is the acceleration of the host vehicle with respect to the obstacle vehicle.

[3]Time-to Touch(TTT).

Then the critical distance (x_o) can be found out as shown in equation 2-14.

$$x_o = -\frac{v_{ho}^2}{2a_{ho}} + \epsilon \tag{2-14}$$

Distance between host and obstacle is greater than critical distance - In case the host is well above the critical distance, the acceleration required to stop the host before the obstacle vehicle with some safety margin (ϵ) is given by equation 2-15.

$$a_{l,req,ho} = -\frac{v_{ho}^2}{2x_{ho}} \tag{2-15}$$

On rearranging equation 2-15, the required acceleration for the host vehicle to avoid collision can be obtained as given in equation 2-16,

$$a_{l,req,h} = a_{l,o} - \frac{(v_h - v_o)^2}{2(s - \epsilon)} \tag{2-16}$$

whereas, TTT can be obtained from equation 2-17.

$$TTT = \frac{2(s - \epsilon)}{v_h - v_o} \tag{2-17}$$

where, $v_h > v_o$.
Therefore, if the critical distance is not reached the host vehicle can apply a longitudinal acceleration in the set, $a_{l,h} \in [a_{l,max,h}\, , a_{l,req,h}]$, to evade the collision with the dynamic obstacle in the event, when the obstacle vehicle stops *before* coming in contact with the host vehicle.

■

Based on the state of the host and the obstacle vehicle, the required accelerations obtained either from equation 2-6, 2-10 or equation 2-16 is substituted in equation 2-4 to compute the BTN.

So far this chapter has focused on quantifying the avoidance actions by computing BTN. This process entailed the computation of the required acceleration, a_l to avoid the hazard for the whole time horizon[4]. Note that the required acceleration obtained to avoid the hazard is of a constant value.

However, on considering both lateral and longitudinal dimensions, there does not exists one such analytic way to compute the required acceleration to avoid the hazard.

Therefore, in the next section, a metric is constructed by employing a simple method to obtain the required acceleration(s) (*constant*), $\vec{a}_{lc} = \vec{a}_l + \vec{a}_c$, to avoid the hazard in a two dimensional scenario (both lateral and longitudinal).

[4]By assuming a constant acceleration model for the obstacle vehicle.

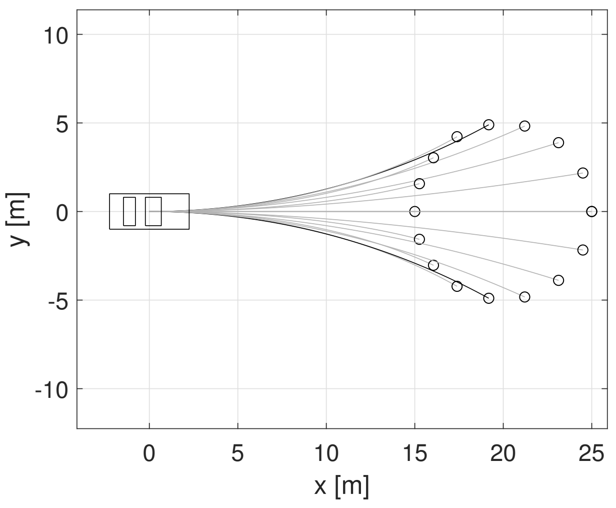


Figure 2-4: Examples of projected trajectories of a point mass vehicle model generated by constant control over a finite horizon for $v_{x,init} = 20\,[m/s]$, $a_{max} = 10\,[m/s^2]$ over a time period of 1 [s]. The thick black line on either sides represent the pure lateral acceleration, a_c. [1]

2-2-2 Metric based on constant control for fixed horizon

This section initially describes a method proposed in [1] to construct a set of trajectories for a given time horizon based on the host vehicle, and then check for collision (Refer to Appendix A for the collision check procedure) between the host (given by the trajectories) and the obstacle's position (prediction of obstacle's position, if dynamic). Next, the set of trajectories that do not collide, is utilised to construct a simple avoidance metric.

The set of trajectories are constructed by varying the angle γ (as shown in Figure 2-1) from $0°$ to $360°$ in the steps of $22.5°$ to obtain a set of lateral ($a_{c,h}$) and longitudinal acceleration ($a_{l,h}$)[5]. The maximum lateral ($a_{c,h}$) and longitudinal ($a_{l,h}$) accelerations are obtained (in the inertial frame) from equations 2-18, 2-19.

$$a_{c,h} = a_{max}\, cos(\gamma) \tag{2-18}$$

$$a_{l,h} = a_{max}\, sin(\gamma) \tag{2-19}$$

These accelerations are applied on a point-mass model (Refer Appendix C) for a given time horizon to obtain the trajectories shown in Figure 2-4.

A simple extension to the method proposed by [1] would be to sample the *inside* of the Kamm's circle (and not just the boundary) with a given discretisation interval[6], a_{int}. Then, just as described above, to obtain the collision-free trajectories, the sampled accelerations can be applied to the point mass model, to obtain the trajectories which can be later checked for collision with the obstacle's position (prediction of obstacle's position, if dynamic).

Based on these trajectories, a simple avoidance metric can be defined as given in equation 2-20.

$$M_{cc} = \frac{n_1}{n_t} \tag{2-20}$$

[5] Control actions for the host vehicle (Appendix C describes the vehicle model).
[6] Determinitic sampling [2].

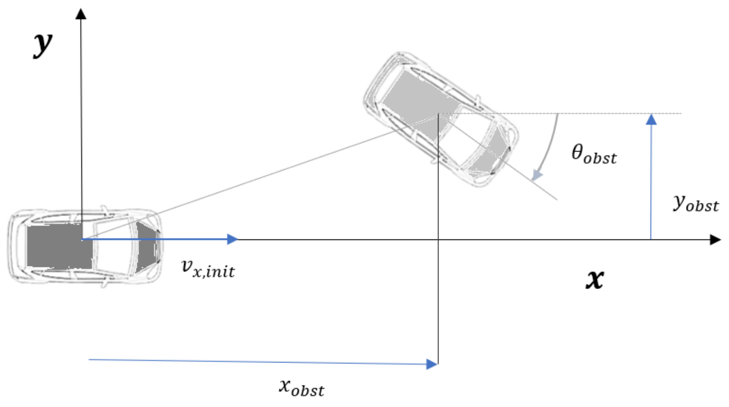


Figure 2-5: Scenario considered to compute the metric based on constant control for fixed horizon. The parameters for the scenario are listed out in table 2-1.

Parameters	**Value**
Sampling discretisation interval for acceleration, $\mathbf{a_{int}}$	$0.5\,[m/s^2]$
Maximum (and Minimum) acceleration, $\lvert\mathbf{a_{max}}\rvert$	$10\,[m/s^2]$
Time horizon, $\mathbf{T}$	1 [s]
Obstacle distance in x, $\mathbf{x_{obst}}$	21 [m]
Obstacle distance in y, $\mathbf{y_{obst}}$	3 [m]
Obstacle orientation, $\theta_{\mathbf{obst}}$	$-\pi/4$ [rad]
Obstacle initial velocity, $\mathbf{v_{x,init,obs}}$	$0\,[m/s]$
Host Initial Velocity, $\mathbf{v_{x,init}}$	$20\,[m/s]$

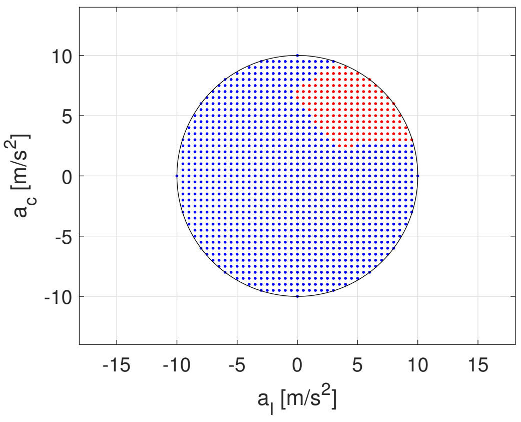
Table 2-1: Parameters for the scenario shown in Figure 2-6.

where, M_{cc} is the avoidance metric, n_1 is the number of trajectories that avoid the hazard, n_t is the total number of trajectories.

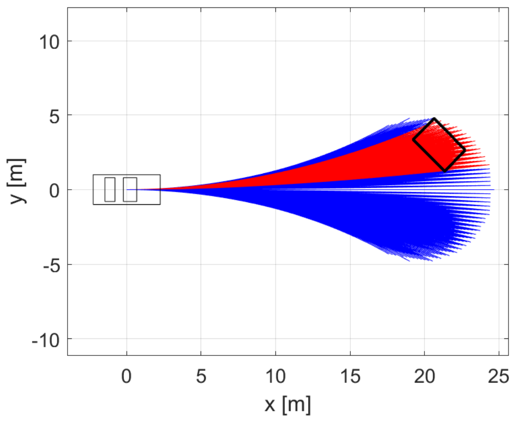
For example, consider a scenario as illustrated in Figure 2-5 where, the host vehicle has an initial velocity, $v_{x,init} = 20\,[m/s]$ and maximum acceleration, $a_{max} = 10\,[m/s^2]$. The obstacle vehicle is at rest, and at a distance 21 $[m]$ from the obstacle and at an angle, $-\pi/4$, $[rad]$ from the host's X-axis in a counter-clockwise sense. The parameters of the scenario, host and the obstacle vehicle are listed out in the table 2-1.

Figure 2-6a shows the sampled accelerations with a given discretisation interval, a_{int}. The accelerations that correspond to collision are in red while the ones that are collision free are in blue. Figure 2-6b shows the trajectories, as a result of applying sampled acceleration on the point-mass model. These trajectories are further checked for collision, by checking if any of the points on the trajectory (the host vehicle is assumed to be point-mass) enter the obstacle boundary.

On using equation 2-20, the metric took a total computation time of 0.377 [s] on MATLAB which was run on a computer with 16GB RAM and 2.8GHz processor Core i7 (7th Gen).



(a) Sampled Kamm's circle.



(b) Examples of projected trajectories of a point mass vehicle model generated by constant control over a finite horizon for $v_{x,init} = 20\,[m/s]$, $a_{max} = 10\,[m/s^2]$ over a time period of 1 [s].

Figure 2-6: (a) Red: Accelerations that result in Collision trajectories. Red: Accelerations that result in Collision-free trajectories. (b) Thick Black lines: Obstacle vehicle boundary. Red: Collision Trajectories. Blue: Collision-free trajectories.

The results presented so far are all based on the use of constant accelerations while performing a collision avoidance manoeuvre (see, e.g., equation 2-4 and 2-20. However, collisions can also be avoided by applying time-varying acceleration commands to the host vehicle. Such manoeuvres cannot be taken into account with the methods described so far.

Such manoeuvres, however, can be taken into account by using the exhaustive search method described next.

2-2-3 Metric from an exhaustive search method based on motion primitives

Motion primitives are the trajectories obtained by applying constant control actions (constant acceleration, $\vec{a}_{lc}$) to the point-mass model over a fixed time interval, Δt for a given set of initial states.

This method [2], comprises of a multi-stage process (n stages). At every stage, the vehicle initials states are propagated forward in time using constant acceleration input on a point mass model. The final states from one stage become the initial states for the next stage.

At the end of n stages, the set of piece-wise constant acceleration that correspond to collision-free motion primitives are obtained. It is to be noted that, at the end of every stage, the motion primitives are checked for collision (Refer to Appendix A for the collision check procedure).

The control actions to be applied, are obtained by sampling the input space (acceleration, $\vec{a}_{lc}$) with a given discretisation interval, a_{int} as shown in Figure 2-6a. Also, the time horizon, T is discretised into n steps of time interval, Δt. Over every Δt, the control action (acceleration) is constant.

Figure 2-7 shows the recursive application of motion primitives based on the procedure described above. The region shown in red is the obstacle region. The black dots correspond to

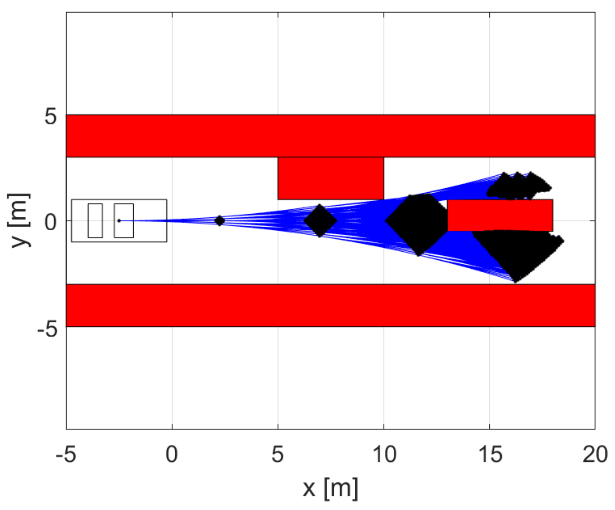


Figure 2-7: Motion primitives generated for $v_{x,init} = 25\,[m/s]$, $a_{max} = 10\,[m/s^2]$ for a time horizon of 0.8 [s]. Four stages of recursive application of motion primitives. Red: Obstacle region. Black dot: Initial/End state of a motion primitive.

the initial or the end state of a motion primitive. The parameters of the scenario shown in Figure 2-7 is given in table 2-2. It is to be noted that, in this scenario, every motion primitive comprises of 1000 points[7].

Based on these motion primitives, an avoidance metric can be defined (on similar lines to the metric based on constant control) as given in equation 2-21.

$$M_{mp} = \frac{n_1}{n_t} \tag{2-21}$$

where, M_{mp} is the avoidance metric, n_1 is the number of trajectories that avoid the hazard, n_t is the total number of trajectories. n_1 can be obtained by simply counting the number of collision free states at the last stage, n. The total number of trajectories, n_t can be obtained from equation 2-22.

$$n_t = 13^n \tag{2-22}$$

The computation of the motion primitives for the scenario shown in Figure 2-7 took a total computation time of around 16.7 [s] on MATLAB.

Scalability issue: On sampling the input space with a sampling interval, $a_{int} = 5[m/s^2]$, it results in 13 combinations of accelerations (lateral and longitudinal). And, therefore at the end of every stage there exists, 13^n motion primitives. Furthermore, every motion primitive has 1000 points for which a collision-check needs to be performed. This results in a serious drawback (intractability) since the computation time increases exponentially with an increase in the number of stages, n. This is important since on increasing the number of stages, the computed allowable input commands approach the actual allowable input commands to avoid the hazard (*static scenario*) for the considered time horizon, thereby, increasing the accuracy of the metric.

[7]Every trajectory connecting an initial state (at stage, i) to a forward propagated state (at stage, $i+1$) has a total of 1000 states.

Parameters	**Value**
Sampling discretisation interval for acceleration, $\mathbf{a_{int}}$	$5\,[m/s^2]$
Maximum (and Minimum) acceleration, $\lvert\mathbf{a_{max}}\rvert$	$10\,[m/s^2]$
Number of time steps, $\mathbf{n}$	4 [-]
Time-step, $\mathbf{\Delta t}$	0.2 [s]
Time horizon, $\mathbf{T}$	0.8 [s]
Host Initial Velocity, $\mathbf{v_{x,init}}$	$25\,[m/s]$

Table 2-2: Parameters for the scenario shown in Figure 2-7.

■

This chapter began by describing the ideal properties of a metric. It was followed by introducing metrics to quantify the avoidance capability in the presence of a hazard. Avoidance metrics such as BTN and the metric based on constant control for fixed horizon took low computation time, but the computation procedure of the metrics only considered the constant acceleration required to avoid the hazard and not the time-varying acceleration signals, which comes as a limitation. However, the avoidance metric from the exhaustive search, that did account for varying acceleration signals was found to be computationally expensive. Some of these shortcomings will be addressed in the following chapter.

Chapter 3

Proposed Metrics

In the previous chapter, the limitation of trying to explore the allowable input space versus the complete input space as a measure of the capability of the vehicle is that it was computationally hard.

Therefore, in this chapter, another principle is proposed. This principle estimates the vehicle avoidance capability based on the space of vehicle trajectories instead of the space of control inputs.

The main goal of this chapter is to develop a metric, based *not* on the available input commands you can give the vehicle to avoid an accident but, based on the positions the vehicle can actually reach (output reachability).

The metric is constructed by first computing a ratio of the size of the output reachable set to the reachable set without considering any obstacle, at future time-steps and then, taking a weighted combination of the ratios.

In order to construct the metric: 1) The concept of reachability is explained and analysed, 2) *novel* algorithms to compute the output reachable set for lateral and, combination of lateral and longitudinal scenarios will be described, 3) *novel* metric used to compute the avoidance capability of the vehicle based on output reachable sets is defined.

3-1 Fundamental on Reachability Analysis

Reachable set:

Consider the dynamical system given by

$$\dot{x} = f(x(t), u(t)) \tag{3-1}$$

where, $x(t) \in \mathbb{R}^m$ is the state of the system at time, t, $u(t) \in U$ (equation 3-2), are the inputs to the system at time, t, and $f : \mathbb{R}^m \times \mathbb{R}^n \mapsto \mathbb{R}^m$ be the function that governs the system dynamics.

Formally, the admissible control inputs, U is given by

$$U \triangleq \{\vec{u} : \exists\, \vec{u} \in \mathbb{R}^n,\ \circ(g(\vec{u}))\} \tag{3-2}$$

where, $\vec{u} = [u_1, u_2, ..u_n]^T$, $g()$ is the constraint equation for the inputs, $\circ(g())$ is a function that returns a true value if the constraint equation is satisfied i.e. $\circ(g())$ is a rule (or predicate) and if for a certain $\vec{u} \in \mathbb{R}^n$, the predicate holds true, $\vec{u}$ belongs to the set being defined.

The reachable set $\mathbf{R}(X_0)$ contains all the possible states that can be reached, from a set of initial states, X_o for the dynamical system given in equation 3-1.

The reachable set can be computed, by forward propagating the system model with all the possible combinations of the control input[1], u and all initial states, X_o. Formally, this can be defined as:

$$\mathbf{R}(X_0) \triangleq \{x(\alpha, u, x_o)\,, \forall \alpha \in [t_o,\ \infty),\ \forall u \in U, \forall x_0 \in X_0\} \tag{3-3}$$

where, t_0 is the initial time, $x(t, u, x_o)$ is the end state of the trajectory traversed by dynamical system from an initial state, x_o on applying a control input, $u(t)$, for a time interval, t.

However, the definition of reachable set as presented in equation 3-3 does not hold true in the presence of obstacles, since the reachable set according to the definition also includes states that belong in the obstacle region (X_{obs}) or, those states that can only be reached, by visiting the states in the obstacle region. Therefore, a new reachable set needs to be defined to account for the obstacle region.

Obstacles can be dynamic. This implies that the obstacle region changes with time, t. At every time-step, the obstacle region can be predicted using motion models. Therefore, the obstacle region can be accounted for, by incorporating time t in the definition of the reachable set.

Definition:

$\mathbf{R}(t, X_0)$ is defined as the states that can be reached by the dynamical system in a time interval, t, from a set of initial states, X_o in the presence of obstacles. Formally, the reachable set can be defined as:

$$\mathbf{R}(t, X_0) \triangleq \{x(t, u, x_o) : \exists u \in U, \exists x_0 \in X_0,\ x(\tau, u, x_o) \notin X_{obs}\, \forall\, \tau \in [t_0, t]\} \tag{3-4}$$

where, X_{obs} is the set of states that belong in the obstacle region.

3-1-1 Computation of reachable sets

Computing the exact reachable set of a nonlinear system is an open question in the literature [14]. However, a way to compute the over-approximation of the reachable set has been developed in [15]. An over-approximation of the reachable set is larger in the sense that it contains the exact reachable set. Note that if the over-approximation of the reachable set does not contain a set of restricted states, then the exact reachable set does not either. The general steps in computation of over-approximation of a reachable set involve numerical integration

[1]for a non-autonomous system.

and computational geometry, that makes use of different geometrical representations such as polytopes [16], zonotopes [17], ellipsoids [18], etc, to represent the reachable sets. The choice of the set representations become paramount since it has a direct consequence on the scalability of the over-approximation algorithm. This means that the computational complexity of the algorithm increases with an increase in the number of states of the dynamical system.

Another way is through abstraction, i.e. to replace the non-linear dynamics with simple linear dynamics for which the exact reachable set can be computed. [19]. [20] gives a nice overview on the various reachability techniques for different types of system dynamics.

However, none of the reachability algorithms surveyed accounts for a forbidden (i.e., obstacle) region in the output space of the system model. In Section 3-2-3, an algorithm is proposed, that deals with a forbidden region for a static scenario that computes an approximation of the reachable set ($\mathbf{R^3}$ in Figure 3-2) for the given system model and road scenario.

3-2 Reachable vehicle states

The computation of reachable set, however, becomes difficult in the presence of obstacles. As the computation would require the representation of the obstacle-free region, which can turn out to be non-convex in some scenarios and also non-connected, thereby making it difficult to compute. Therefore, we simplify the vehicle dynamics to a double integrator model, with bounded input accelerations.

3-2-1 System model

The simplified vehicle model under consideration is given in equation 3-5.

$$\begin{pmatrix}\dot{x}_s\\ \ddot{x}_s\\ \dot{y}_s\\ \ddot{y}_s\end{pmatrix} = \begin{pmatrix}0&1&0&0\\0&0&0&0\\0&0&0&1\\0&0&0&0\end{pmatrix}\begin{pmatrix}x_s\\ \dot{x}_s\\ y_s\\ \dot{y}_s\end{pmatrix} + \begin{pmatrix}0&0\\1&0\\0&0\\0&1\end{pmatrix}\begin{pmatrix}u_{x_s}\\ u_{y_s}\end{pmatrix} \qquad (3\text{-}5)$$
$$|u_{x_s}| \leq a_{max,x}$$
$$|u_{y_s}| \leq a_{max,y}$$

where, $\hat{x} \triangleq (x_s, \dot{x}_s, y_s, \dot{y}_s)^T$ is the state vector, u_{x_s}, u_{y_s} are the longitudinal and lateral accelerations respectively, $a_{max,x}$, $a_{max,y}$ are the maximum longitudinal and lateral accelerations. Note that the allowable input space in this section results in a rectangle, whose limits (length, breadth) are given by $a_{max,x}$, $a_{max,y}$ unlike Chapter 2, where, the allowable input space was given by the Kamm's circle.

Positions, x_s, y_s are measured with respect to a random fixed inertial frame of reference. In Figure 3-1, the inertial frame of reference is situated at the same place as the vehicle's position at time, t_o.

Figure 3-2 gives an illustration of the different reachable sets that are considered and computed in this thesis.

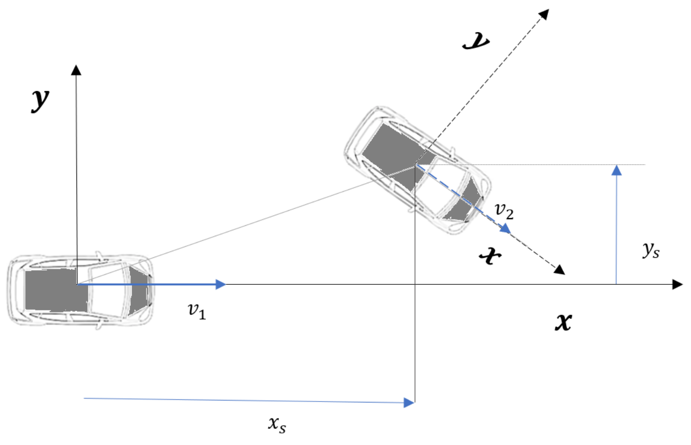


Figure 3-1: Illustration of the local frame of reference of the vehicle. The dashed axes are the frames of reference at a later time-step. x_s, y_s are the positions of the host vehicle at a later time-step, as measured by the host vehicle at the current time step (t_o).

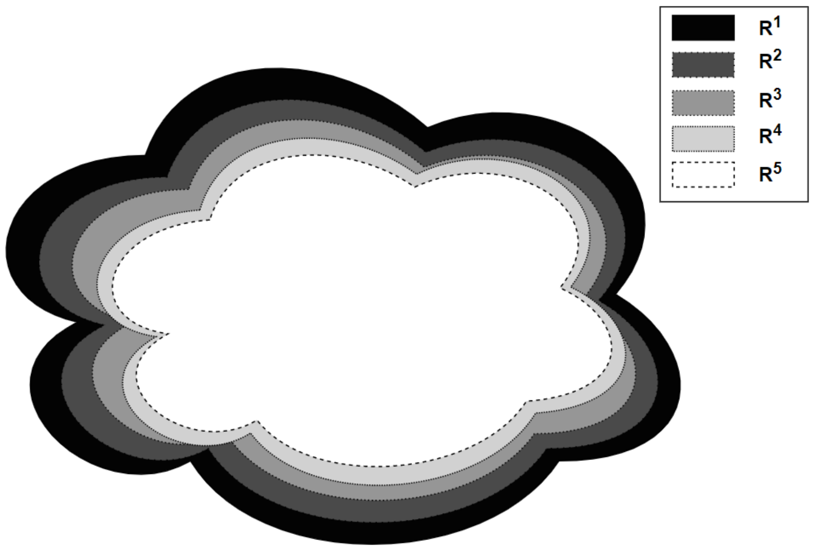


Figure 3-2: Illustration of Reachable sets. $\mathbf{R^1} \supseteq \mathbf{R^2} \supseteq \mathbf{R^3} \approx \mathbf{R^4} \supseteq \mathbf{R^5}$. $\mathbf{R^1}$ is the reachable set of the double integrator system model *without* considering obstacles. $\mathbf{R^2} = \mathbf{R^1} \setminus X_{obs} = \mathbf{R^1} \cap \overline{X}_{obs}$. $\mathbf{R^3}$ is the reachable set obtained from the proposed algorithms in Section 3-2-2 and Section 3-2-3. $\mathbf{R^4}$ is the reachable set of a double integrator system model in the presence of obstacles. $\mathbf{R^5}$ is the exact reachable set of a real-life vehicle. Note that unlike $\mathbf{R^4}$, $\mathbf{R^2}$ also contains states that can only be reached, by visiting the states in the obstacle region.

In Figure 3-2, $\mathbf{R^4}$ denotes the reachable state set of the double integrator vehicle model in equation 3-5. Note that this set contains many more states than those reachable by an actual real-life vehicle (denoted $\mathbf{R^5}$). In this sense, $\mathbf{R^3}$ is an over approximation. Note that unlike $\mathbf{R^4}$, $\mathbf{R^2}$ also contains states that can only be reached, by visiting the states in the obstacle region.

A closed-form solution to compute the reachable set, $\mathbf{R^1}$ for the model given in equation 3-5 exists in the absence of any obstacle [21]. However, in the presence of obstacles, numerical methods need to be employed to compute the reachable set, since the obstacle and reachable regions overlap.

As it can be seen from equation 3-5, the dynamics in longitudinal and lateral directions are decoupled. Therefore, in the absence of obstacles, the closed form solution can be computed independently. However, to check if a state is in the obstacle region, the position from both the dimensions, (x_s, y_s) must be considered.

In the subsequent subsections, reachability is discussed, first in the 1-dimensional sense for a pure longitudinal motion and then, an algorithm is proposed to compute this reachable set. Then, this algorithm is extended to the 2-dimensional sense for the combination of lateral and longitudinal motion.

3-2-2 Reachability in one dimension (pure longitudinal)

Considering only the longitudinal motion of the double integrator model. The longitudinal dynamics are given by equation 3-6.

$$\begin{aligned} \begin{pmatrix} \dot{x}_s \\ \ddot{x}_s \end{pmatrix} &= \underbrace{\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}}_{A} \begin{pmatrix} x_s \\ \dot{x}_s \end{pmatrix} \quad + \quad \underbrace{\begin{pmatrix} 0 \\ 1 \end{pmatrix}}_{B} \begin{pmatrix} u_{x_s} \end{pmatrix} \\ |u_{x_s}| &\leq a_{max,x} \end{aligned} \tag{3-6}$$

where, $x' \triangleq (x_s, \dot{x}_s)^T$ is the state vector and input, u_{x_s} is the longitudinal acceleration.

In case of pure longitudinal dynamics scenario, the output space of the system model is the same as the state space of the system model as given in equation 3-7.

$$\begin{pmatrix} x_s \\ \dot{x}_s \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{C} \begin{pmatrix} x_s \\ \dot{x}_s \end{pmatrix} + \underbrace{\begin{pmatrix} 0 \\ 0 \end{pmatrix}}_{D} \begin{pmatrix} u_{x_s} \end{pmatrix} \tag{3-7}$$

where, $y' \triangleq (x_s, \dot{x}_s)^T$ are the outputs of the system model.

In the absence of obstacles, i.e. $X_{obs} = \emptyset$, the reachable set for a given set of initial states, X_o and initial time, t_o can be obtained by [21]:

$$\begin{aligned}
\mathbf{R}(t, X_0) &= \{x \in \mathbb{R}^2 \,|\, x = e^{At}x_o + \int_0^t e^{A(t-\tau)Bu_{x_s}(\tau)d\tau}, \forall\,(x_0, u_{x_s}) \in X_o \times U\} \\
&= e^{At}X_o \bigoplus \underbrace{\{x \,|\, x = \int_0^t e^{A(t-\tau)Bu_{x_s}(\tau)d\tau},\ \forall u_{x_s} \in U\}}_{=:P_{u_{x,s}}(t)} \\
&= e^{At}X_o \bigoplus P_{u_{x,s}(t)}
\end{aligned}$$

where, $\bigoplus$ denotes the Minkowski sum.

The first part of the reachable set ($e^{At}X_o$), can get computationally expensive in the case of high dimensional linear systems [22]. However, in this case, A is a 2 × 2 matrix and therefore, can be computed by the means of Taylor series or other methods as enumerated in [23].

$P_{u_{x,s}}(t)$ represents the set of states that can be reached in time, ‘t’, when the initial state is at the origin i.e., $(x_s, \dot{x}_s)^T = (0, 0)^T$. From now on, the initial set is considered to be at origin, thereby making the reachable set $\mathbf{R}(t, X_0) = P_{u_{x,s}}(t)$ [2]. Also, for the sake of brevity, ‘t’ will be dropped from the notation of the reachable set, $\mathbf{R}(t, X_0) = P_{u_{x,s}}$.

Now, the objective is to compute the reachable set, $P_{u_{x,s}}$. This can be done by computing the boundary of the reachable set since the states that belong either inside or on the boundary of the reachable set, comprises the reachable set, $P_{u_{x,s}}$. In order to obtain the boundary of these set of states ($P_{u_{x,s}}$), for a given time interval, the maximum and minimum velocities at every position that can be reached in time, t is required. This can be obtained for a double integrator model by understanding the optimal planning (minimum time) to navigate from an initial ($x'_{in} = (x_s^{in}, \dot{x}_s^{in})^T$) and final state ($x'_f = (x_s^f, \dot{x}_s^f)^T$). This is done in [2, Example 15.4], by utilising Pontryagin principle.

Without going too much in detail, the procedure entails the minimisation of the Hamiltonian function to obtain the optimal control action from the evolution of adjoint variables. The Hamiltonian function is given in equation 3-8.

$$H(x', u_{x_s}, \lambda) = j(x', u_{x_s}) + \sum_{i=1}^{n} \lambda_i f_i(x', u_{x_s}) \tag{3-8}$$

where, $j(x', u_{x_s})$ is the cost function, λ_i are the adjoint variables, f_i is the system model (double integrator in this case).

In [2], the cost function, $j(x', u_{x_s}) = 1$ is defined for all $x' \in \mathbb{R}^2$ and $u_{x_s} \in U$. Then, the Hamiltonian function is given in equation 3-9.

$$H(x_s, u_{x_s}, \lambda) = 1 + \lambda_1(\dot{x}_s) + \lambda_2(u_{x_s}) \tag{3-9}$$

The optimal trajectory is then obtained by minimising the Hamiltonian as given in equation 3-10.

$$u_{x_s}(t)^* = \text{argmin}\{1 + \lambda_1(t)(\dot{x}_s(t)) + \lambda_2(t)(u_{x_s}(t))\} \tag{3-10}$$

[2]If the initial set is not at the origin, the reachable set would simply be the Minkowski sum of $P_{u_{x,s}}(t)$ (needs to be developed) and $e^{At}X_o$ (can be computed fairly easily).

As, it can be noted from equation 3-10, the minimum depends on the sign of $\lambda_2(t)$. If $\lambda_2(t) < 0$, then the minimum can be obtained by choosing $u_{x_s}(t) = -a_{max,x}$, and if $\lambda_2(t) > 0$ then $u_{x_s}(t) = a_{max,x}$. Hence, based on the sign of $\lambda_2(t)$, the input $u_{x_s}(t) = -sign(\lambda_2(t))$. However, when $\lambda_2(t) = 0$, the input can be chosen to be anything in it's domain, since it does not contribute in the equation 3-10.

In order to choose the the values of the adjoint variables $(\lambda_1(t), \lambda_2(t))$, the adjoint transition equation as given in equation 3-11 must be solved [2].

$$\dot{\lambda}_i = -\frac{\partial H}{\partial x'_i} \tag{3-11}$$

On solving 3-11, we obtain $\dot{\lambda}_1 = 0$ and $\dot{\lambda}_2 = -\lambda_1$. The solution are then of the form, $\lambda_1(t) = a$ and $\lambda_2(t) = b - at$, where a and b are constants. Since the solution to $\lambda_2(t)$ is linear, it can be observed that, based on the values of a, b, the sign of $\lambda_2(t)$ can change at most once.

Therefore, as remarked in [2], there are four possible actions depending on the initial and final states (x'_{in} and x'_f).

1. Absolute acceleration, $u_{x_s} = a_{max,x}$ is applied for time 't'.
2. Absolute deceleration $u_{x_s} = -a_{max,x}$ is applied for time 't'.
3. Absolute acceleration upto a certain time (γt), followed by absolute deceleration for the rest of the time $(t - \gamma t)$.
4. Absolute deceleration upto a certain time (γt), followed by absolute acceleration for the rest of the time $(t - \gamma t)$.

where, $\gamma \in [0, 1]$ and t is the time interval.

The first two actions can be obtained by either applying $\gamma = 0$ or 1, in both, the third and fourth actions. Note that if the time interval, t is fixed, the third and the fourth actions can be used to obtain the maximum and minimum velocities for a given a position, i.e. the actions can be used to obtain the lower and upper bounds of the reachable set respectively [21] as given in equations 3-12, 3-13 and 3-14, 3-15 respectively.

$$x_s^{(l)} = x_{s,0} + \dot{x}_{s,0}t - a_{max,x}\, t^2\,(\gamma^2 - 2\gamma + 0.5) \tag{3-12}$$
$$\dot{x}_s^{(l)} = \dot{x}_{s,0} - a_{max,x}\, t\,(1 - 2\gamma) \tag{3-13}$$

$$x_s^{(u)} = x_{s,0} + \dot{x}_{s,0}t + a_{max,x}\, t^2\,(\gamma^2 - 2\gamma + 0.5) \tag{3-14}$$
$$\dot{x}_s^{(u)} = \dot{x}_{s,0} + a_{max,x}\, t\,(1 - 2\gamma) \tag{3-15}$$

The bounds at a given time t are obtained by varying γ in the interval $[0, 1]$ and substituting them in the equations 3-12, 3-13 and 3-14, 3-15.

Figure 3-3 shows the trajectories obtained by applying the third and fourth actions when the switching interval (γ) is varied in the steps of 0.001. It can be observed that the end states of these trajectories form the boundary of the reachable set $(P_{u_{x,s}})$.

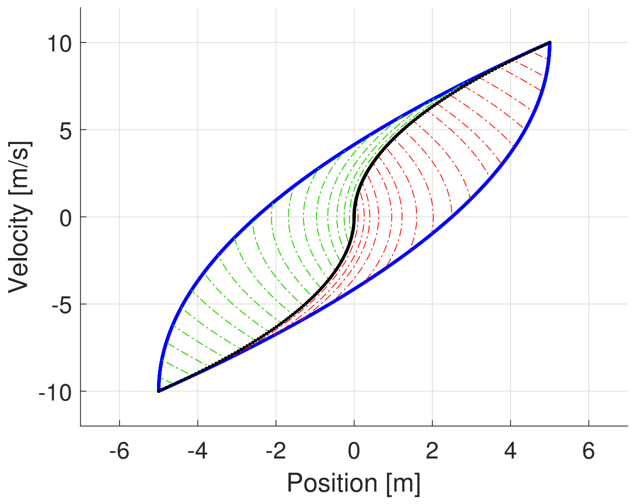


Figure 3-3: Blue: Boundary of the reachable set ($P_{u_{x,s}}$) for a double integrator model for $t = 1s$ and $a_{max,x} = 10m/s^2$. Red (dashed line): Trajectories obtained by applying absolute acceleration followed by absolute deceleration, given the initial state $(0, 0)$. Green (dashed line): Trajectories obtained by applying absolute deceleration followed by absolute acceleration. Black: Absolute acceleration (in the 1st quadrant) and absolute deceleration (in the 3rd quadrant).

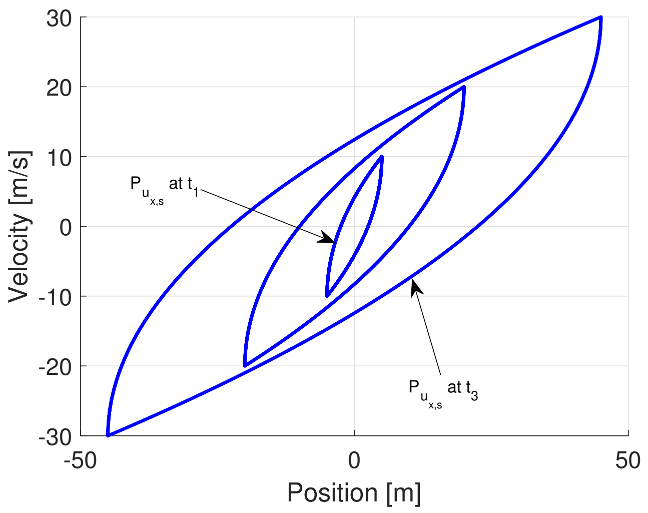


Figure 3-4: Boundary of the reachable sets ($P_{u_{x,s}}(t)$) for a double integrator model at $t = 1, 2, 3s$ and $a_{max,x} = 10m/s^2$.

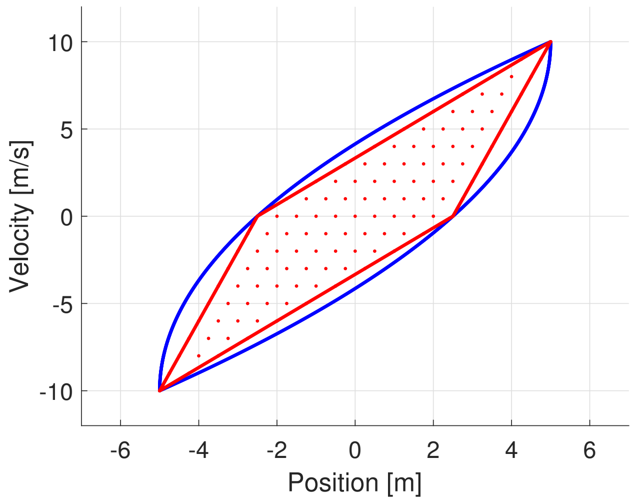


Figure 3-5: Blue: Boundary of the reachable set ($P_{u_{x,s}}(t)$) for a double integrator model for $t = 1s$ and $a_{max,x} = 10m/s^2$. Red dots: Some of the reachable states for a double integrator model for $t = 1s$ and $a_{max,x} = 10m/s^2$. Red: Approximation of the reachable set by calculating the convex hull after applying discretised control inputs to the system.

Figure 3-4 shows the boundary of the reachable set at different time steps. Figure 3-5 shows the boundary of the reachable set obtained by substituting γ from [0,1] in the steps of 0.001.

Obstacles: Until now, the process to calculate boundary (as shown in Figure 3-4) did not include any obstacles. In the presence of a static obstacle, let's say a lane boundary that must not be exited, the reachable set would shrink. The reachable set can now be computed, by dismissing the states that end up in the obstacle region (exit the lane boundaries).

The reachable set then becomes:

$$P^{'}_{u_{x,s}} = P_{u_{x,s}} \setminus P_{u_{x,col}} \tag{3-16}$$

where, $P_{u_{x,col}}$ are the collision states.

In addition to the collision states in $P_{u_{x,col}}$, there might exist states outside that set that are such that were the vehicle to reach them at time, t, due to its speed and position the vehicle would eventually reach X_{obs} at a later time regardless of what input action u were applied. These states are termed as inevitable collision states (ICS)[24].

Definition:
A state, x_o is an inevitable collision state, if there exists a time, τ, for all inputs, u, $x(\tau, u, x_o)$ belongs in the obstacle region X_{obs}. Formally, ICS can be defined as follows:

$$\textbf{ICS} = \{x_o : \exists \tau \in [t_0, \infty),\, x(\tau, u, x_o) \in X_{obs}\, \forall u \in U\} \tag{3-17}$$

In case of a static obstacle, ICS can be computed by backward propagating the obstacle region as will be shown subsequently. This is also illustrated in Chapter 4 (Figure 4-2b).

The reachable set in the presence of static obstacle is given in equation 3-18.

$$P^{''}_{u_{x,s}} = P'_{u_{x,s}} \setminus ICS = P'_{u_{x,s}} \cap \overline{ICS} \tag{3-18}$$

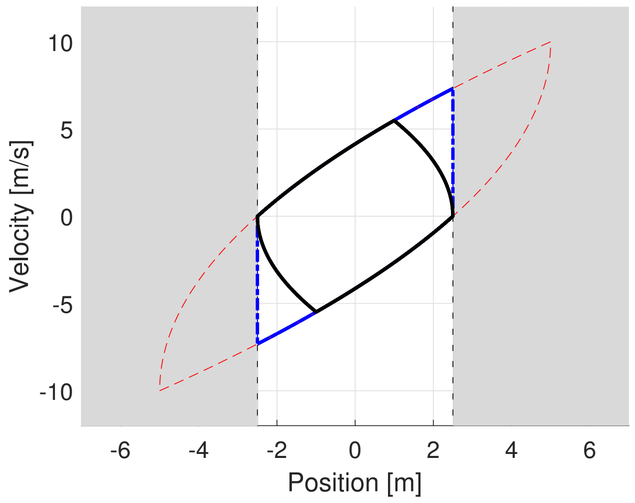


Figure 3-6: Red: Boundary of the reachable set ($P_{u_{x,s}}$) for a double integrator model for $t = 1s$ and $a_{max,x} = 10m/s^2$ without considering the obstacle. Blue: Boundary of the reachable set ($P^{'}_{u_{x,s}}$) by dismissing the states in the obstacle region. Black (thick): Boundary of the reachable set ($P^{''}_{u_{x,s}}$) by disregarding the states (inevitable collision states), that eventually end up in the obstacle region. Grey: Obstacle region.

where, ICS are states, that regardless of the input action applied, end up in the obstacle region.

Given a state, based on the velocity, it can be found out if it can avoid ending up in the obstacle region. For example, let's consider an obstacle at -2.5 & $2.5m$ from the host vehicle which is at the origin $(0, 0)$. When the obstacle is at $2.5m$, the obstacle region consists of all states beyond $2.5m$ with any velocity, which translates to vertical lines as shown in Figure 3-6.

If the initial state of the vehicle has a velocity (v), the distance travelled (s_{tr}) before it comes to a stop is given by equation 3-19.

$$s_{tr} = \frac{v^2}{2a_{max,x}} \tag{3-19}$$

Since the distance between the host vehicle and the obstacle is known, the maximum velocities for the vehicle, given the vehicle position can be obtained by making v the subject of the equation 3-19. This results in the boundary of ICS states.

Figure 3-6 shows the reachable set at three stages. The first stage is by not considering the presence of obstacle. The second stage is by disregarding the states that belong in the obstacle region and third stage is by removing the states (ICS) that eventually end up in the obstacle region. The boundary of the ICS is computed by calculating the minimum of the result from the second stage and the maximum velocities obtained for (collision-free) positions ($[-2.5, 2.5]$ in steps of $0.01m$) from equation 3-19.

Approximation: In the presence of obstacles, there does not exist any analytic way to compute the reachable set at every time interval. Therefore, a way has to be devised to approximate this reachable set ($\mathbf{R^3} \approx \mathbf{R^4}$).

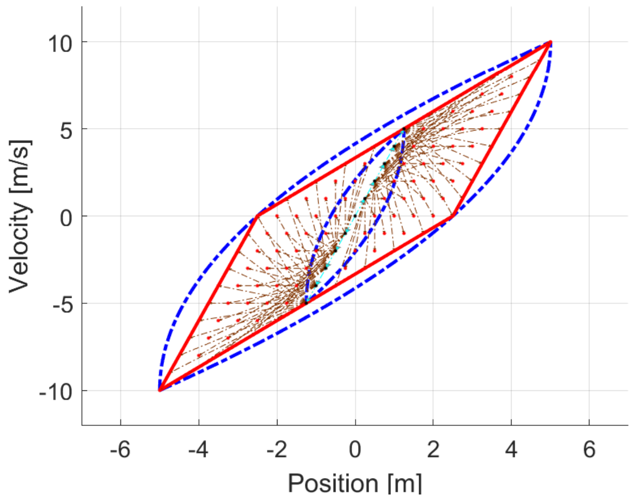


Figure 3-7: For the magnified plot, refer to Figure 3-8. Blue: Boundary of the reachable set ($P_{u_{x,s}}$) for a double integrator model at $t = 0.5, 1s$ and $a_{max,x} = 10m/s^2$. Cyan (Arrows): States obtained after forward propagating the model by applying discretised inputs when the initial state is at $(0, 0)$ Red: Approximation of the reachable set by calculating the convex hull of the set (end of the Firebrick coloured arrows) obtained by applying discretised control inputs to the system for the initial states (obtained from previous time step [cyan]).

One of the ways would be to approximate the reachable set at a given time interval by forward propagating the system model, by applying discretised control actions for a given initial state(s). Then, the sets obtained from forward propagation, are checked for collision with the obstacle (presence in the inevitable collision region) and the states that are present in the inevitable collision region are removed. Afterwards, a convex hull of the collision-free states is computed. This convex hull becomes the set of initial states for the next time interval.

Figure 3-8 shows approximations of the reachable set at two time steps $(k-1, k)$. The states represented by the black dots (end of the cyan coloured arrows) are obtained by applying discretised control inputs ($u_{x_s} = [-10, 10]$ in the steps of $2\,[m/s^2]$) to the system model (equation 3-6), given that the initial state is at the origin $(0, 0)$.

A convex hull is computed for these states (a straight line at the first time step, since the states lie on a line). The convex hull is represented by a convex polygon[3] (sorted set of states) which is an intersection of half-planes that become the set of initial states for the next time step, k which are forward propagated with the discretised control inputs.

This results in states represented by red dots, at the end of the firebrick arrows. A convex hull is computed for these states represented by red lines in Figure 3-8 which become the set of initial states for the next time step $(k+1)$. This process describes the forward propagation procedure, which is repeated for the total time in consideration.

Note that in the case of dynamic obstacles, only collision states are removed and not ICS. This is done since the computation of the ICS region in the presence of dynamic obstacles is a challenging problem. Although algorithms do exist for the case of dynamic obstacles with

[3]Refer Appendix A-1-1 for representation of convex polygon.

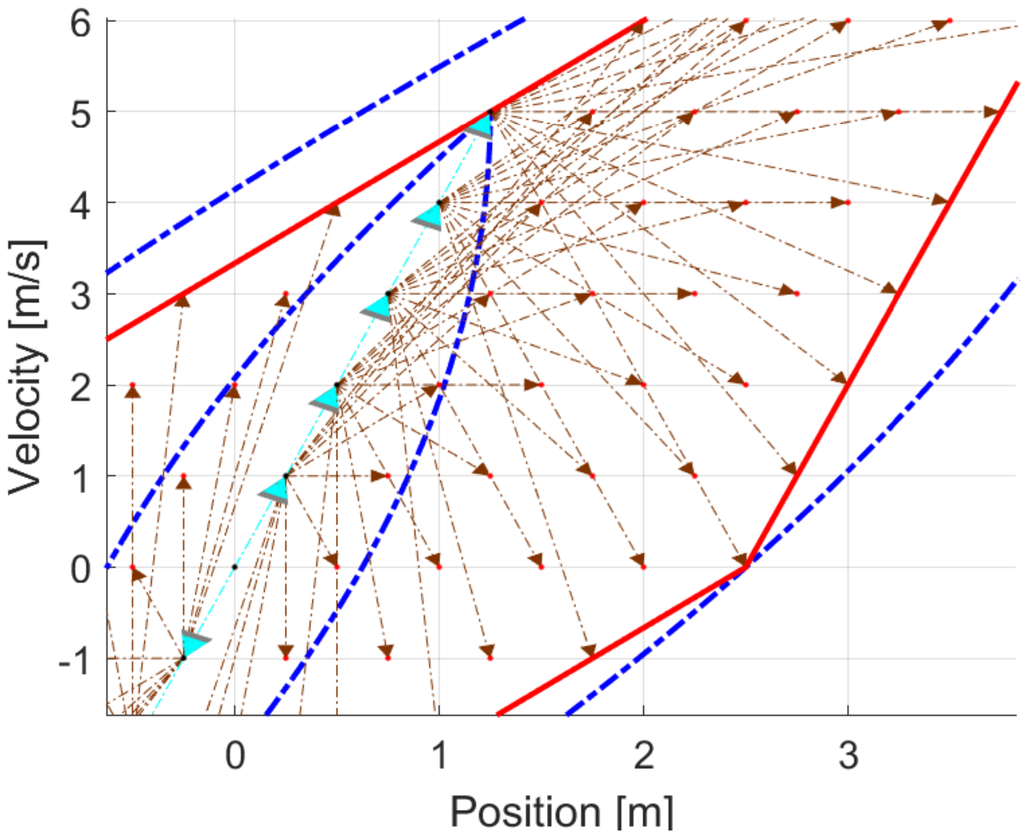


Figure 3-8: Zoomed in plot of Figure 3-7. Blue: Boundary of the reachable set ($P_{u_{x,s}}$) for a double integrator model at $t = 0.5, 1s$ and $a_{max,x} = 10m/s^2$. Cyan (Arrows): States obtained after forward propagating the model by applying discretised inputs when the initial state is at $(0, 0)$ Red: Approximation of the reachable set by calculating the convex hull of the set (end of the Firebrick coloured arrows) obtained by applying discretised control inputs to the system for the initial states (obtained from previous time step [cyan]).

perfectly-known behaviour ([24], [25]), they are not trivial. Hence, the computation of the ICS region in the presence of dynamic obstacles is outside the scope of this work.

Algorithm 1 and 2 present the procedure to compute the approximation of the reachable set for a double integrator model, for a pure longitudinal motion *without* and *with* the consideration of obstacles.

From here on, P_k^x is used to represent the convex hull of the approximation of reachable set ($P_{u_{x,s}}$) in longitudinal direction at time step, k.

Algorithm 1 Algorithm for the reachable set, R^3 in 1-dimension (Pure longitudinal motion) *without* considering obstacles.

```
1:  $k = 1$ ; $G.v(k) \leftarrow z_{init}$ ;
2:  while k < N do
3:      $P_k^x \leftarrow$ forwardpropagate $(G, \tilde{P}_{k-1}^x, param)$ ;
4:      if !(linecheck $(P_k^x)$) then
5:          $\hat{P}_k^x \leftarrow$ convhull $(P_k^x)$ ;
6:          $\tilde{P}_k^x \leftarrow$ interpolation $(\hat{P}_k^x, param)$ ;
7:          else
8:          $\tilde{P}_k^x \leftarrow P_k^x$ ;
9:      end if
10:
11:     $Area_k \leftarrow$ polyarea $(\tilde{P}_k^x)$ ;
12:     $G \leftarrow$ make-graph $(G, \tilde{P}_k^x)$ ;
13:     $k \leftarrow k + 1$
14: end while
```

Algorithm 2 Algorithm for reachable set, R^3 in 1-dimension (Pure longitudinal motion) considering obstacles.

```
1:  $k = 1$ ; $G.v(k) \leftarrow z_{init}$ ;
2:  while k < N do
3:      $P_k^x \leftarrow$ forwardpropagate $(G, \tilde{P}_{k-1}^x, param)$ ;
4:      if !(linecheck $(P_k^x)$) then
5:          $\hat{P}_k^x \leftarrow$ convhull $(P_k^x)$ ;
6:          $\tilde{P}_k^x \leftarrow$ interpolation $(\hat{P}_k^x, param)$ ;
7:          else
8:          $\tilde{P}_k^x \leftarrow P_k^x$ ;
9:      end if
10:     $\tilde{P}_k^x \leftarrow$ collisionfree-ics$(P_{ICS}^x, \tilde{P}_k^x)$ ;
11:     $Area_k \leftarrow$ polyarea $(\tilde{P}_k^x)$ ;
12:     $G \leftarrow$ make-graph $(G, \tilde{P}_k^x)$ ;
13:     $k \leftarrow k + 1$
14: end while
```

A short description on the functions used by Algorithm 1 are enumerated below:

1. *Forward propagation (in $x - v_x$ plane):*
   The convex polygon enclosing the reachable set at the previous time step, $k-1$ is prop-

agated according to the approximation method described in Section 3-2-2 (by forward propagating a select set of states ($\tilde{P}^x_{k-1}$) with a system model given by equation 3-6), also illustrated in Figure 3-8 resulting in P^x_k. Note that at the first time-step, $\tilde{P}^x_0$, is the initial state of the vehicle.

2. *Line check:*
   All the states i.e. $(x\,, v_x)$[4] pairs in the set, P^x_k are checked if they are collinear i.e. if they lie on a line. Note that the output of *linecheck* function is a boolean value. If the points are collinear, the output is *True* and *False* if otherwise.

3. *Collision-free-ics:*
   The forward propagated set ($\tilde{P}^x_k$), are checked if they lie outside the ICS region. This region is obtained from equation 3-19, which gives the maximum velocity for a given distance between the obstacle and the host. The states that have higher velocities than the maximum (absolute value), fail this check and are subsequently removed, resulting in:
   $$\tilde{P}^x_k = \tilde{P}^x_k \setminus P^x_{k,ICS} \tag{3-20}$$
   where, $P^x_{k,ICS}$ are the inevitable collision states at time interval, k.

   Note that the discussed function is only valid for static obstacles. In case of dynamic obstacles, $P^x_{k,ICS}$ is replaced with $P^x_{k,col}$ in the function, where, $P^x_{k,col}$ are the collision states at time interval, k.

4. *Convex hull computation:*
   Given a set, P^x_k , a convex hull (represented by a convex polygon) is constructed in $x - v_x$ plane to obtain the boundary of the reachable set at every time step, k:
   $$\hat{P}^x_k = \text{convhull}\Big(P^x_k\Big) \tag{3-21}$$
   where, $\hat{P}^x_k$ is an *ordered* set that represents a convex polygon which encloses an approximation of the reachable set P^x_k.

5. *Interpolation function:*
   Based on the length of the boundary of the reachable set ($\tilde{P}^x_k$), a fixed number of states are interpolated to represent the boundary of the reachable set. This helps to keep the total number of states (x, v_x) bounded, for every road scenario.

6. *Polyarea:*
   This function computes the area enclosed by the set $\tilde{P}^x_k$.

7. *Make-graph:*
   This function stores $\tilde{P}^x_k$ in a graph, G. The graph, G comprises of vertices, where each vertex contains (x, v_x, k).

Initialised with z_{init} $(= (\tilde{P}^x_0, 0))$ as the only vertex, the algorithm builds a tree of samples, whose boundary gives an approximation of the reachable set at a given time step, k. This is

[4]$(x\,, v_x)$=$(x_s, \dot{x}_s)$

done by first forward propagating in the (x, v_x) plane for given initial states, $\tilde{P}^x_{k-1}$ (Line 3) as shown in Figure 3-8. Note that at the first time-step, $\tilde{P}^x_0$, is the initial state of the vehicle.

At the first time step, on propagating with a certain set of discrete control inputs, the states , $\tilde{P}^x_0$ lie on a straight line. However, when it is not the first time step, a convex hull is computed for the set of states in P^x_k (no longer lies on a line) which is later interpolated based on the length of the convex hull, $\hat{P}^x_k$ (Lines 6).

Subsequently, based on the type of obstacle i.e., either static or dynamic, the convex hull is checked if it lies in ICS region or collision region respectively. The states on the convex hull that lie in this region are removed, thereby resulting in a collision-free convex hull, $\tilde{P}^x_k$.

The area of the convex hull is computed and the convex hull is then added into the graph as described by $make - graph$ function above (Line 11-12).

3-2-3 Reachability in two dimensions (lateral and longitudinal)

This section proposes an algorithm that computes the reachable set in the presence of obstacles, by computing an estimate of the approximation of the reachable set at discrete time intervals.

In case of combination of both lateral and longitudinal dynamics, the output space of the system model is given in equation 3-22.

$$\begin{pmatrix} x_s \\ y_s \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_s \\ \dot{x}_s \\ y_s \\ \dot{y}_s \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} u_{x_s} \\ u_{y_s} \end{pmatrix} \tag{3-22}$$

where, $y'' \triangleq (x_s, y_s)^T$ are the outputs of the system model.

Up to now, the reachability analysis was applied to x-direction (pure longitudinal motion).

A reasonable extension to compute the reachable set in two dimensions would be to independently propagate the reachable set in both the dimensions ($x - v_x$ and $y - v_y$ planes) and then, take a Cartesian product between them to obtain the reachable set. However, this extension would only be valid either in the absence of obstacles or in the presence of uniform obstacles[5].

Alternatively, one could parameterise the growth of the reachable set (or forward propagation of reachable set with time) as a function of 'x'. This means the reachable set in x-direction, $\tilde{P}^x_k$ is grown first and then for *each* of the states that constitute the boundary of reachable set in x-direction, a corresponding reachable set in y-direction, $\tilde{P}^y_{k,j}$ is grown.

As shown in the sequel, this approach was developed into an algorithm that accounts for a static obstacles and generate the reachable set of the vehicle.

Note that in in the combination of lateral and longitudinal motion, the extension would only eliminate the collision states and not the ICS unlike the proposed procedure for pure longitudinal motion. This is done since the computation of ICS in the presence of static

[5]Obstacles for which the y-limits are constant for any value of x (ex: lane boundaries (given by red dots) as shown in Figure 3-9).

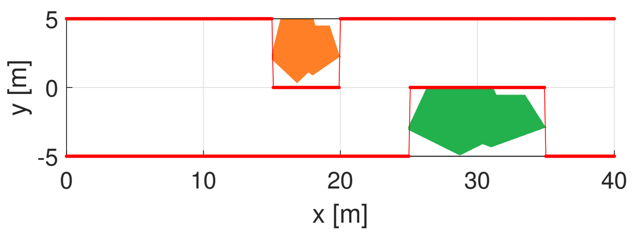


Figure 3-9: Black: Lane boundaries without considering obstacles. Red: New lane boundaries (maximum and minimum lane limits) in presence of obstacles (orange and green).

obstacles for the combination of lateral and longitudinal motion requires reachability analysis in itself which is not trivial [26]. Hence, the computation of the ICS region in the case of static obstacles for the combination of lateral and longitudinal motion is outside the scope of this work.

Before the algorithm is introduced, representation of the obstacle representation needs to be defined in order to perform a collision-check.

Obstacle representation: With the help of sensors, it is assumed that all the obstacles with a certain maximum distance in 'x' and 'y' directions from the host vehicle can be identified i.e. all the obstacle vehicles present in a window[6] are known. It is also assumed a convex polygon representation of obstacle is known (by enclosing the obstacle (with a safety margin) with a polygon [rectangle]).

The length of the window gives us the limits in x-direction. For example in Figure 3-9, the limits in 'x' are $0, 40\,[m]$. The limits in y-direction, however, are parameterised as a function of 'x' as illustrated in Figure 3-9.

The approximation is obtained by carefully selecting a set of samples from both x (longitudinal) and y (lateral) directions. These samples are obtained by computing the Cartesian product between the convex polygon enclosing the reachable set in x direction, $\tilde{P}^x_{k,j}$, and the union of the reachable set in y direction, $\tilde{P}^y_{k,j,i}$, as given in equation 3-23.

$$\mathbf{R}(t,(X_0,Y_0)) \quad \approx \quad \mathbf{R}_k = \bigcup_j \left(\tilde{P}^x_{k,j} \times \left(\bigcup_i \tilde{P}^y_{k,j,i} \right) \right) \tag{3-23}$$

where, $\mathbf{R}_k$ $(= \mathbf{R^3}$ in Figure 3-2) is the reachable set at time step, k and j, i are the indices of every (x, v_x) and (y, v_y)[7] pair respectively, $\tilde{P}^{x/y}_k$ is the reachable set at time-step, k, that is represented by an intersection of half-planes (similar to one dimensional case discussed in the previous section, also seen in Figure 3-5) resulting in a convex polygon, X_o, Y_o are the set of initial sets in x and y direction respectively. The axes of convex polygon for every direction (x, y), correspond to velocity $(v_{x/y})$ and position (x/y), just like in the one dimensional case.

[6]length and width are given by a maximum distance in 'x' and 'y' directions that the sensor can detect obstacles from the host vehicle.

[7]$(y\,, v_y)$=$(y_s, \dot{y}_s)$

Reachability algorithm for two dimensions (lateral and longitudinal)

A short description on the functions used by the Algorithm 2 are enumerated below:

1. *Forward propagation (both $x - v_x$ and $y - v_y$ planes):*
   The convex polygon enclosing the reachable set at the previous time step, $k-1$ ($\mathbf{R}_{k-1}$) is propagated according to the approximation method described in Section 3-2-2 (by forward propagating a select set of states ($\tilde{P}_{k-1}^{x/y}$) with a system model given by equation 3-6), also illustrated in Figure 3-8 resulting in $P_k^{x/y}$. Note that at the first time-step, $\tilde{P}_0^{x/y}$, is the initial state of the vehicle.

2. *Line check:*
   All the points $(x\,, v_x)$ or $(y\,, v_y)$ in the set, $P_k'^{x/y}$ are checked if they are collinear. Note that the output of *linecheck* function is a boolean value. If the points are collinear, the output is *True* and *False* if otherwise.

3. *Collision check:*
   The forward propagated sets ($P_k^{x/y}$), in both the dimensions, are checked if they are within the maximum and minimum lane limits. The states that fail this check are removed, resulting in:
   $$P_k'^{x/y} = P_k^{x/y} \setminus P_{k,col}^{x/y} \tag{3-24}$$
   where, $P_{k,col}^{x/y}$ are the collision states at time interval, k.

   Note that this function eliminates the collision states and not the ICS unlike the proposed procedure for pure longitudinal motion in the case of static obstacles.

4. *Convex hull computation:*
   Given a set, $P_k'^{x/y}$, a convex hull (represented by a convex polygon) is constructed in both x and y directions to obtain the boundary of the reachable set at every time step, k:
   $$\hat{P}_k^{x/y} = \text{convhull}\left(P_k'^{x/y}\right) \tag{3-25}$$
   where, $\hat{P}_k^{x/y}$ is an *ordered* set that represents a convex polygon which encloses an approximation of the reachable set $P_k'^{x,y}$.

5. *Interpolation function:*
   This function is used for both x and y dimensions. Based on the length of the boundary of the reachable set, ($\hat{P}_k^{x/y}$), a fixed number of states are interpolated to represent the boundary of the reachable set. This helps to keep the total number of states (x, v_x, y, v_y) bounded, for every road scenario.

6. *Selecting initial states ($y - v_y$ dimension):*
   At time step, k, the initial set of states ($\tilde{P}_{k-1}^{y}$) to forward propagate, are chosen based on the location of ($\tilde{P}_{k,j}^{x}$) i.e. $(x, v_x)_{k,j}$ pair amongst the ($\tilde{P}_{k-1}^{x}$) pairs obtained at time step $k-1$.

If the $(x, v_x)_{k,j}$ value at time step, k is lower than the minimum value of 'x' in $(\tilde{P}^x_{k-1})$ pairs at time step $k-1$, then the $\tilde{P}^y_{k-1}$ set that corresponds to the minimum of 'x' in $(\tilde{P}^x_{k-1})$ set is chosen as the initial set.

Similarly, if the $(x, v_x)_{k,j}$ value at time step, k is higher than the maximum value of 'x' in $(\tilde{P}^x_{k-1})$ pairs at time step $k-1$, then the $\tilde{P}^y_{k-1}$ set that corresponds to the maximum of 'x' in $(\tilde{P}^x_{k-1})$ set is chosen as the initial set.

However, there could exist a possibility that $(x, v_x)_{k,j}$ value at time step k lies in between the maximum and minimum value of 'x' in $(\tilde{P}^x_{k-1})$ set, then the $(\tilde{P}^y_{k-1})$ set that corresponds to the $(x, v_{x,})_{k-1}$ pair with the closest distance to $(x, v_x)_{k,j}$ in 'x' dimension is chosen as the initial set of states.

Note that in the absence of obstacles, at time-step, k , the set $\tilde{P}^y_k$ assumes the same set of states for all the the pairs in the set $\tilde{P}^x_k$.

7. *Shrink states ($x - v_x$ dimension):*
For a given $(\tilde{P}^x_{k-1,j})$ pair in $\tilde{P}^x_{k-1}$, there exists a corresponding set $\tilde{P}^y_{k-1}$ set. It does happen that sometimes due to the number/shape of obstacles, for certain pairs in $\tilde{P}^x_{k-1}$, there exist no $\tilde{P}^y_{k-1}$ set. In such a case, the set should not include those pairs in $\tilde{P}^x_{k-1}$, which has an *empty* $\tilde{P}^y_{k-1}$ set.

8. *Make-graph:*
For every j, this function computes the cartesian product between the set $\tilde{P}^y_{k,j}$ and $(\tilde{P}^x_{k,j})$ pair and stores them in a graph, G. The graph, G comprises of vertices, where each vertex contains $(x, v_x, y, v_y, k, j, i)$. The graph, G can be used to obtain the corresponding $\tilde{P}^y_{k,j}$ set for a given $(\tilde{P}^x_{k,j})$ pair. This attribute is utilised in functions, *Shrink states ($x - v_x$ dimension)* and *Selecting initial states ($y - v_y$ dimension).*

The algorithm to calculate the approximation of the reachable set in two dimension for a system model given by equation 3-5, is given in Algorithm 3.

Initialised with z_{init} $(= (\tilde{P}^x_0, \tilde{P}^y_0, 0, 1, 1))$ as the only vertex, the algorithm builds a tree of samples, whose boundary gives an approximation of the plausible vehicle position at a given time step, k. This is done by first forward propagating the (x, v_x) plane for given initial states, $\tilde{P}^x_{k-1}$ and removing the states that collide, i.e. outside the window in 'x' direction (Lines 3-4). At the first time step, on propagating with a certain set of discrete control inputs, the collision-free states, (P'^x_0) lie on a straight line. However, when it is not the first time step, a convex hull is computed for the set of states in P'^x_k (no longer lies on a line) which is later interpolated based on the length of the convex hull, $\hat{P}^x_k$ (Lines 7-8).

Now, that the convex hull, $\tilde{P}^x_k$ is known, a set P^y_k needs to be obtained for every (x, v_x) pair present on the convex hull, $\hat{P}^x_k$ (Lines 13-22). This is done as the limits for y is known for every x (from the way the obstacle is represented). Therefore, it is imperative that for every (x, v_x) pair, a set of (y, v_y) pairs, i.e. $\tilde{P}^y_{k-1}$ needs to be selected at time step $k-1$ that will be propagated at time step k. The procedure to select the (y, v_y) pairs has already been discussed in the function, *Selecting initial states ($y - v_y$ dimension)* explained above. Then, just like in (x, v_x) plane, a convex hull is computed for all sets P'^y_k, corresponding to all pairs of (x, v_x) in P'^x_k and also interpolated based on the length of the convex hull. The interpolations in both the dimensions are done in order to keep the number of samples bounded for a given number of time steps.

Once the boundary in both the planes are obtained, a Cartesian product is found between $\tilde{P}_k^x$ and $\tilde{P}_k^y$. The result of this Cartesian product is then added into the graph as described by $make - graph$ function above (Line 27).

Due to the obstacles' position, it can happen that at certain time intervals and (x, v_x) pair, $\tilde{P}_k^y = \emptyset$, i.e., there exists no position in the y dimension where the vehicle can be. Hence, the corresponding (x, v_x) pairs are removed from $\tilde{P}_k^x$ and also interpolated again based on the length of the boundary of $\tilde{P}_k^x$ (Lines 31-33). This procedure is described in the function, *Shrink states ($x - v_x$ dimension)* discussed above. This step makes the system dynamics in x depend on y.

This algorithm makes use of certain carefully selected samples to obtain an approximation to the reachable set. Note that due to the interpolation function, the total number of samples remain bounded, which turns out to be a desirable attribute of the algorithm.

3-3 Avoidance Capability Metric

Now that an approximation of the reachable set is known for a given initial state of the vehicle, an avoidance capability metric needs to be derived. This is done by keeping the desirable properties of a metric in mind, that effectively quantifies the avoidance capability of the host vehicle in the presence of a hazard.

3-3-1 Avoidance capability for only longitudinal motion

The metric can simply be the average of the ratio of the area enclosed by the boundary of the approximation of the reachable set with and without obstacles as given in equation 3-26.

$$AM = \left(\frac{A_{t_1}^{obs}}{A_{t_1}^{w/o-obs}} + \frac{A_{t_2}^{obs}}{A_{t_2}^{w/o-obs}} + \ldots. + \frac{A_{t_n}^{obs}}{A_{t_n}^{w/o-obs}} \right) \frac{1}{n} \tag{3-26}$$

As it can be seen, the avoidance metric given in equation 3-26 weighs the avoidance capability at every time step (t_1, t_2, $t_3, \ldots., t_n$) uniformly. However, it is desirable to have the weighting parameters decrease in value with increase in the number of time steps, thereby making the avoidance capability metric depend heavily on the initial time steps.

Therefore, the avoidance capability metric can be defined as the ratio of the area enclosed by the boundary of the approximation of the reachable set with and without obstacles, *multiplied with the appropriate weighting parameters* as given in equation 3-27. The weighting parameters are based on the total time for which the reachable set is computed and also the given time step.

$$\begin{aligned} AM = \left(\frac{T - t_2}{T_a}\right) \frac{A_{t_2}^{obs}}{A_{t_2}^{w/o-obs}} + \left(\frac{T - t_3}{T_a}\right) \frac{A_{t_3}^{obs}}{A_{t_3}^{w/o-obs}} + \ldots. \\ \ldots + \left(\frac{T - t_n}{T_a}\right) \frac{A_{t_n}^{obs}}{A_{t_n}^{w/o-obs}} \end{aligned} \tag{3-27}$$

Algorithm 3 Algorithm for approximation of reachable set in 2-dimensions

```
1: $k = 1$ ; $G.v(k) \leftarrow z_{init}$ ;
2: while k < N do
3:     $P_k^x \leftarrow$ forwardpropagate $(G, \tilde{P}_{k-1}^x, param)$ ;
4:     ${P_k^{'}}^x \leftarrow$ collisioncheck $(P_k^x, param)$ ;
5:
6:
7:     if !(linecheck $({P_k^{'}}^x)$) then
8:         $\hat{P}_k^x \leftarrow$ convhull $({P_k^{'}}^x)$ ;
9:         $\tilde{P}_k^x \leftarrow$ interpolation $(\hat{P}_k^x, param)$ ;
10:        else
11:        $\tilde{P}_k^x \leftarrow {P_k^{'}}^x$ ;
12:    end if
13:
14:    for all $\tilde{P}_{k,j}^x$ in $\{\tilde{P}_k^x\}$ do
15:        if !(linecheck $({P_k^{'}}^x)$) then
16:            ${P'}_{k-1}^y \leftarrow$ select-y-vystates ( $\tilde{P}_{k,j}^x$, $\tilde{P}_{k-1,j}^x$, $G$)
17:            $\tilde{P}_{k-1}^y \leftarrow {P'}_{k-1}^y$
18:        end if
19:        $P_k^y \leftarrow$ forwardpropagate $(G, \tilde{P}_{k-1}^y, param)$ ;
20:        ${P_k^{'}}^y \leftarrow$ collisioncheck $(P_k^y, param)$ ;
21:        if !(linecheck $({P_k^{'}}^x)$) and !(linecheck $({P_k^{'}}^y)$) and ${P_k^{'}}^y \neq \emptyset$ then
22:            $\hat{P}_k^y \leftarrow$ convhull $({P_k^{'}}^y)$ ;
23:            $\tilde{P}_k^y \leftarrow$ interpolation $(\hat{P}_k^y, param)$ ;
24:            elseif (linecheck $({P_k^{'}}^x)$ or linecheck $({P_k^{'}}^y)$) and ${P_k^{'}}^y \neq \emptyset$ then
25:            $\tilde{P}_k^y \leftarrow {P_k^{'}}^y$ ;
26:        end if
27:        if ${P_k^{'}}^y \neq \emptyset$ then
28:            $G \leftarrow$ make-graph $(G$, $\tilde{P}_k^y$,$\tilde{P}_{k,j}^x)$ ;
29:        end if
30:    end for
31:
32:    for all $\{\tilde{P}_k^y\} = \emptyset$ do
33:        $\tilde{P}_k^x \leftarrow$ shrinkstates-x-vx$(\tilde{P}_{k,j}^x)$ ;
34:        $\tilde{P}_k^x \leftarrow$ interpolation $(\tilde{P}_k^x)$ ;
35:    end for
36:    $k \leftarrow k + 1$
37: end while
```

where, T is the time window for which the reachable set is computed, t_2, $t_3, \ldots., t_n$ are the respective time steps that satisfy $\Sigma_{j=2}^{n} t_j = T_a$, A^{obs} is the area enclosed by the boundary of the reachable set considering obstacles and $A^{w/o-obs}$ is the area enclosed by the boundary of the reachable set *without* considering obstacles. Note that the Avoidance metric begins at the second time-step, t_2 since at the first time step the area enclosed is zero for both A^{obs} and $A^{w/o-obs}$.

3-3-2 Avoidance capability for both lateral and longitudinal motion

The avoidance capability metric for the case of motion considering both lateral and longitudinal dynamics is equivalent to the avoidance capability metric obtained from the pure longitudinal case. However, in this case area (A^{obs}, $A^{w/o-obs}$) is in the output space only (x, y), i.e. the boundary that encloses all the possible *positions* where the vehicle can be at discrete time intervals.

Therefore,

$$\begin{aligned} AM = \left(\frac{T-t_1}{T_a}\right)\frac{A_{t_1}^{obs}}{A_{t_1}^{w/o-obs}} + \left(\frac{T-t_2}{T_a}\right)\frac{A_{t_2}^{obs}}{A_{t_2}^{w/o-obs}} + \ldots. \\ \ldots + \left(\frac{T-t_n}{T_a}\right)\frac{A_{t_n}^{obs}}{A_{t_n}^{w/o-obs}} \end{aligned} \tag{3-28}$$

where, T is the time window for which the reachable set is computed, t_1, t_2, $t_3, \ldots., t_n$ are the respective time steps that satisfy $\Sigma_{j=1}^{n} t_j = T_a$, A^{obs} is the area enclosed by the boundary of the all the plausible positions of the host vehicle considering obstacles and $A^{w/o-obs}$ is the area enclosed by the boundary of all the plausible positions of the host vehicle *without* considering obstacles.

Chapter 4

Simulations on Use Cases

This chapter will elaborate on the implementation of the algorithms described in Chapter 3 on certain use cases (scenarios).

Firstly, the algorithms (Algorithm 1 & 2) devised in Section 3-2-2 is implemented to compute the avoidance metric for 1-D scenario (pure longitudinal motion). Afterwards, this avoidance metric is compared with the Brake Threat Number (BTN) via simulation. The computation procedure of BTN is described in Section 2-2-1.

Then, the algorithm (Algorithm 3) devised in Section 3-2-3 is implemented to compute the avoidance metric for a 2-D scenario (lateral and longitudinal motion). This algorithm entails the computation of approximation of the reachable set in x and y dimensions. Finally, this algorithm is compared with Rapidly-exploring Random Trees (RRT), which gives an under-approximation of the exact reachable set, $\mathbf{R^5}$.

All the simulations were implemented on MATLAB and run on a computer with 16GB RAM and 2.8GHz processor Core i7 (7th Gen).

4-1 One Dimension (Pure Longitudinal Scenario)

In this section, the reachable set in one dimension is computed and discussed, based on the procedure discussed in Section 3-2-2 for pure longitudinal motion in a static scenario. Then, a sensitivity analysis is also performed to analyse the variation of the avoidance metric which is constructed based on the computed reachable set. Later, the avoidance metric is computed for pure longitudinal motion in dynamic scenarios.

Finally, the avoidance metric computed through the procedure is compared via simulation with BTN for pure longitudinal motion in both static and dynamic scenarios.

4-1-1 Static obstacle

Scenario 1: This scenario consists of a host vehicle initially travelling at a speed of $v_h = 20\,[m/s]$ and the obstacle is at $x_{obs} = 40\,[m]$ from the host vehicle as shown in Figure 4-1.

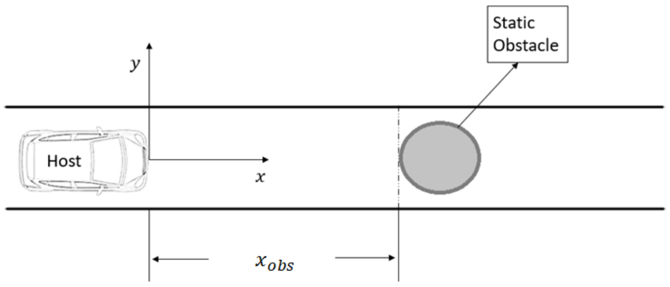


Figure 4-1: Scenario 1: One dimensional motion where the host vehicle has $v_h = 20\,[m/s]$ and obstacle is at a distance $x_{obs} = 40\,[m]$ from the host vehicle.

Parameters	**Value**
Discretisation interval for acceleration, $\mathbf{a_{int}}$	$2.5\,[m/s^2]$
Maximum (and Minimum) acceleration, $\lvert\mathbf{a_{l,max}}\rvert$	$10\,[m/s^2]$
Discretisation interval for interpolation, $\delta_\mathbf{s}$	0.0075 [-]
Number of time steps, $\mathbf{n}$	10 [-]
Time step, $\mathbf{\Delta t}$	0.25 [s]

Table 4-1: Parameters used to compute the approximation of the reachable set in pure longitudinal motion (1-D) for the host vehicle.

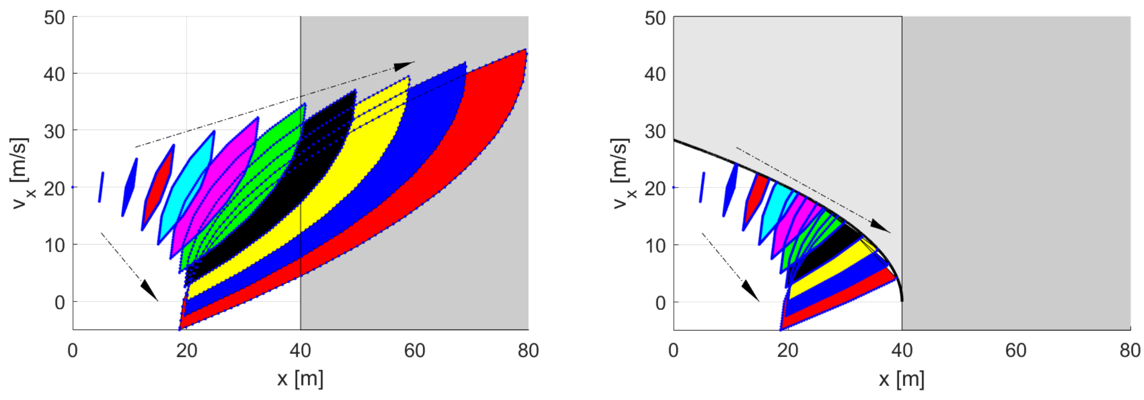
The parameters to compute the approximation of the reachable set are given in table 4-1.

Figure 4-2 shows the reachable set ($\mathbf{R}(t, x_o)$) of the vehicle, for two cases, i) without considering obstacles, and ii) with considering obstacles, by removing the collision states and also the Inevitable Collision States (ICS). The Dark Gray region in Figure 4-2 represents the obstacle region, and hence, the states that lie in this region are the collision states, whereas the states that fall beyond the black thick line represent the ICS. In Figure 4-2b, it can be seen that the black thick line varies quadratically with speed. The dashed black arrows in Figure 4-2 show the direction of growth of the reachable set ($\mathbf{R}(t, x_o)$) for the total time horizon, n. The bottom left part of the reachable set shows the braking region, where at the 9th time step (blue patch), the host vehicle comes to rest (around 20 [m]) and at the 10th time step, the host vehicle begins to move back (< 20 [m]) due to the negative acceleration.

Figure 4-3 shows the variation in the fraction of area ($A^{obs}/A^{w/o-obs}$) with increase in time, where, $A^{obs}, A^{w/o-obs}$ are the area enclosed by the boundary of the reachable set with and *without* the consideration of obstacles (also shown in Figure 4-2).

As it can be seen, the ratio decreases rather quickly with time as the reachable set begins to overlap with the obstacle region. Note that, as mentioned in Section 3-3-1, the avoidance metric is a weighted sum of the the fraction of areas (for 10 time steps), with the weights whose value decreases with time.

The computation of the Avoidance metric as described in subsection 3-26, took around 380 [ms] of time, for a total time horizon of 2.5 [s].



(a) Reachable set *without* accounting for obstacles.

(b) Reachable set by accounting for obstacles and removing the states that belong to Inevitable Collision States (ICS).

Figure 4-2: Reachable set at time-step, $k = 1, 2, .., 10$ for a host vehicle with initial state, $(x^i = (x^i_s, \dot{x}^i_s)^T = (0, 20)^T)$ and obstacle at $x_{obs} = 40\,m$ from the host. Dark Gray: Obstacle region. Light Gray: ICS. Thick Black line (figure 4-2b): Area beyond which, the states fall in ICS. Dashed Black arrows: Direction of growth of the reachable set with increase in number of time steps.

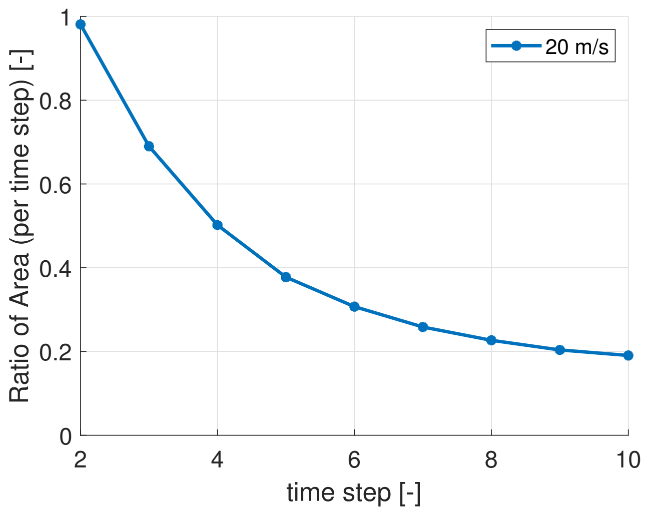


Figure 4-3: Variation in ratio of the area $(A^{obs}/A^{w/o-obs})$ with increase in time, given that the initial velocity $v_h = 20\,[m/s]$, where, A^{obs}, $A^{w/o-obs}$ are the area enclosed by the boundary of the reachable set with and *without* the consideration of obstacles (also shown in figure 4-2).

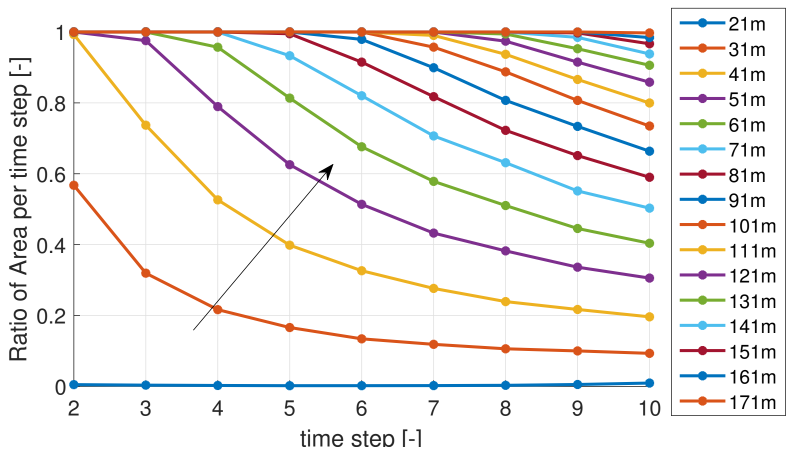


Figure 4-4: Variation in ratio of the area ($A^{obs}/A^{w/o-obs}$) with increase in time, given that the initial velocity $v_h = 20\,[\text{m/s}]$, where, A^{obs}, $A^{w/o-obs}$ are the area enclosed by the boundary of the reachable set with and *without* the consideration of obstacles. Black Arrow: Direction, in which the ratio of the area varies with increase in distance between the host and the obstacle vehicle.

Sensitivity analysis:

Scenario 1 focuses on a particular scenario with a given initial speed and obstacle distance. However, in order to analyse the variation in avoidance metric, a sensitivity analysis needs to be performed since this highlights the general trend of the avoidance metric with the change in obstacle distance or/and host vehicle velocity.

Figure 4-4 shows how the ratio of the area ($A^{obs}/A^{w/o-obs}$) varies with increase in time. It can be noted that as the obstacle distance increases, the ratio of area increases indicating an increase in the avoidance action available at the host vehicle's discretion that avoids the hazard (collision). However, for lower values of obstacle distance, the ratio of area becomes zero, signifying that the collision becomes imminent (i.e. there is nothing that the host vehicle can do, to avoid the collision).

In Figure 4-5, the variation in the Avoidance metric, with an increase in obstacle distance is shown, for different initial velocities. As expected, with an increase in obstacle distance, for a given initial velocity, the Avoidance metric increases and, with an increase in initial speed, the avoidance metric curve moves to the right, implying a decrease in avoidance capability of the host vehicle.

4-1-2 Dynamic obstacle

In case of a dynamic obstacle, the bulk of the process to compute the reachable set remains the same (with the static scenario), except for certain changes. Firstly, the computation of ICS would no longer be trivial. Therefore, *unlike* as shown in Figure 4-2b, only the states that belong in the collision region will be removed. Secondly, in the computation procedure of the Avoidance metric, the ratio of the area enclosed by the boundary of the reachable set with and without the obstacles require the obstacle region. This obstacle region at every time step has to be calculated based on a prediction model of the obstacle's position (constant acceleration

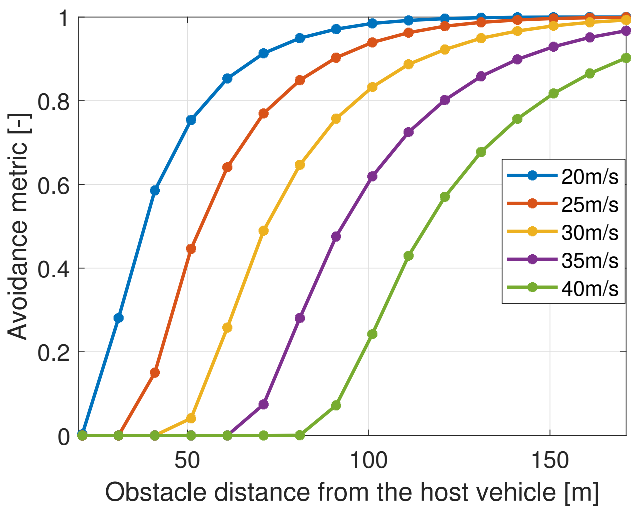


Figure 4-5: Variation in the Avoidance metric with obstacle distance [m] and host vehicle velocity [m/s]

model in this case). Scenario 2 described subsequently considers the above changes and computes the Avoidance metric.

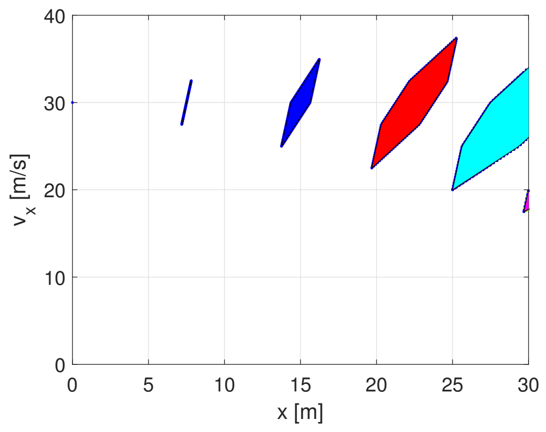
Scenario 2: This scenario consists of a host vehicle and an obstacle vehicle separated by a distance of $8\,m$ initially and travelling with a speed of $v_h = 30\,[m/s]$ and $v_o = 20\,[m/s]$ respectively. The parameters used to compute the approximation of the reachable set is given in table 4-1.

Figure 4-6 shows the reachable set of the host vehicle, for two cases, i) without considering obstacles, and ii) with considering obstacles, by removing the collision states. The Dashed black arrow in Figure 4-6b show the direction of growth of the reachable set for the total time horizon, n. The Vertical black lines (dashed and thick) give the obstacle's position at kth time-step. The obstacle positions are obtained from the prediction model (constant acceleration model). At every time-step, k, the states that lie beyond the obstacle position are removed. It is apparent from Figure 4-6b that after five time-steps, the reachable set shrinks to zero, i.e. despite the acceleration applied, the host vehicle ends up in the collision region. However, the Avoidance metric was found out to be 0.278 [-] and not 0 [-] due to its dependency on the future time steps.

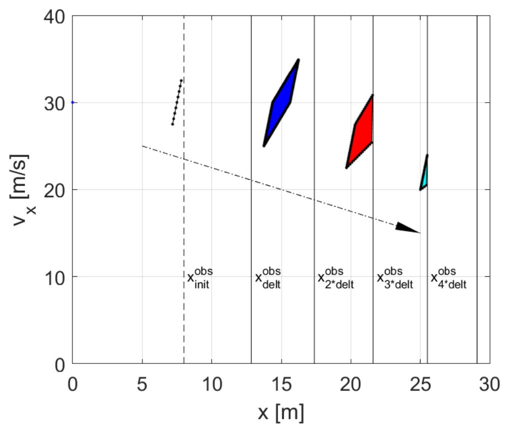
The computation of the Avoidance metric for the dynamic scenario, took around the same time as that of the static scenario (380 [ms]), for a total time horizon of 2.5 [s].

4-1-3 Comparison with BTN

In this subsection, the Avoidance metric devised in Section 3-3-1 for pure longitudinal motion, is compared with BTN whose computation procedure is detailed out in Section 2-2-1.



(a) Reachable set *without* accounting for obstacles.



(b) Reachable set by accounting for obstacle's position and removing the states that belong in the collision region.

Figure 4-6: Reachable set at time-step, $k = 1, 2, .., 5$ for a host vehicle with initial state, $(x^i = (x^i_s, \dot{x}^i_s)^T = (0, 30)^T)$ and obstacle vehicle with initial state, $(x^i = (x^i_{obs}, \dot{x}^i_{obs})^T = (8, 20)^T)$. Dashed Black arrow: Direction of growth of the reachable set with increase in number of time steps. Vertical black lines (dashed and thick in figure 4-6b): Positions of the obstacle vehicle at time-step, $k = 1, 2, .., 5$.

Parameters	**Value**
Initial distance between the host and the obstacle, (with safety margin), $\mathbf{x_{ho}} = (\mathbf{s} - \epsilon)$	$100\,[m]$
Host initial velocity, $\mathbf{v_h}$	$20\,[m/s]$
Host initial acceleration, $\mathbf{a_{l,h}}$	$0\,[m/s^2]$

Table 4-2: Parameters used in the case of static obstacle scenario.

Static obstacle:

In the case of a static obstacle, BTN is calculated using equations 2-4, 2-6 at every time step. When BTN crosses the threshold ($BTN_{thresh} = 1$), it indicates the collision of the host vehicle with the obstacle.

A scenario where the host vehicle is travelling at a constant speed towards an obstacle vehicle is considered. The parameters for the scenario are listed out in table 4-2. The parameters used to compute the Avoidance metric are given in table 4-1.

In Figure 4-7, the variation of the BTN (blue), Avoidance metric (red), with time is shown. Un-avoidance metric (uAM = 1 - AM) is also plotted, to make it easier to compare the differences in BTN and AM. Interestingly, the BTN and uAM cross the threshold at the same period in time. This implies that, beyond 3.9 [s], there is nothing the host vehicle can do to avoid the collision, which is accurately predicted by the Avoidance metric. What is interesting to see is that, in the beginning (0-3 [s]), the Un-avoidance metric was lower than the BTN and somewhere in the middle (3.2 - 3.9 [s]), the Un-avoidance metric goes higher than BTN.

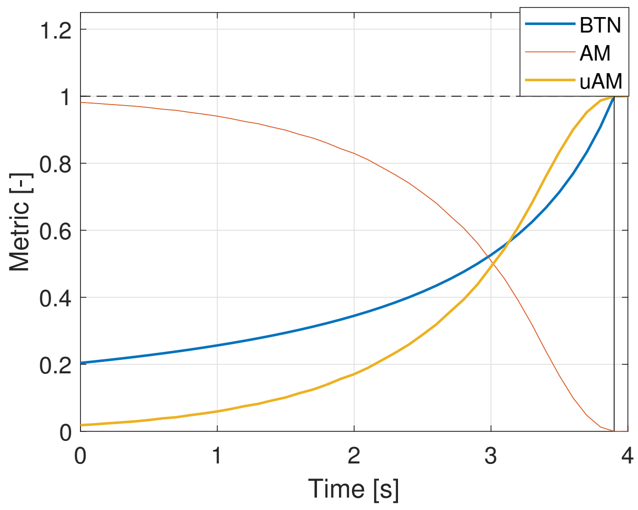


Figure 4-7: Variation of metrics (BTN, AM) for a static obstacle scenario. Dashed black line represents the BTN_{thresh}. 'uAM' is the Un-avoidance metric $(1 - AM)$. Vertical black line represents the point of crossing for both BTN and uAM and also represents the time of collision.

This happens since the Avoidance metric is explicitly dependent on the future time steps, whereas the BTN is not.

The computation of the Avoidance metric (AM) took around 231 [ms] of time, whereas, the computation of the BTN consumed around 0.3 [ms] of time.

Dynamic obstacle:

In the case of a dynamic obstacle, BTN is calculated using equations 2-4, 2-10, 2-16 at every time-step.

A scenario is considered where, the host vehicle and an obstacle vehicle are separated by a distance of $100\,[m]$ initially and travelling with a speed of $v_h = 30\,[m/s]$ and $v_o = 20\,[m/s]$ respectively in the same direction. The obstacle vehicle at time, $t_o = 0$, begins to decelerate at a constant acceleration, $a_{l,o} = -2\,[m/s^2]$. The host vehicle begins to accelerate with a constant acceleration, $a_{l,h} = 10\,[m/s^2]$ at time, $t_o = 0$ until it collides with the obstacle. These parameters are summarised in table 4-3.

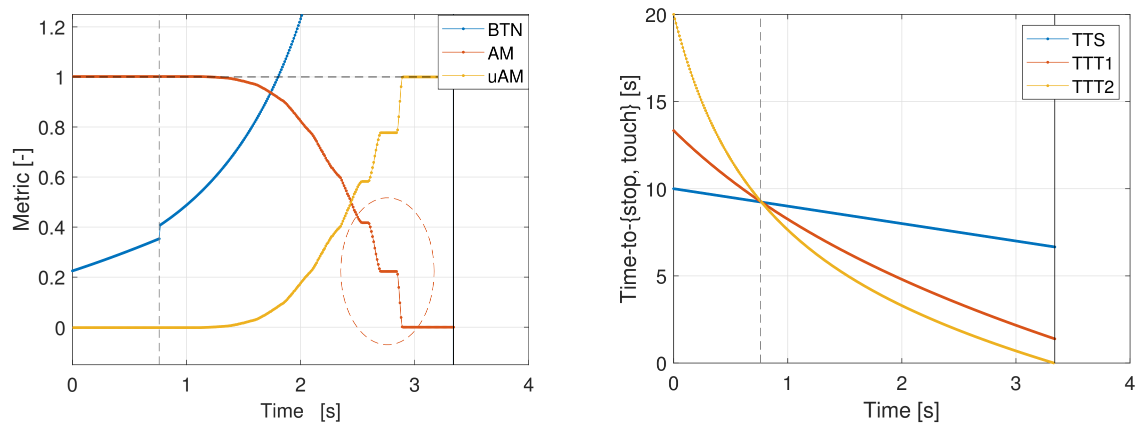
The parameters used to compute the AM are listed out in table 4-1.

In Figure 4-8a, a comparison is made between the variation of BTN (blue) and Avoidance metric (red) with time, for a dynamic obstacle scenario. Three main observations can be made from Figure 4-8a.

Firstly, it can be noted that there exists a step (around 0.8 [s]) in the variation of BTN. This can be attributed to the different computation procedure of BTN based on Time-to-stop (TTS) and Time-to-Touch (TTT). This is further illustrated in Figure 4-8b as the step in BTN variation corresponds to the time when TTT crosses TTS (Vertical dashed gray lines in Figures 4-8a, 4-8b).

Parameters	Value
Initial distance between the host and the obstacle (with safety margin), $\mathbf{x_{ho}} = (\mathbf{s} - \epsilon)$	$100\,[m]$
Host initial velocity, $\mathbf{v_h}$	$30\,[m/s]$
Host acceleration, $\mathbf{a_{l,h}}$	$10\,[m/s^2]$
Obstacle initial velocity, $\mathbf{v_o}$	$20\,[m/s]$
Obstacle acceleration, $\mathbf{a_{l,o}}$	$-2\,[m/s^2]$

Table 4-3: Parameters used in the case of a dynamic obstacle scenario.



(a) Variation of metrics (BTN, AM) for a dynamic obstacle scenario.

(b) Variation of TTS and TTT with time.

Figure 4-8: Vertical black line represents the time of collision. Vertical dashed gray line: Time of discontinuity in BTN. (a) Horizontal Dashed black line represents the BTN_{thresh}. 'uAM' is the Un-avoidance metric $(1 - AM)$. (b) TTT1 and TTT2 are TTT for both the cases involved in the computation of BTN (explained in section 2-2-1). Dashed red ellipse: Step-like variation of AM.

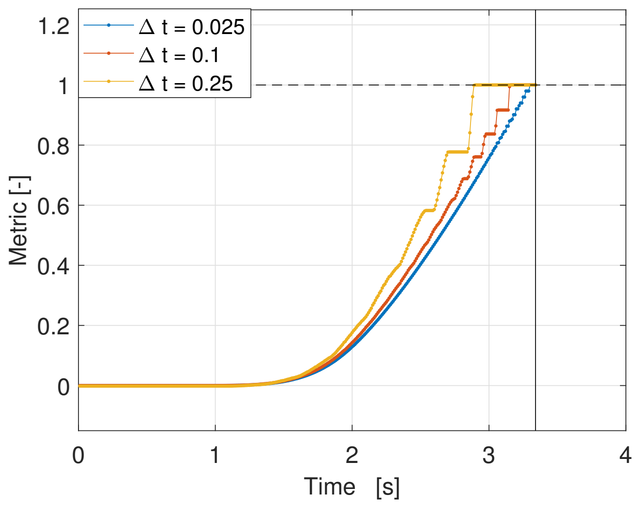


Figure 4-9: Variation of the Un-avoidance metric (uAM) with time.

Δt[s]	**Average computation time** [ms]
0.025	87.22
0.1	17.5
0.25	6.7

Table 4-4: Computation times obtained on varying Δt, keeping the prediction horizon constant at 2.5 [s].

Secondly, few step-like variation (represented by dashed red ellipse in Figure 4-8a) can be seen from the variation of AM. This is due to the value of the parameters (mainly the size of the time step, **Δt** and the number of time steps, **n** whose values are listed out in table 4-1) involved in the computation of AM. Let's consider the the behaviour of AM when its value converges to $\approx$ 0.21 [-] at time interval ($\approx$ 2.66 - 2.83 [s] in Figure 4-8a). This means that the AM takes the same value at different time steps. This mostly happens when the ratio of area ($A^{obs}/A^{w/o-obs}$, *similar* to Figure 4-3) remains the same for those time instants. Or the combination of the sum of the areas remain the same at different time steps (highly unlikely). A possible explanation for the ratio staying the same is that for the given prediction horizon ($\Delta t{*}n$), the value of Δt is *not* small enough.

To clarify this, the variation of AM with time, is plotted for different values of Δt for the same scenario in Figure 4-9. The total prediction horizon ($\Delta t{*}n$) is kept constant at 2.5 [s]. It can be seen that on decreasing the value of Δt, the step-like behaviour increases, but takes the same value for smaller intervals of time. However, this comes at the expense of computation power, as presented in table 4-4.

Thirdly, the difference between the performance of BTN and AM is quite large as seen from Figure 4-8a. A possible explanation for this might be the exclusion of the collision states and *not* the Inevitable collision states (like in the static scenario). Also, unlike the static scenario, the Avoidance metric doesn't cross the threshold at the same time as BTN.

BTN as a metric quantifies the avoidance action of the host vehicle by looking at the whole

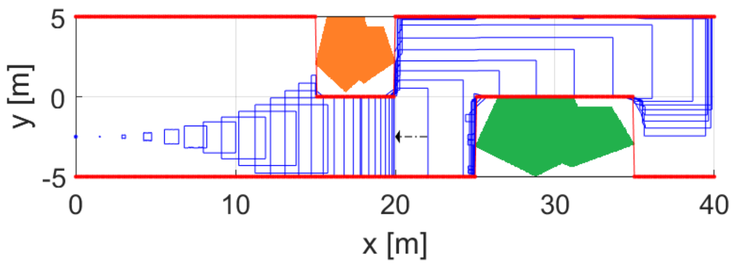


Figure 4-10: Approximation of the boundary of the reachable set of the host vehicle in the output space (x, y) for 25 time-steps with considering obstacles (orange and green). Red: New lane boundaries (maximum and minimum lane limits) in presence of obstacles. Black dashed arrow: Braking region.

Parameters	**Value**
Host initial position in 'x', $\mathbf{x,init}$	$0\,[m]$
Host initial position in 'y', $\mathbf{y,init}$	$-2.5\,[m]$
Host initial velocity in 'x', $\mathbf{v_{x,init}}$	$20\,[m/s]$
Host initial velocity in 'y', $\mathbf{v_{y,init}}$	$0\,[m/s]$

Table 4-5: Initial state of the host vehicle for a static obstacle scenario in two dimensions (both lateral and longitudinal).

time horizon and assuming that the future motion of the obstacle is perfectly known. However, in real-life scenarios, this is not the case and, models like constant acceleration model can only predict the future motion of the obstacle, for utmost 2-3 [s]. In contrast, the AM quantifies the avoidance action by looking into the future for a certain time horizon and identifying the set of states that the vehicle can be at every time step for the considered time horizon.

4-2 Two Dimensions (Lateral and Longitudinal Scenario)

In this section, the reachable set in two dimensions is computed based on the procedure discussed in Section 3-2-3 for the combination of lateral and longitudinal motion in a static obstacle scenario. Subsequently, the computed approximation of the reachable set is compared with the nodes obtained from RRT to examine the accuracy of the approximation of the reachable set.

4-2-1 Static scenario

Scenario 3: This scenario consists of a host vehicle initially travelling at a speed of $v_{x,init} = 20\,[m/s]$ and two static obstacles at certain distance from the host vehicle as shown in Figure 3-9.

The parameters that describe the host vehicle parameters are given in table 4-5, while the parameters required to compute the approximation of the reachable set is given in table 4-6 .

Parameters	**Value**
Discretisation interval for acceleration, $\mathbf{a_{int}}$	2.5 $[m/s^2]$
Maximum (and Minimum) acceleration, $\lvert\mathbf{a_{max}}\rvert$	10 $[m/s^2]$
Discretisation interval for interpolation, $\delta_\mathbf{s}$	0.0125 [-]
Number of time steps, $\mathbf{n}$	25 [-]
Time step, $\mathbf{\Delta t}$	0.075 [s]
Time horizon, $\mathbf{T}$	1.875 [s]

Table 4-6: Parameters used to compute the approximation of the reachable set for the host vehicle in both lateral and longitudinal motion (2-D).

Figure 4-10 shows the boundary of the approximation of the reachable set ($\mathbf{R}(t, x_o)$) on considering two static obstacles for 25 time-steps. The *boundary* function is used on MATLAB along with a tightening factor of 0.7 [-] to obtain the boundary of the approximation of the reachable set.

It can be observed from the figure that the boundary of the reachable set for the last time-step on the left, starts from $x = 20\,[m]$ which corresponds to the position where the host vehicle would be, on applying pure braking action (illustrated by black dashed arrow).

Figure 4-12 shows the variation in the fraction of area ($A^{obs}/A^{w/o-obs}$) with increase in time, where, $A^{obs}, A^{w/o-obs}$ are the area enclosed by the boundary of the reachable set with and *without* the consideration of obstacles (also shown in Figure 4-11).

The initial dip in the ratio is caused due to the tightening factor of the boundary function. Also, the ratio decreases with time. This occurs as with increase in time, the boundary of the reachable set begins to overlap with the obstacle region. Note that, as mentioned in Section 3-3, the avoidance metric is a weighted sum of the ratios, with the weights whose value decreases with time. The avoidance metric (weighted sum of the fraction of areas for 25 time steps) for Scenario 3 was found to be 0.793 [-].

The computation of the Avoidance metric as described in subsection 3-3-2, took around 10.87 [s] of time, for a total time horizon of 1.875 [s] and the total number of samples obtained for Scenario 3 is 157546 [-].

Sensitivity to the number of obstacles:
It is imperative to examine the dependence of the computation time on the number of obstacles. On average, the computation time remained constant with a decrease in the number of obstacles, from 3 (as seen in Scenario 3) to 0 (only lane boundaries are present in the scenario). The same effect on the computation time (unaffected) was found even on increasing the number of obstacles from 3 to 5.

This behaviour is explained by the choice of representation of the obstacles, in the proposed algorithm. Since the y-limits are parameterised as a function of 'x', the presence of obstacles do not have any effect on the number of operations performed by the collision check function in Subsection 3-2-3. This comes as a desirable attribute for the proposed algorithm since, on typical road scenarios, the number of obstacles varies at every time-step and, this does not affect the computation time of the avoidance metric.

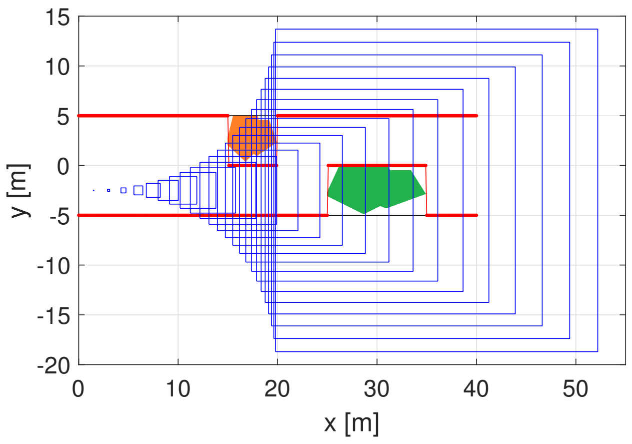


Figure 4-11: Approximation of the boundary of the reachable set of the host vehicle in the output space (x, y) *without* considering obstacles (orange and green), for 25 time-steps. Red: New lane boundaries (maximum and minimum lane limits) in presence of obstacles.

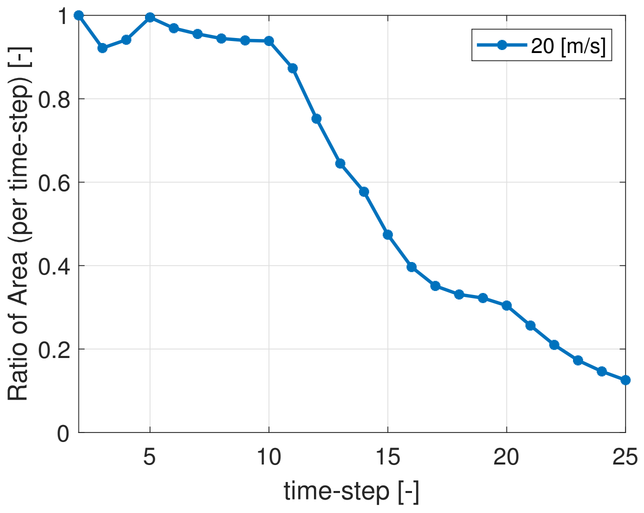


Figure 4-12: Variation in ratio of the area $(A^{obs}/A^{w/o-obs})$ with increase in time, given that the initial velocity $v_{x,init} = 20\,[m/s]$, where, A^{obs}, $A^{w/o-obs}$ are the area enclosed by the boundary of the reachable set in the output space (x, y) with and *without* the consideration of obstacles.

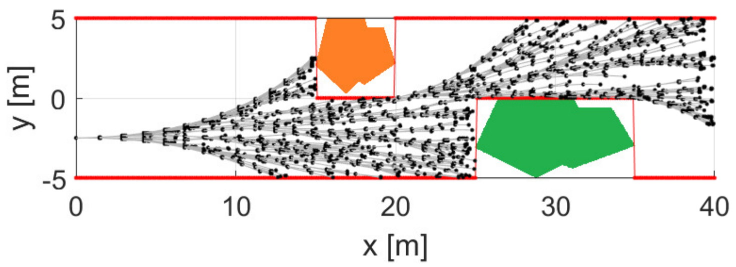


Figure 4-13: Rapidly-exploring Random Tree (RRT) algorithm run in an environment with obstacles (orange and green) for a dynamical system with double integrator dynamics. Red: New lane boundaries (maximum and minimum lane limits) in presence of obstacles. Black dots: Nodes of the RRT. Grey: Edges connecting every pair of nodes of the RRT.

So far, this section has focused on the implementation and the computational complexity of the algorithm to obtain the approximation of the boundary of the reachable set. The following subsection will examine the performance of the algorithm, by comparing the boundary of the approximation of the reachable set to the under-approximation computed from Rapidly-exploring Random Trees (RRTs).

4-2-2 Comparison with Rapidly-exploring Random Tree (RRT)

In this subsection, the boundary of the approximation of the reachable set is compared with an under-approximation. The under-approximation of the reachable set is computed using Rapidly-exploring Random Trees (RRT) for a dynamical system with double integrator dynamics [27], [28]. The description and construction of the RRT is detailed out in Appendix B.

In Figure 4-13, RRT algorithm is run in an environment with obstacles for a dynamical system with double integrator dynamics. There exists a total of 6000 nodes, where the time taken to steer the host vehicle from one node to an other is equal to the time-step, $\mathbf{\Delta t}$ considered in computing the approximation of the reachable set in the previous subsection.

Figure 4-14 presents both the reachable set obtained from the proposed algorithm (Algorithm 3) and RRT. However, to compare them, Figure 4-15 is plotted, which shows the boundary of the approximation of the reachable set and RRT at six subsequent time-steps. As can be seen, the nodes obtained through RRT lie either well inside the boundary or very close to the boundary, which is obtained from the algorithm proposed in Section 3-2-3 to approximate the reachable set.

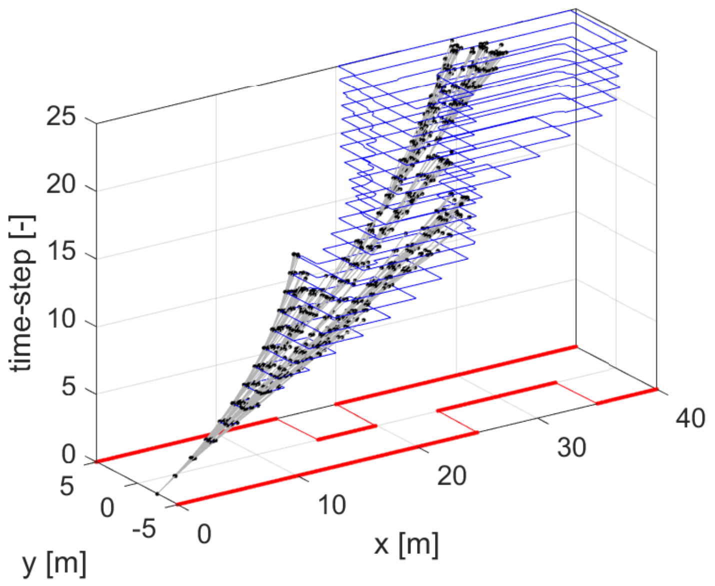
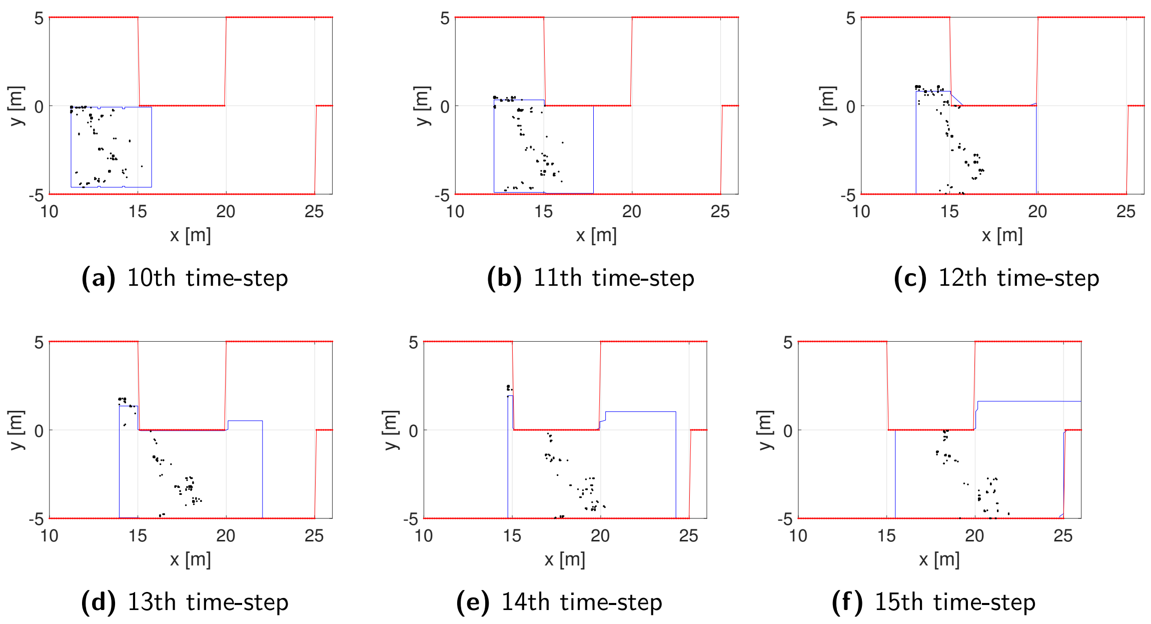


Figure 4-14: Black and Grey: Under-approximation of the reachable set of the host vehicle constructed through RRT at every time-step. Blue: Boundary of the approximation of the reachable set constructed using the proposed algorithm (Algorithm 3).



(a) 10th time-step **(b)** 11th time-step **(c)** 12th time-step

(d) 13th time-step **(e)** 14th time-step **(f)** 15th time-step

Figure 4-15: Blue: Boundary of the approximation of the reachable set at six subsequent time-steps. Black: Under-approximation of the reachable set obtained through RRTs. Red: Lane boundaries.

Chapter 5

Conclusion and Future Work

5-1 Summary and Conclusion

Chapter 1 motivates and introduces the problem statement for this thesis report. Chapter 2 begins with the ideal properties for an avoidance metric and then, proposes few candidate metrics to quantify the avoidance capability. Firstly, BTN, a threat measure, that can also be used as an avoidance metric for pure longitudinal scenario is introduced, and its computational procedure is detailed out. However, this metric could not be extended to both the lateral and longitudinal scenario since there does not exist one such analytic way to compute the metric. Therefore, a metric based on constant control for fixed horizon was briefly described, that generates a set of trajectories by applying constant control inputs (sampled at constant angles from the Kamm's circle) to the vehicle model for a fixed time horizon and then, checks for collision with the obstacle vehicle. This metric, however, does not account for the varying control inputs that can avoid the hazard. Next, a metric based on motion primitives was put forth that generates trajectories by recursively applying the motion primitives. However, this procedure was found to be computationally expensive and intractable for certain cases.

Therefore, instead of exploring the allowable input space like the previous metrics, Chapter 3 introduces the concept of reachability for a dynamical system as a means to construct a new avoidance metric based on the output reachable states. Novel analytical procedures are provided to compute the reachable set of a double integrator model in the absence of obstacles for a pure longitudinal motion. An approximation method, based on half-planes is employed, to obtain the reachable set of one-dimensional motion in the presence of obstacles. This approximation method is used in developing a computation procedure (algorithm) to obtain the boundary of the reachable set (in the *output space* of the system model) for a pure longitudinal scenario. This computation procedure is extended to the combination of lateral and longitudinal scenario to obtain the approximation of the reachable set of the outputs of the system model. The approximate reachable sets are then used to devise avoidance metrics (pure longitudinal and combination of lateral and longitudinal) of the host vehicle to avoid the hazard.

Chapter 4 elaborates on the implementation part of the computation procedures described in Chapter 3. In a pure longitudinal scenario, the avoidance metric is computed for both static and dynamic scenario and, is also compared with BTN. The avoidance metric was found to show expected behaviour with a change in initial speed and distance between the obstacle. On comparison with BTN, in a static case, it was observed that the metric shows similar behaviour to that of BTN. However, in the case of a dynamics scenario there were three main differences between the behaviour of BTN and the avoidance metric. Firstly, unlike the avoidance metric, a discontinuity was observed in the BTN metric at the time instant when TTT crosses TTS. Secondly, a step like variation was observed in the behaviour of avoidance metric which can be attributed to the step-size of the time interval. Thirdly, the difference between the performance of BTN and avoidance metric was large. A possible explanation for this behaviour is the exclusion of only the collision-states and not the ICS.

In the case of the combination of lateral and longitudinal scenarios, the computation procedure was used to quantify the avoidance capability. Furthermore, it was also observed that the computation times for the proposed procedure was independent of the number of obstacles. Next, the approximation of the reachable set provided by the computation procedure was compared with the under-approximation provided by RRT. The nodes obtained through RRT were found to lie either well inside the boundary or very close to the boundary obtained from the proposed computational procedure.

5-1-1 Conclusions:

The main goal of this thesis was to describe the ability of a vehicle to avoid a collision. This is achieved by developing an avoidance metric by the means of computational procedures to quantify the avoidance capability of the vehicle for both pure longitudinal (1-D) and, lateral and longitudinal (2-D) scenario for a given vehicle model, in the presence of hazards.

The following conclusions can be drawn from the present study:

1. In the case of the combination of both lateral and longitudinal manoeuvres, the metric from an exhaustive search method based on motion primitives, was found to intractable, on increasing the number of stages, n. This is important since on increasing the number of stages, the computed allowable input commands approach the actual allowable input commands to avoid the hazard (*static scenario*) for the considered time horizon, thereby, increasing the accuracy of the metric.

2. In the case of pure longitudinal manoeuvres, BTN is computed considering the whole time horizon, while the proposed avoidance metric is computed by only looking into the future for a couple of seconds. Despite this, in the presence of a static obstacle, the computed avoidance metric showed similar performance to that of BTN. Moreover, the computation time for the avoidance metric (*static scenario*) was found to be around 380 [ms] for a time horizon of 2.5 [s] implying that this can be computed real-time on a vehicle.

   However, in the presence of a dynamic obstacle with perfectly known behaviour, the variation of the proposed avoidance metric shows a huge difference with the variation of BTN. One of the main reasons for this behaviour is due to the exclusion of only the collision states and not ICS, since computation of ICS is not trivial unlike the static scenario. This comes as a major drawback for the proposed avoidance metric.

3. In the case of combination of both lateral and longitudinal manoeuvres, the proposed avoidance metric is computed based on the size of the boundary of the approximation of the reachable set. Whilst this thesis did not confirm the level of approximation of the reachable set for a double integrator model (*due to the absence of ground truth*), it did partially substantiate on the accuracy of the approximation, by comparing the approximation with the nodes obtained from RRT.

   The proposed computational procedure has two desirable attributes. Firstly, the total number of samples remain bounded irrespective of the type of road scenario. Secondly, the computation time is unaffected by the number of obstacles present in the road scenario.

   However, the main drawbacks of the proposed avoidance metric are (1) the computation time which took around 10.87 [s] for a time horizon of 1.875 [s] and, (2) only valid in the presence of static obstacles.

5-2 Future Work

In this thesis, two computational procedures were developed to compute the avoidance metric in (1) pure longitudinal and, (2) lateral and longitudinal manoeuvres. However, there still exists several open questions and challenges to be solved. A few of them are listed below:

1. *Lateral and longitudinal scenario (static case)*

   The proposed computational procedure to obtain the boundary of the approximation of the reachable set does not exclude the ICS. A further area of research would be to incorporate the procedure to exclude the ICS in the computational procedure. This would result in the reduction of the approximation of the reachable set to the reachable set of a double integrator system model. The procedure to compute ICS is put forward in [26]. Informally speaking, the procedure involves growing the obstacles based on the system dynamics which result in the ICS set[1] (see [26] for more details). Incorporation of this procedure would particularly be interesting since this would potentially streamline the algorithm making it faster (positively impacting the computation time) and accurate.

2. *Computation time for the proposed algorithms*

   The major drawback of the proposed computational procedure is the computation time of around 10.87 [s] for a time horizon of 1.875 [s]. Since the procedures are implemented in MATLAB, the computation times calculated are very dependent on the overhead[2]. Furthermore, not all algorithms are optimized through LAPACK/SLICOT routines. Therefore one could think of implementing the proposed procedures in a different programming language with/without framework and then, compare the computation times with the ones obtained in this thesis.

[1]Backwards Minimal Reachable set [29] of the obstacle region.

[2]Overhead can come from two main places. One is “first time costs" which includes parsing and optimizing program files the first time they are called. The second is from the function call overhead.

3. *Validation with escape trajectories*

   The proposed computational procedure to compute the avoidance metric has not been tested on candidate escape trajectories. Therefore, an area of future work would be to test it on candidate escape trajectories to investigate if the variation in the avoidance metric can provide any insight on traffic safety by itself. Furthermore, the variation of a combination of factors such as the distance between predicted states of the host and obstacle vehicle, the avoidance metric can also be assessed on multiple candidate trajectories to examine if the combination as such can be used to provide insights into traffic safety.

4. *Motion planning in dynamic scenarios using the approximation of the reachable set obtained through the proposed procedure*

   Another area of possible future work would be to employ the proposed procedure to compute the approximation of the reachable set for all the obstacle vehicles present in the road scenario. These reachable sets can be used to plan candidate escape trajectories in the complement of the union of the obstacle's reachable sets (Reachable sets obtained from the proposed procedure).

5. *Accounting for uncertain prediction using probabilistic occupancies*

   Currently, the proposed procedure has assumed that the position, velocity of the obstacle vehicle[3] obtained from the sensors are accurate. However, this is seldom the case. This warrants the use of probabilistic occupancies (by specifying the uncertainties in sensor data) to reason about the obstacle vehicle's position. This probabilistic occupancy can be employed to extend the proposed procedure. This results in probabilistic level sets of the boundary of approximation of the reachable set of the host vehicle. However, with probabilistic approximation of the reachable set, a new formulation of the avoidance metric would be required.

6. *Validation of the proposed algorithm with an over-approximation algorithm*

   The boundary of the approximation of the reachable set for a double integrator system model is compared with an under-approximation, i.e. the nodes obtained from the RRT. A natural progression of this work would be to compare the level of approximation from the proposed computational procedure to that of an over-approximation considering double integrator system model from [21]. This would be interesting since [21] obtained a lower level of maximum over-approximation and hence, help in quantifying the level of approximation from the proposed procedure.

7. *New application in Sampling-based motion planning*

   The computation procedure to approximate the reachable set opens up a new application in Sampling-based motion planning. Based on the time-step, a new sample can be chosen randomly inside the boundary of the approximation of the reachable set to grow the tree (Refer Appendix B). This step increases the number of collision-free trajectories that can be constructed on the tree since, all the samples that are sampled from the computed boundary already account for the static obstacles present in the scenario.

[3]just the position in the case of lateral and longitudinal scenario.

8. *Dynamic scenario*

   In this work, the computation procedure for lateral and longitudinal scenario has been implemented only for a static scenario. An extension would be develop a computational procedure on similar lines assuming a prediction model for the dynamic obstacle vehicle. As the *forward propagation* in the computational procedure is done in the phase plane (position, velocity), a cluster analysis would be required to select the states to propagate. Algorithms such as Expectation-maximisation [30] can be utilised to select the states to be propagated.

Appendix A

Collision Check

This chapter describes the collision-check algorithm utilised in Sections 2-2-2, 2-2-3, and Appendix B based on [2, Section 3.1.1]. The first section describes the representation of the obstacle region. The second section moves on to use the obstacle representation, to decide whether a point, (x, y) lies inside the obstacle region.

A-1 Obstacle Representation

The obstacle region can be represented in several ways. This is usually done by bounding the obstacle region with geometrical shapes such as a sphere, convex polygon and a non-convex polygon. However, in this case, as the obstacles are common road-obstacles (cars, road-works, cows, etc.) convex polygons are used to bound the obstacle region.

A-1-1 Convex polygon

A convex polygon, A is a polygon if for any two chosen points, $(A_1,\, A_n)$ inside the polygon, all the points that lie on the line $(A_1,\, A_2,\, \ldots A_n)$ connecting the two chosen points also lie inside the polygon.

The polygon can be described by a sorted (clockwise) order, $(x_1,\, y_1), (x_2,\, y_2), \ldots (x_n,\, y_n)$. The obstacle region can be represented as an intersection of n half-planes as shown in Figure A-1. Each half plane can be represented by obtaining the equation of the line connect the two points. For example, let's consider two points, $(x_1,\, y_1), (x_2,\, y_2)$. The equation of the line connecting these two points is of the form, $ax + by + c = 0$, where, a, b, c are constants based on $x_1,\, y_1,\, x_2\, y_2$. Let's define a function 'f' such that $f(x, y) = ax + by + c$. The function, $f(x, y) = 0$ if the point (x, y) lies on the line connecting the points, $(x_1,\, y_1), (x_2,\, y_2)$. However, if the point does not lie on the line, then the function, $f(x, y) < 0$, or $f(x, y) > 0$, based on the which side of the line, the point lies on. Therefore, all the points that lie on one side of the line (like illustrated in Figure A-1 (b)) can be represented by $f(x, y) < 0$. This region is known as a half-plane.

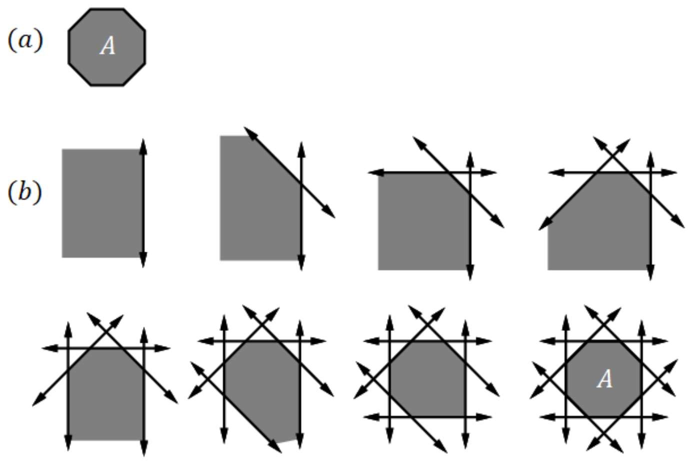


Figure A-1: (a) Obstacle region, 'A'. (b) Representation of the obstacle region, 'A' through multiple half-planes [2].

If there exists multiple (i) pair of points, then multiple functions, $f_i(x, y)$ can be obtained. Then, the convex region (Figure A-1(a)) can be obtained by a intersection of n half-planes. An n sided convex polygon (obstacle region) can be expressed as given in equation A-1.

$$\begin{aligned} A &= \bigcap_{i}^{n}\Big(f_i(x,y) < 0\Big) \\ &= (f_1(x,y) < 0) \cap (f_2(x,y) < 0) \cap \ldots. \cap (f_n(x,y) < 0) \end{aligned} \tag{A-1}$$

A-2 Boolean Value for Collision Check

The main objective is to determine whether a point, (x, y) is in the collision-free or collision region. The set representation used in Section A-1, equation A-1 can be transformed into a combinations of logicals (ORs, ANDs), resulting in equation A-2.

$$B = (f_1(x,y) < 0) \wedge (f_2(x,y) < 0) \wedge \ldots. \wedge (f_n(x,y) < 0) \tag{A-2}$$

Note that in equation A-2, the unions are replaced by logical AND operators. Therefore, if a point (x, y) belongs inside the convex region, A, then the boolean value of B is *True*.

However, in most scenarios, there exist multiple obstacles (m). Then, a logical combination of ANDs can be used to obtain if a point (x, y) lies in any of the obstacles. This is given in equation A-3.

$$C = B_1 \vee B_2 \vee \ldots. \vee B_m \tag{A-3}$$

If the point (x, y) lies in any of the obstacles, then the boolean value of C is *True* and, *False* if otherwise.

■

In this thesis report, the obstacle (vehicle) is represented with a rectangle. Therefore, n is 4 in equation A-1.

Appendix B

Description for Rapidly-exploring Random Tree (RRT)

This chapter describes the RRT algorithm employed in Chapter 4 for comparison, is partly based on [27]. The first section specifies the motion model and, describes the steering procedure to steer from one state to another. The second section details the procedures utilised by the main algorithm, RRT.

B-1 Motion Model

A double integrator model (just like in the proposed algorithm) is used to describe the system dynamics (given in equation 3-5).

B-1-1 Steering procedure

A steering procedure was proposed in [28] to steer the system from $z_1 = (x_{1s}, \dot{x}_{1s}, y_{1s}, \dot{y}_{1s})$ to $z_2 = (x_{2s}, \dot{x}_{2s}, y_{2s}, \dot{y}_{2s})$.

In brief, the procedure obtains an optimal steering procedure for each axis individually, i.e., from $(x_{1s}, \dot{x}_{1s}) \rightarrow (x_{2s}, \dot{x}_{2s})$ and $(y_{1s}, \dot{y}_{1s}) \rightarrow (y_{2s}, \dot{y}_{2s})$. The optimal steering procedure for each of the axis is already discussed in Section 3-2-2. The optimal procedure can be a set of four control actions: (1) acceleration and then complete deceleration (2) deceleration and then complete acceleration (3) complete acceleration (4) complete deceleration. There is a possibility that the none of the four control action provide the steering procedure due to the upper bound on the maximum acceleration and deceleration.

Let t_x be the time taken in x-axis and t_y in y-axis. If $t_x < t_y$, then the maximum acceleration in x-axis ($|a_{max,x}|$) is reduced such that $t_x \approx t_y$ and the steering procedure is obtained subsequently. $|a_{max,x}|$ is found by applying a binary search over the range of a.

Since the RRT developed in this thesis is mainly used for comparison with the proposed method (Section 3-2-3), the time taken to steer from one state to another must be equal to the time-step ($\mathbf{\Delta t}$).

In order to make both $t_x, t_y = \mathbf{\Delta t}$, the second state, z_2 to which it should steer, is checked if it lies in the reachable set for time, $\mathbf{\Delta t}$. The bounds for the reachable set can be obtained from Chapter 3 (specifically equations 3-12, 3-13 and 3-14, 3-15). If the state, z_2 is within the reachable set, then a steering procedure can be obtained based on the above described procedure. However, if z_2 lies outside the reachable set, then the state closest (Euclidean distance) to z_2 is chosen as the new z_2.

B-2 RRT Algorithm Procedures:

This section describes the procedures chosen for RRT considering non-holonomic constraints.

Sampling: The sampling procedure Sample : $\mathrm{N} \rightarrow X_{free}$ returns independent and identically distributed samples from the obstacle-free space. A pseudo-random generator is used in MATLAB to sample the state $(x_s, \dot{x}_s, y_s, \dot{y}_s)$.

Distance function: This function measures the closeness between any two given states, z_1, z_2. The distance function used in this algorithm is a Euclidean distance measure as given in equation B-1.

$$J = ((x_{1s} - x_{2s})^2 + (y_{1s} - y_{2s})^2)^{\frac{1}{2}} \tag{B-1}$$

Nearest Neighbour: Given a graph $G = (V, E)$ on X_{free} and a state $z \in X$, the procedure Nearest(G, z) returns the vertex $v \in V$ that is closest to z, by minimising the distance function J.

Local Steering: Given two states $z_1, z_2 \in X$, the Steer procedure returns the optimal trajectory starting at z_1 and ending at z_2 in some local neighbourhood. The steering procedure for the non-holonomic robot is returned by making use of a double integrator model (exact steering) discussed in Subsection B-1-1.

Collision Check: This function checks for collision, given the whole trajectory from z_1 to z_2. The ObstacleFree procedure returns *false* if at least some part of the trajectory lies in the obstacle region. The details of this function is discussed in Appendix A.

Algorithm 4 Algorithm for RRT

```
1: $V \leftarrow z_{init}; E \leftarrow \{\}; i \leftarrow 0;$
2: while i < N do
3:     $G \leftarrow (V, E);$
4:     $z_{rand} \leftarrow$ Sample($i$); $i \leftarrow i + 1;$
5:     $(V, E) \leftarrow$ Extend($G, z_{rand}$);
6: end while
```

Algorithm 5 Extend RRT

```
1: $V' \leftarrow V; E' \leftarrow E$;
2: $z_{nearest} \leftarrow \text{Nearest}(G, z)$;
3: $(x_{new}, u_{new}, T_{new}) \leftarrow \text{Steer}(z_{nearest}, z)$;
4: $z_{new} \leftarrow x_{new}(T_{new})$;
5: **if** ObstacleFree($x_{new}$) **then**
6:     $V' := V' \cup z_{new}$;
7:     $E' \leftarrow E' \cup (z_{nearest}, z_{new})$;
8: **end if**
9:     **return** $G' = (V', E')$
```

Appendix C

Vehicle model

C-1 Point-mass model

The dynamic equations that describe the point mass model in an inertial frame are given in equations C-1 to C-4 [1].

$$\dot{v}(t) = a\,cos(\gamma) \tag{C-1}$$

$$\dot{\psi}(t) = min\left(\frac{a\,sin(\gamma)}{v(t)}, \frac{v(t)}{r_{min}}\right) \tag{C-2}$$

$$\dot{x}(t) = v(t)\,cos(\psi(t)) \tag{C-3}$$

$$\dot{y}(t) = v(t)\,sin(\psi(t)) \tag{C-4}$$

where, x, y are the positions, v, the velocity in the inertial frame, ψ, orientation angle w.r.t the global X axis, a, is the acceleration of the vehicle, r_{min} is the minimum turning radius of the vehicle, γ, angle subtended by the acceleration vector with the global X axis as illustrated in Figure 2-1. The control commands for the point-mass model is the acceleration, a of the vehicle.

Bibliography

[1] N. Kaempchen, B. Schiele, and K. Dietmayer, “Situation assessment of an autonomous emergency brake for arbitrary vehicle-to-vehicle collision scenarios,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 4, pp. 678–687, 2009.

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

[3] European Road Safety Observatory, “Traffic Safety Basic Facts 2017,” tech. rep., European Road Safety Observatory, 2017. [Online]. Available at https://ec.europa.eu/transport/road_safety/sites/roadsafety/files/pdf/statistics/dacota/bfs2017_motomoped.pdf. [Accessed Aug. 18, 2019].

[4] Official Journal of the European Union, “REGULATION (EC) No 661/2009 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL,” tech. rep., Official Journal of the European Union, 2009. [Online]. Available at http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:200:0001:0024:en:PDF. [Accessed Aug. 18, 2019].

[5] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, “A survey of motion planning and control techniques for self-driving urban vehicles,” *IEEE Transactions on intelligent vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[6] L. Fletcher, S. Teller, E. Olson, D. Moore, Y. Kuwata, J. Leonard, I. Miller, M. Campbell, A. Nathan, and F. R. Kline, “The MIT - Cornell Collision and Why it Happened,” *Journal of Field Robotics*, vol. 25, no. 10, pp. 775–807, 2008.

[7] A. Tejada and M. J. E. Legius, “Towards a Quantitative "Safety" Metric for Autonomous Vehicles,” No. 19-0012, pp. 1–15. [Online]. Available at https://www-esv.nhtsa.dot.gov/Proceedings/26/26ESV-000012.pdf. [Accessed Aug. 18, 2019].

[8] M. Althoff and J. M. Dolan, “Online verification of automated road vehicles using reachability analysis,” *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 903–918, 2014.

[9] D. Heß, M. Althoff, and T. Sattel, "Comparison of trajectory tracking controllers for emergency situations," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, pp. 163–170, IEEE, 2013.

[10] D. Ferguson, T. M. Howard, and M. Likhachev, "Motion planning in urban environments," *Journal of Field Robotics*, vol. 25, no. 11-12, pp. 939–960, 2008.

[11] J. Jansson, *Collision Avoidance Theory: With application to automotive collision mitigation.* PhD thesis, Linköping University Electronic Press, 2005.

[12] A. Eidehall, "Multi-target threat assessment for automotive applications," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 433–438, IEEE, 2011.

[13] E. R. Gelso and J. Sjoberg, "Consistent threat assessment in rear-end near-crashes using btn and ttb metrics, road information and naturalistic traffic data," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 74–89, 2017.

[14] G. Lafferriere, G. J. Pappas, and S. Yovine, "A new class of decidable hybrid systems," in *International Workshop on Hybrid Systems: Computation and Control*, pp. 137–151, Springer, 1999.

[15] G. Lafferriere, G. J. Pappas, and S. Yovine, "Symbolic reachability computation for families of linear vector fields," *Journal of Symbolic Computation*, vol. 32, no. 3, pp. 231–253, 2001.

[16] A. Chutinan and B. H. Krogh, "Computational techniques for hybrid system verification," *IEEE transactions on automatic control*, vol. 48, no. 1, pp. 64–75, 2003.

[17] M. Althoff, *Reachability analysis and its application to the safety assessment of autonomous cars.* PhD thesis, Technische Universität München, 2010.

[18] A. B. Kurzhanski and P. Varaiya, "Ellipsoidal techniques for reachability analysis," in *International Workshop on Hybrid Systems: Computation and Control*, pp. 202–214, Springer, 2000.

[19] M. Althoff, "Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets," in *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pp. 173–182, ACM, 2013.

[20] E. Asarin, T. Dang, G. Frehse, A. Girard, C. Le Guernic, and O. Maler, "Recent progress in continuous and hybrid reachability analysis," in *2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control*, pp. 1582–1587, IEEE, 2006.

[21] S. Söntges and M. Althoff, "Computing the drivable area of autonomous road vehicles in dynamic road scenes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 6, pp. 1855–1866, 2017.

[22] M. Althoff, "Reachability analysis of large linear systems with uncertain inputs in the krylov subspace," *IEEE Transactions on Automatic Control*, 2019.

[23] Y. Saad, *Iterative methods for sparse linear systems*, vol. 82. Society for Industrial and Applied Mathematics (SIAM), 2003.

[24] T. Fraichard and H. Asama, "Inevitable collision states - A step towards safer robots?," *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.

[25] R. Parthasarathi and T. Fraichard, "An inevitable collision state-checker for a car-like vehicle," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 3068–3073, IEEE, 2007.

[26] A. Lawitzky, A. Nicklas, D. Wollherr, and M. Buss, "Determining states of inevitable collision using reachability analysis," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4142–4147, IEEE, 2014.

[27] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *49th IEEE conference on decision and control (CDC)*, pp. 7681–7687, IEEE, 2010.

[28] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *Journal of guidance, control, and dynamics*, vol. 25, no. 1, pp. 116–129, 2002.

[29] I. M. Mitchell, "Comparing forward and backward reachability as tools for safety analysis," in *International Workshop on Hybrid Systems: Computation and Control*, pp. 428–443, Springer, 2007.

[30] F. D. College and F. Dellaert, "The Expectation Maximization Algorithm," Tech. Rep. GIT-GVU-02-20, College of Computing, Georgia Institute of Technology, 2002. [Online]. Available at https://www.cc.gatech.edu/~dellaert/em-paper.pdf. [Accessed Aug. 23, 2019].