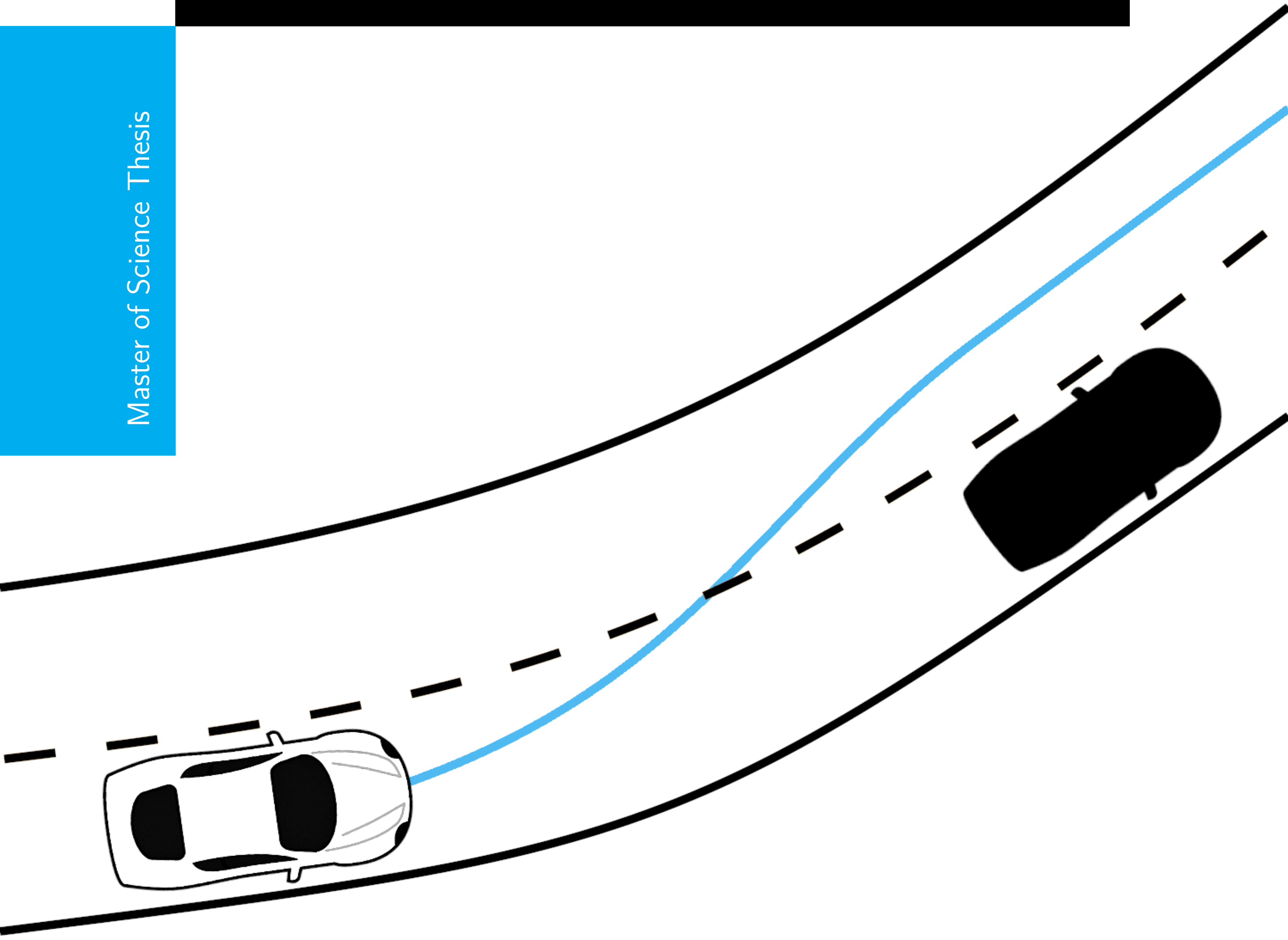
Curvature Aware Motion Planning with Closed-Loop Rapidly-exploring Random Trees

Towards a generic motion planning algorithm.

Berend van den Berg

Master of Science Thesis



Delft University of Technology

Department of Cognitive Robotics

Curvature Aware Motion Planning with Closed-Loop Rapidly-exploring Random Trees

Towards a generic motion planning algorithm.

Master of Science Thesis

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

Berend van den Berg

January 16, 2020

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

Delft University of Technology
Department of
Cognitive Robotics

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

Curvature Aware Motion Planning with Closed-Loop Rapidly-exploring Random Trees

by

Berend van den Berg

in partial fulfillment of the requirements for the degree of

Master of Science Mechanical Engineering

Dated: January 16, 2020

Supervisor(s):

Dr. M. Alirezaei

Dr. J. Alonso-Mora

Reader(s):

Dr. R. Happee

PhD. Candidate B. Brito

Abstract

Last few decades the autonomous driving research field has shown exponential growth. The social benefits, which include increased safety, mobility and productivity, are the main factor that drive this growth. One of the most difficult problems that vehicle engineers must solve to develop autonomous vehicles is the motion planning problem.

They must solve the motion planning problem for environments ranging from unstructured to structured, such as parallel parking up to high-speed highway driving. Current literature presents many implementations that solve either the structured or unstructured planning environment or a small range of environments. Yet, the generic implementation of a single motion planning method, that can plan in the full range of environments, is still an open question.

The aim of this thesis is to address the identified gap in the literature, by realizing a real-time implementation of a single motion planning method, that shows human-like and safe driving behavior and can deal with any environment it encounters.

In this thesis, a method is proposed that solves the planning problem by enhancing the Closed-Loop Rapidly-exploring Random Tree (CL-RRT) algorithm for planning on curved structured roads. The planner is aware of the road curvature and deforms the motion plan so it follows the shape of the road. Extensive simulations have demonstrated that the proposed method can improve the path quality on curved highway roads when compared with the standard RRT and CL-RRT. Although the method can plan in any environment it encounters, it demonstrated limitations in its capability of dealing with complex dynamic environments.

Table of Contents

List of Figures

List of Tables

Preface

Dear reader,

This thesis presents the results of the research that was performed to obtain my Master's degree in Mechanical Engineering at Delft University of Technology. The thesis represents the toughest, but also most challenging and motivating, year of my entire studies. The past year was utterly dedicated to researching motion planning methods for autonomous driving. During this year, many tough obstacles appeared on the road towards success. My passion for the subject made me overcome these obstacles and allowed me to reach the results that are presented in this thesis.

I would like to express my gratitude to my daily supervisor Mohsen Alirezaei for the time that he has invested in guiding the research, and the help he provided in finding solutions to the occurred challenges. I feel that without his years of experience, the research would never have come as far as it is today. I also want to thank my second supervisor, Javier Alonso-Mora, for the excellent feedback he has given throughout the project, which has motivated me to push the research to the limits. Furthermore, I would like to express my appreciation for the help that Bruno Brito provided with setting up the simulations.

Finally, I would like to thank my family who has supported me throughout my seven years of studying.

Thank you, and enjoy the read.

Berend van den Berg
January 16, 2020

Glossary

List of Acronyms

2D	2-Dimensional
A-RRT	Adaptive RRT
BVP	Boundary Value Problem
CL	Closed Loop
CL-RRT	Closed-Loop RRT
CVP	Constraint Violation Probability
KF	Kalman Filter
EG-RRT	Environment Guided RRT
I-RRT*	Informed RRT*
MPC	Model Predictive Control
OBB	Oriented Bounding Box
OL	Open Loop
RC-RRT	Resolution Complete RRT
RG-RRT	Reachability Guided RRT
RRT	Rapidly-exploring Random Tree
RRT*	Asymptocically Optimal RRT
SAT	Separating Axis Theorem

List of Symbols

Greek Symbols

δ	Steer angle of front wheels

Latin Symbols

C_f	Front tire cornering stiffness
C_r	Rear tire cornering stiffness
I_z	Inertia around vertical axis
l_f	Distance from center of mass to front axle
l_r	Distance from center of mass to rear axle
m	Mass
r	Yaw-rate
v	Lateral velocity

Chapter 1

Introduction

1-1 Backgrounds

Last few decades the research on Autonomous Vehicles has been growing rapidly. This becomes clear when inspecting the number of publications on Autonomous Driving, presented in Figure 1-1. The illustrated exponential growth is driven by the benefits that Autonomous Driving can offer society. Autonomous driving has the potential of eliminating all human error, thereby increasing traffic safety. Furthermore, passengers can engage in other activities while driving towards their destination. This can increase their productivity.

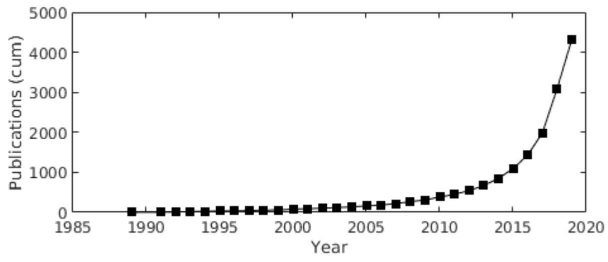


Figure 1-1: Autonomous Driving publications. Obtained from Scopus with search terms: "Autonomous" AND "Driving"

To realize these benefits, the Autonomous Vehicle (AV) must completely take over all driving tasks. These tasks include perceiving the environment, planning motion through the environment and controlling the vehicle to follow the desired trajectory. To achieve a fully Autonomous Vehicle, complicated systems must be designed. These systems must exhibit the desired performance for safe operation in complex dynamic environments. The research in this thesis focuses on the motion planning layer of the AV.

A motion planner suitable for autonomous driving must be able to plan in environments ranging from unstructured to structured. Each of these environments introduces unique challenges for the planner.

1-2 Analysis of the Operating Domain

Structured environment
Structured environments are encountered most often by the AV. These environments include urban and rural roads, with the most extreme example being highways (see Figure 1-2). Highways are designed such that the vehicle's lateral acceleration is constrained under normal operating conditions. Thereby safe driving can be achieved at high speeds. This is accomplished by constraining the road curvature and curvature transitions [1]. These constraints result in roads that have simple geometry. Even though this scenario presents simple geometry, it still introduces several challenges. The scenario is highly dynamic because of the high speeds that are involved. Hence, the planner must be capable of dealing with dynamic obstacles and guaranteeing vehicle stability.

Unstructured environment
Contrary to structured environments, in unstructured environments dynamics are often less important. An example of such an environment is an urban parking lot (see Figure 1-3). Here, the planner mostly deals with static obstacles (being parked cars). The lack of geometric structure makes presents the planner with a new challenge: motion must be planned through an area with cluttered many obstacles and tight passages. Thus, maneuverability becomes important for the planner.

Planner requirements
The environments introduced in the previous sections are the most extreme examples the vehicle can encounter. We can consider anything in between as a combination of the two. For example: performing a parking maneuver to steer the vehicle into a parking spot parallel to the lane. If a planner is constructed that can deal with the two extremes, it is safe to assume it can plan in any environment. This results in the following requirements for the planner:

- Dynamic obstacles must be safely avoided
- Vehicle stability must be guaranteed up to high speeds
- Reverse driving must be possible for performing complex maneuvers
- Computational time must be short enough for real-time implementation

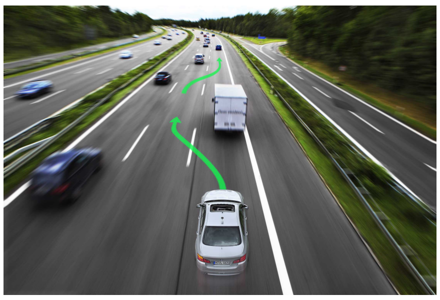


Figure 1-2: Structured highway scenario[1]

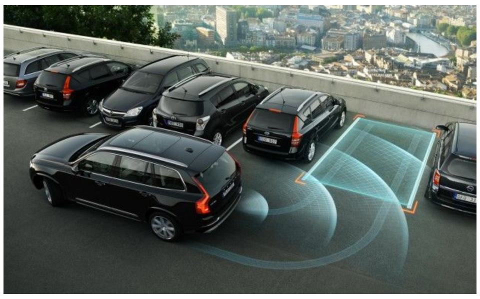


Figure 1-3: Unstructured parking scenario[2]

[1]Reprinted from BMW Group, Retrieved December 10, 2019, from https://mediapool.bmwgroup.com/cache/P9/201108/P90081822/P90081822-research-project-highly-automated-driving-on-highways-08-2011-2250px.jpg

[2]Reprinted from Planning the path for a Self-Driving Car on a Highway, In TowardsData-

1-3 Objectives

The research performed in this thesis has the aim to:

> *'Realize a real-time implementation of a single motion planning method, applied to Autonomous Driving, that demonstrates human-like and safe driving, and is flexible enough to deal with any environment it encounters.'*

Since the start of the research, the focus has been on realizing this with a Rapidly-exploring Random Tree (RRT) based method. Therefore, we composed the following research questions to guide the research:

1. Can an RRT based generic motion planning algorithm be developed, that is capable of planning motion in real-time, for environments ranging from unstructured to structured while considering stability and non-holonomic constraints of the vehicle?
    (a) Which flaws do current planners present?
    (b) How can the performance of these planners be enhanced?
    (c) How can the generality of the proposed method be proved?
    (d) How does the performance of the proposed method compare to others?

A comprehensive literature survey was done before starting the research. This survey is presented in Chapter 3. The survey revealed that Closed-Loop RRT is most suitable for our objective. Additional research questions were defined that steered the research into this direction:

2. How can the Closed-Loop RRT be enhanced to satisfy the requirements?
    (a) How can structured road planning be improved?
    (b) How can dynamic obstacles be considered during planning?
    (c) Under which conditions can vehicle stability be guaranteed?

The research that was performed had the objective to prove our hypothesis:

> *'The Closed-Loop Rapidly-exploring Random Tree can be enhanced such that it becomes capable of planning motion for highway driving also. Thereby, we can realize a generic implementation of the algorithm, which can plan motion for environments ranging from unstructured to structured.'*

Science, Retrieved December 10, 2019, from https://hips.hearstapps.com/hmg-prod/amv-prod-cad-assets/wp-content/uploads/2015/02/2016-Volvo-XC90-T6-223-626x382.jpg

1-4 Thesis content and organization

Chapter 2 introduces several basic motion planning definitions that are required for obtaining an understanding of the motion planning problem. Readers familiar with motion planning can skip this chapter missing no important information. Those who are not, are recommended to read this chapter in preparation to the rest of the thesis.

Chapter 3 analyzes works related to RRT motion planning. It describes how the RRT algorithm works, and which variants exist. Reading this chapter can enhance the readability of the article for those who are not familiar with RRT motion planning.

Part 1 presents the main results in the form of a conference article for the IEEE Intelligent Vehicles Symposium. In this article, the state-of-the-art and preliminaries for the proposed method are discussed. Then, it is discussed how the proposed method builds on the preceding method. The proposed method is evaluated with a variety of structured and unstructured simulation scenarios.

Part 2 provides material that gives further insight into the topics discussed in the article. Chapter 4 present the stability analysis which forms the foundation for the stability criteria that are considered during motion planning. In Chapter 5 specifics are discussed on the implementation of the lateral controller presented in the article. Chapter 6 presents simulation results that are complementary to the results presented in the scientific article. Lastly, Chapter 7 discusses some necessities for reproducing the results of the statistical analysis presented in the article.

Part 3 consists of several self-contained appendices with supplementary material. These appendices are mainly interesting to those interested in reproducing the motion planner implementation.

Chapter 2

Motion planning definitions

Definition 1 (Workspace). The workspace is the environment where the vehicle and obstacles live in. Autonomous vehicles navigate on a 3-dimensional surface that can be locally approximated as a flat 2-dimensional plane. Therefore, the workspace can be described as $W = \mathbb{R}^2$. Obstacles within this workspace are defined as the *obstacle region* $O \subset W$ and the region that the vehicle (Agent) occupies as $A \subset W$.

Definition 2 (Configuration space). The configuration space C is the set of all possible vehicle configurations. For planar vehicles, the 2-Dimensional (2D) pose of the vehicle (x, y, θ) is often considered as a configuration. Again an obstacle region C_{obs} is defined and an obstacle free region $C_{free} = C \setminus C_{obs}$ [3].

Definition 3 (State space). The state-space must be used when problems are time-varying. The state space is defined as the configuration space, extended with the time element $X = C \times T$. The obstacle space X_{free} and obstacle free space $X_{free} = X \setminus X_{obs}$.

The goal of motion planning is to plan a path from an initial configuration to a goal configuration. This path must only traverse the obstacle-free space. When planning motion for

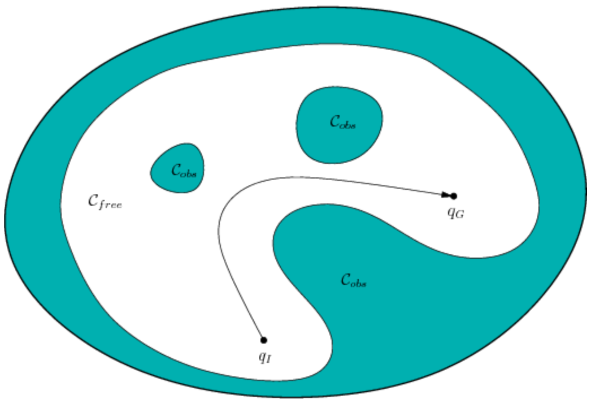


Figure 2-1: Basic motion planning problem [2]

vehicles, the problem is subject to a set of differential constraints D. These constraints often arise from rolling contact or momentum conservation laws [4]. When only first-order constraints are present, the problem is referred to as nonholonomic motion planning. When both constraints of first- and second-order are present, the problem is referred to as kinodynamic motion planning. Often a cost function is used for finding the optimal path. The formal path planning definition is introduced in Problem 1. For time-varying problems, the trajectory planning problem is introduced in Problem 2.

Problem 1 (Path planning).

$$\begin{array}{lll} \arg\min & J(\sigma) & \text{subject to} \\ & \sigma(0) = q_{init} & \\ & \sigma(1) = q_{goal} & \\ & \sigma(\alpha) = C_{free} & \forall \alpha \in [0,1] \\ & D(\sigma(\alpha), \sigma'(\alpha), \sigma''(\alpha), ...) & \forall \alpha \in [0,1] \end{array}$$

Problem 2 (Trajectory planning).

$$\begin{array}{lll} \arg\min & J(\pi) & \text{subject to} \\ & \pi(0) = x_{init} & \\ & \pi(T) = x_{goal} & \\ & \pi(t) = X_{free} & \forall t \in [0,T] \\ & D(\pi(t), \pi'(t), \pi''(t), ...) & \forall t \in [0,T] \end{array}$$

Definition 4 (Complexity). The complexity of a Motion Planning algorithm can be described by an upper and lower bound. The upper bound can be established by evaluating the run-time of an implementation. The lower bound is the minimum theoretical complexity that a class of algorithms can have. This gives a good prior estimate of the difficulty of the problem to be solved. Several classes of complexity exist in literature [2]:

- **P**: can be solved in polynomial time
- **NP**: can be solved in polynomial time by a nondeterministic Turing machine
- **PSPACE**: no more than polynomial amount of storage is used
- **EXPTIME**: can be solved in $O(2^{n^k})$ time, for some integer k

A lower bound was established for a 2D path planning problem with curvature constraint in [5]. In this article the problem was proven to be NP-hard. Hence, no exact solution exists for the problem that remains to be solved during this thesis.

Definition 5 (Properties of Path Planning Methods).

- Completeness: A motion planning algorithm is complete if for any input it correctly reports whether a solution exists in a finite amount of time. This solution must also be returned within a finite amount of time [2].
- Optimality: Return a feasible path that optimizes performance in finite time.
- Anytime: Can be terminated at any time, but solution quality improves with computation time.
- Asymptotic Optimality: The algorithm returns a sequence of solutions that converge to the optimal solution.

Chapter 3

Related works

The objective of this chapter is to provide a comprehensive review of the state-of-the-art related to RRT motion planning. The survey is limited to reviewing the literature on non-holonomic and kinodynamic motion planning only.

First the basic theory behind the RRT is discussed, explaining step by step how the algorithm works. This will provide a key understanding of how the method works, which is required for analyzing all its basic components in section 3-2. This understanding is required when advancing to more complex RRT variants, which are discussed in section 3-3. Last, a concise comparison will be provided. This comparison includes the complexity, optimality and real-time applicability.

3-1 Motion planning with Rapidly-exploring Random Trees

The Rapidly-exploring Random Tree (RRT) was introduced by LaValle as an algorithm that is particularly suitable for planning motion for kinodynamic systems [3]. It uses an efficient data structure and sampling scheme to quickly search high-dimensional spaces that have both algebraic constraints (arising from obstacles) and differential constraints (arising from nonholonomy and dynamics). Due to the nature of the RRT algorithm, exploration is biased to unexplored areas [6]. The algorithm avoids explicitly constructing the obstacle region and instead conducts a search that probes the configuration space with a sampling scheme. This probing is enabled by a collision detection module, which the motion planning algorithm considers as a 'black box' (see Figure 3-1).

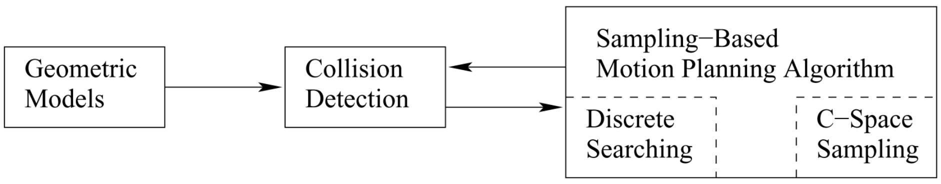


Figure 3-1: Sampling-based planning structure with "black box" collision detection [2]

To keep things simple during the introduction, the algorithm will be introduced for planning in the configuration space. Extending this to state-space planning is straightforward. The RRT algorithm builds a tree-like structure (see Figure 3-2) consisting of nodes and edges. A node contains a vehicle state and a parent node from which it was reached. The edge stores the control input that is used to connect two nodes. The *build rrt* operation is shown in Algorithm 1. Here, the tree is initialized with the initial configuration of the vehicle (line 1). Then, repeatedly a random configuration is sampled (line 2-3). This configuration and the tree itself are used as input for the *extend* operation, shown in Algorithm 2.

Algorithm 1: BUILD_RRT(x_{init})

```
1 $\mathcal{T}.init(x_{init})$
2 for $k = 1$ to $K$ do
3     $x_{rand} \longleftarrow$ RANDOM_STATE()
4     EXTEND($\mathcal{T}, x_{rand}$)
5 return $\mathcal{T}$;
```

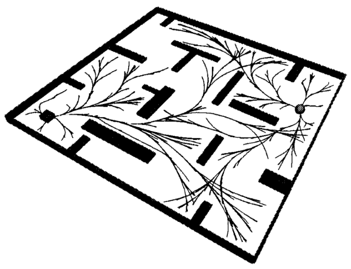


Figure 3-2: Kinodynamic RRT tree [6]

As the name indicates, the extend operation is used for extending the tree towards the sampled configuration (see Figure 3-3). First, the nearest neighboring node of the sampled configuration is determined using a distance measure (line 1). This node is then selected for expansion (line 2). A new configuration is generated through a forward simulation towards the randomly sampled configuration. The input of the simulation can be either selected randomly or be selected as most promising from a discrete set of inputs. The simulated trajectory is checked against collision constraints and if it is collision-free, the node is added to the tree (line 3) and an edge in line (4).

Algorithm 2: EXTEND($\mathcal{T}, x$)

```
1 $x_{near} \longleftarrow$ NEAREST_NEIGHBOR($x, \mathcal{T}$)
2 if $NEW\_STATE(x, x_{near}, x_{new}, u_{new})$ then
3     $\mathcal{T}$.add_vertex($x_{new}$)
4     $\mathcal{T}$.add_edge($x_{near}, x_{new}, u_{new}$)
5     if $x_{new} = x$ then
6         return Reached
7     else
8         return Advanced
9 return Trapped
```

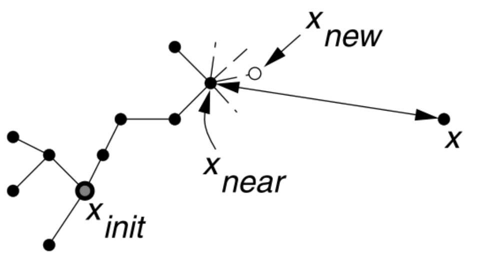


Figure 3-3: Tree extend operation

Ever since the introduction of the algorithm, it has been a popular research topic. This has led to many variants of the algorithm which enhance performance for specific scenarios. Although the algorithms can behave very different, they all consist of certain basic components. These components will be introduced in the next subsection.

3-2 Analysis of the algorithm components

All variants of the algorithm are build of the following components:

- Sampling Function: sample a point in the configuration space
- Metric: a measure to determine the distance between two configurations
- Steer Function: connect samples while considering kinematic or dynamic constraints
- Collision detection: check whether a generated edge is collision-free

Because these components have major impact on the behavior and performance of the algorithm, they will be introduced in the following subsections.

3-2-1 Sampling Function

The ideal algorithm would consider all possible configurations to determine the optimal path. Due to the complexity of the problem, and the limited time available for finding a solution, it is often not realistic to search the entire configuration space. This problem can be solved by sacrificing completeness for computational feasibility [7]. With sparse configuration sampling, the algorithm can perform a faster search. The goal of the sampling function is to sample these configurations.

When the algorithm runs for infinite time, the entire configuration space would be sampled, giving a complete representation. Yet, this is often not feasible due to timing constraints. Thus, the algorithm is terminated early. At the moment of termination, the sampled configurations should be properly distributed over the configuration space.

The standard RRT algorithm has a uniform sampling strategy. This gives it a strong bias towards unexplored areas. In structured environments, such a sampling strategy may not be optimal. Various strategies were developed for enhanced sampling in structured environments. Some examples are sampling around the medial axis, sampling around obstacle boundaries, Gaussian-shaped sampling, the bridge-test for sampling in narrow passages and goal biased sampling [8]. A numerical comparison is presented in [7]. Performance on structured roads can be enhanced by sampling about the road center-line [9]. Once a sample is generated, a node must be selected for expansion using the metric.

3-2-2 Metric

The metric is the measure that is used for calculating the distance between two points in the configuration space. It is used for finding the closest neighboring node to select for expansion. For holonomic systems, the metric is often defined as the Euclidean distance between two points. Nonholonomic systems need a more complicated metric. This is because some maneuvers can be required before two points can be connected. The length of the shortest path possible path (geodesic) is often used as a metric for nonholonomic systems [2]. An algebraic solution for a kinematic vehicle model was developed in [10]. This algorithm returns the shortest possible path between two configurations. It considers a simple car that can only drive forward and has a limited turning radius.

3-2-3 Steer Function

After a node is selected, it must be expanded towards the sampled configuration. The steer function is responsible for this. Finding a path that connects both configurations exactly, requires solving a two-point Boundary Value Problem (BVP). For nonholonomic and dynamical systems, it is often nontrivial to find a closed-form solution to the BVP. This makes the problem a rather time-consuming process.

Solving a BVP for every sample can be avoided through the use of an approximate steer function. These methods often use a simulator to generate new configurations [2, 3, 4, 6, 11]. The simulator performs a forward simulation of the robot dynamics by applying control inputs to the model. For a car, these inputs can be the steering angle and forward velocity. One option is to select the inputs randomly. Another possibility is to simulate all possible inputs and select the path that comes closest to the sample. This method was implemented for a car in [12].

Many methods exist for solving the BVP. For holonomic systems the solution is straightforward. Just connect the configurations with a straight line. Nonholonomic systems require more complicated methods due to the differential constraints. The nonholonomic BVP was solved exactly for a kinematic vehicle model with a fixed cornering radius [10]. The solution considers forward driving only. Later, it was extended with backward driving functionality [13] (see Figure 3-4 b-d). Although these methods solve the problem, they produce paths that contain curvature discontinuities. Therefore, a nonholonomic system may struggle to track the planned path. To address this shortcoming, alternatives were developed that enhance the paths with continuous curvature transitions [14, 15]. All the before mentioned methods use a fixed cornering radius for constructing paths. This makes them feasible for vehicles

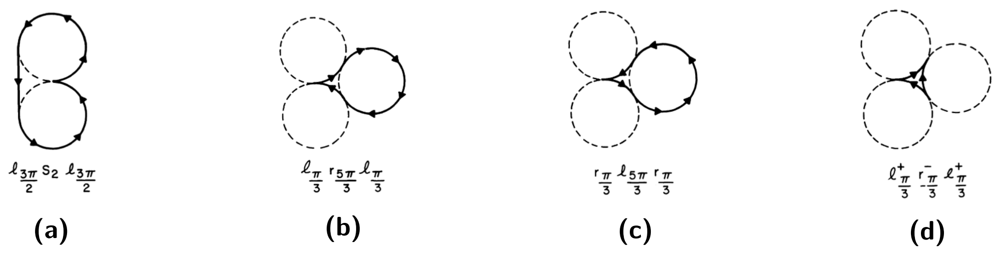

$\ell_{\frac{3\pi}{2}} s_2 \ell_{\frac{3\pi}{2}}$
(a)
$\ell_{\frac{\pi}{3}} r_{\frac{5\pi}{3}} \ell_{\frac{\pi}{3}}$
(b)
$r_{\frac{\pi}{3}} \ell_{\frac{5\pi}{3}} r_{\frac{\pi}{3}}$
(c)
$\ell^+_{\frac{\pi}{3}} r^-_{\frac{\pi}{3}} \ell^+_{\frac{\pi}{3}}$
(d)

Figure 3-4: Dubins and Reeds-Shepp's shortest paths: A,B,C are Dubins, D is Reeds-Shepp [13]

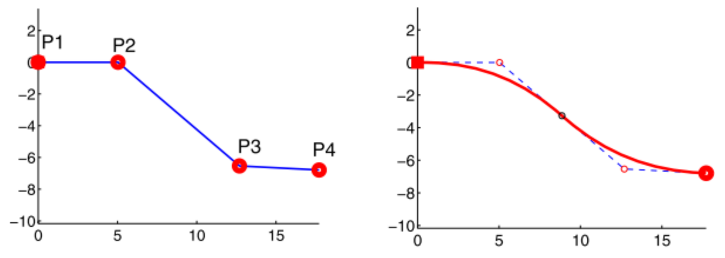


Figure 3-5: Spline fitting to a piece-wise linear path. Adapted from [19].

moving at slow speeds, where vehicle dynamics can be neglected. For fast-moving vehicles, the dynamics become relevant and alternative methods are required.

A method that is not constrained to paths with fixed curvature, is to fit splines between the configurations (see Figure 3-5). By adding constraints on the spline connections, a continuous curvature path can be guaranteed. An upper bound can be defined for the curvature of the path [16].

Lastly, the BVP can be solved through the shooting method [17, 18]. This method iteratively solves the equations of motion for a given BVP. Compared to exact solutions, this method can require more computational effort to find the solution.

3-2-4 Collision Detection

Once a path segment is generated by the steer function, the next task is to check whether it will result in a collision. The choice of the collision detection algorithm has major impact on the computational performance of motion planning methods [2, 20, 21, 22]. Therefore, it is important to use an efficient method. Besides checking whether a collision occurs or not, occasionally the distance to the closest obstacle is logged for path quality assessment. A common approach is to use multiple phases in the detection algorithm [2].

During the *broad phase*, the goal is to avoid exhaustive collision checking. The computational effort is reduced for complex collision checks by simplifying the problem. This can be done for obstacles that are so far apart that the exact geometry is not relevant. During this phase, simple geometry shapes are used for modeling the obstacles. Some of the options are fitting a circle, using an axis-aligned bounding box, an oriented bounding box or a convex polygon (see Figure 3-6).

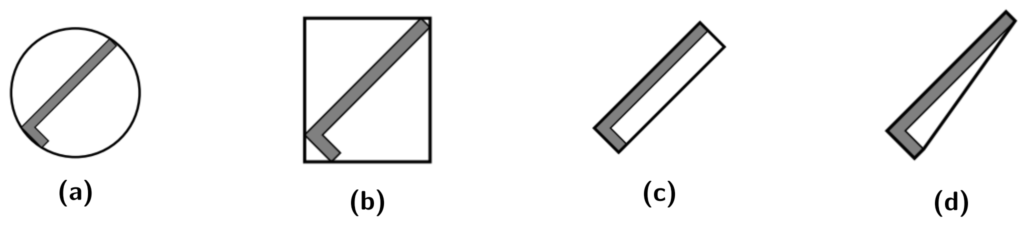


Figure 3-6: Modelling obstacles with a bounding geometric shape [2]. (a) circle, (b) axis aligned bounding box, (c) object oriented bounding box, (d) convex polygon.

The *narrow phase* is only entered when the obstacles are so close together that a more complicated and accurate obstacle representation is required. Two main categories exist which are *hierarchical* methods and *incremental* methods. More information on these methods can be found in [2].

3-3 Exploring variants of the algorithm

In an attempt to enhance the properties of the RRT, many variants of were developed that make adjustments in its components. The main focus is on reducing the algorithm run-time and the cost of the solution.

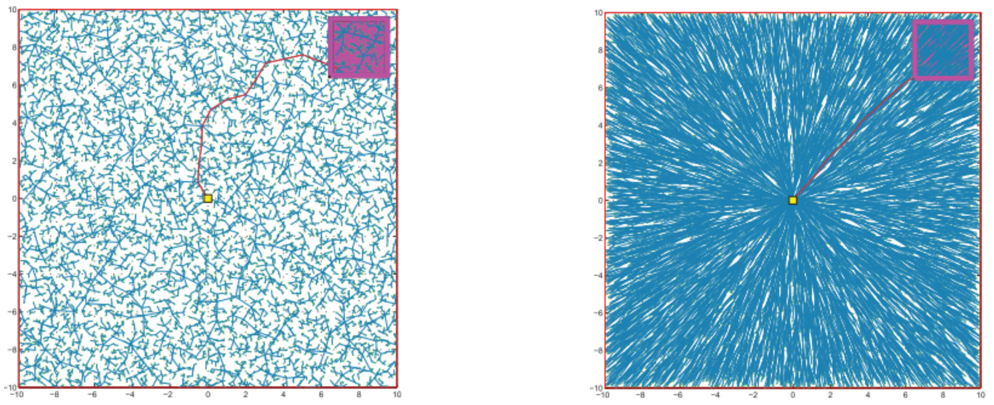
3-3-1 Reducing the Metric sensitivity

The node that is nearest to the sample is expanded and checked against constraints. These are computational expensive operations. An improper metric could lead to more failing expansions. Thus, performance is influenced much by the choice of the metric. The Resolution Complete RRT (RC-RRT) [23, 24] was developed to reduce the metric sensitivity. It does this by logging exploration, collisions, and applied inputs for each node. It uses this information during the nearest node selection as follows. If a control input has already been applied, it will not be considered again. If a control input leads to a collision, the node is penalized in the Constraint Violation Probability (CVP). When the CVP of a node increases, the chance that it will be selected for expansion decreases.

The Reachability Guided RRT (RG-RRT) [25] also reduces the metric sensitivity. It does this by accounting for the limitations of the system's dynamics, by defining a reachable set for each node. A node can only be selected for expansion when the generated sample lies within its reachable set. This method no longer requires a system-specific metric. Instead, the Euclidean distance can be used.

3-3-2 Improving the performance in Heterogeneous Environments

The Environment Guided RRT (EG-RRT) [26] is a combination of the RC-RRT and RG-RRT. The results of this algorithm showed that it had a better exploration strength. It produces paths faster while having a lower probability of collision.



(a) RRT **(b)** RRT*

Figure 3-7: Final path of RRT compared with RRT* [27]

3-3-3 Introducing optimality

The authors of [27] prove that the RRT almost always converges to a sub-optimal solution. This sub-optimality may express itself in the planning of a meandering path. The Asymptocically Optimal RRT (RRT*) addresses this issue by ensuring asymptotic optimality of the solution [27, 28]. This method works as follows. During exploration, a cost function is used for logging path quality. The sample is then connected to the node with the lowest cost. It then rewires all nodes in an area around the selected node to the produce the cheapest paths. For a comparison of the RRT and RRT* algorithms, see Figure 3-7. The method was applied to nonholonomic [18] and kinodynamic systems [29]. Although asymptotic optimality is introduced, the re-wiring operation does add extra complexity to the algorithm, which may restrict real-time applications to using exact steer functions only.

3-3-4 Enhancing the RRT* convergence rate

The Informed RRT* (I-RRT*) [30] aims to enhance the convergence rate of the RRT*. Before a solution is found, this algorithm behaves the same as the RRT*. After a solution is found, a hyper-spheroid is drawn around the initial and goal configuration. Samples are from there on only generated within this ellipse. The size of the ellipsoid is reduced as the solution converges to the optimum (see Figure 3-8). Another variant that aims to increase the convergence rate is the RRT$^{\#}$. It applies relaxation methods to quickly identify the region that contains the optimal solution, and uses to bias the exploration.

3-3-5 Adaptive sampling around obstacles

The Adaptive RRT (A-RRT) [31] improves performance in heterogeneous environments. It does this by adapting growth through a selection method consisting of two levels. The first level selects groups of expansion methods according to the visibility of the node being expanded. The second level uses a cost-sensitive learning approach to select a sampler from a group of expansion methods.

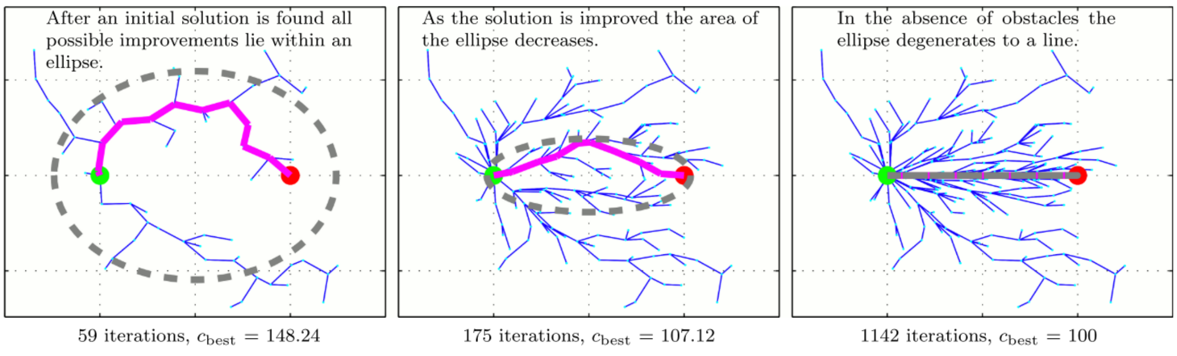


Figure 3-8: The Informed-RRT* algorithm [30]

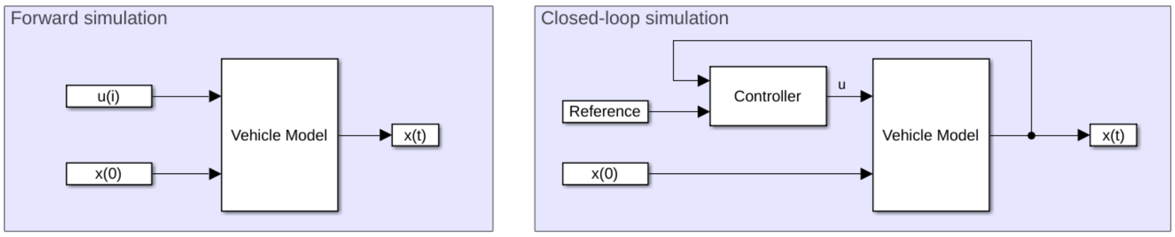


Figure 3-9: Forward simulation versus closed-loop simulation

3-3-6 Closed-Loop RRT

The Closed-Loop RRT (CL-RRT) [9, 32, 33] has a design philosophy that varies from the standard RRT. Where the RRT plans with Open Loop (OL) vehicle dynamics, the CL-RRT does this with Closed Loop (CL) dynamics. The OL dynamics simulate vehicle inputs, e.g. steer angle and longitudinal velocity. The CL dynamics simulate the controller inputs, e.g. a reference path and velocity (see Figure 3-9).

The CL-RRT has several advantages when compared to the OL RRT. First of all, stability can be guaranteed for complex nonlinear system, by using a stabilizing controller. Secondly, the effect of modeling errors can be decreased by applying appropriate controllers.

The algorithm builds two trees. The first tree is built by sampling the vehicle workspace and generating a reference. This reference consists of a path and velocity profile, consisting of a series of data points $(\mathbf{x}_{ref}, \mathbf{y}_{ref}, \mathbf{v}_{cmd})$ with a single driving direction. The reference path is used for generating a second tree. This tree consists of trajectories that are generated through closed-loop prediction (see Figure 3-9). Contrary to the RRT, the CL-RRT has no constraint on path length. This can result in longer path segments. Recent studies have shown that robust feasibility, as well as a bounded error, can be guaranteed for the CL-RRT.

To improve path quality over time, the algorithm is extended with optimization heuristics [34]. Although this does not offer the same optimality guarantees as the RRT*, it does show better optimality than the standard RRT.

Where the standard RRT requires many samples to reach the goal, the CL-RRT can reach the goal with only a few. This is because the segment length is not constrained, and the algorithm has a strong goal biased.

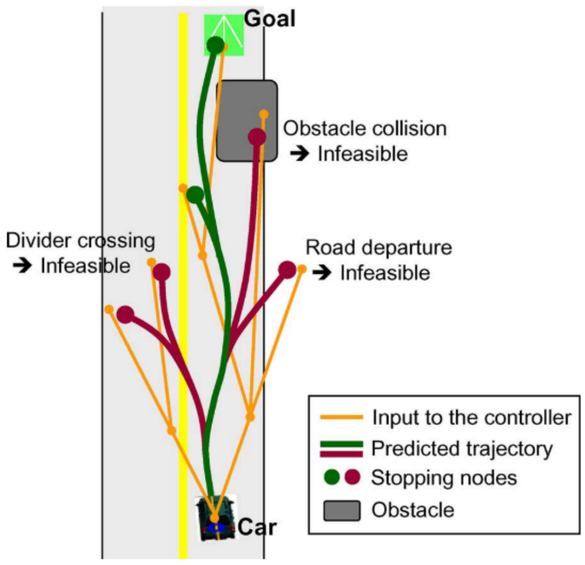


Figure 3-10: Planning with Closed-Loop RRT: Building two trees. The first tree contains piece-wise linear references and the second tree closed-loop trajectories. Green: feasible trajectory. Red: infeasible trajectory. Orange: references. [9]

3-4 Concise overview of discussed variants

A concise literature overview of the algorithm variants that were discussed in the previous section is presented in Table 3-1.

Table 3-1: Consise overview of reviewed RRT variants

Algorithm	Author	Contributions
RRT [6]	S. LaValle	RRT Algorithm
RRT* [28]	S. Karaman, E. Frazzioli	Converge to optimal solution
RRT$^{\#}$ [35]	O. Arslan, P. Tsiotras	Improve RRT* convergence rate
RC-RRT [23]	P. Cheng, S. LaValle	Reduce metric sensitivity
RG-RRT [25]	A. Shkolnik et al.	Node connection based on reachable set
EG-RRT [26]	L. Jaillet et al.	Combine RC-RRT and RG-RRT
I-RRT* [30]	J. Gammell et al.	Improving the convergence rate
A-RRT [31]	J. Denny et al.	Adaptive sampling around obstacles
CL-RRT [9]	Y. Kuwata et al.	Planning over closed-loop dynamics

Part I

Scientific article

Curvature Aware Motion Planning with Closed-Loop Rapidly-exploring Random Trees

Berend van den Berg*, Bruno Brito*, Javier Alonso-Mora* and Mohsen Alirezaei†‡

***Abstract*—The development of a generic motion planning algorithm for Autonomous Driving, that can plan motion in environments ranging from unstructured to structured, is still an open question. This article presents a motion planning framework that can plan motion for environments ranging from unstructured parking to structured highway driving. The presented method is based on the Closed-Loop Rapidly-exploring Random Tree, and is enhanced for motion planning on curved structured roads. Given the road center line, obstacle positions, initial pose and goal pose, the proposed method plans motion on a virtual straight road. Afterwards, the planner deforms the motion plan to follow the road curvature. Extensive simulations have demonstrated that the proposed method can improve the path quality on curved highway roads when compared with the standard RRT and CL-RRT. Although the method can plan in any environment it encounters, it demonstrated limitations in its capability of dealing with complex dynamic environments.**

***Index Terms*—RRT, Autonomous Driving, Curvature, Frénet, Closed-Loop**

I. Introduction

AUTONOMOUS DRIVING requires a motion planner to define a motion plan from its current position to a goal position. A motion planner suitable for Autonomous Driving must be able to plan in environments ranging from unstructured to structured. Unstructured environments are often urban environments such as parking lots. Here, there are low speed limits and vehicle maneuverability is most important for the planner. On the other hand, there are structured environments with simplistic road geometry, such as highway driving. Due to the high speeds involved, the dynamic feasibility of the motion plan becomes crucial for safe driving.

Current literature presents many motion planner implementations that solve either the structured [1], [2], [3] or unstructured planning environment [4], [5]. Some implementations even solve a wider range of environments [6], [7], [8]. Yet, the generic implementation of a single motion planning method, that can plan in the full range of environments, is still an open question. A generic motion planner would eliminate the need for scenario identification that is required with a modular approach. This article addresses the identified gap in the literature with a generic implementation of a Closed-Loop Rapidly-exploring Random Tree (CL-RRT) based algorithm that is enhanced for driving on structured roads.

* The authors are with the department of Cognitive Robotics, Delft University of Technology, 2628 CD, Delft, The Netherlands

† The author is with Siemens Industry Software and Services B.V., Digital Industry Software Simulation and Testing Services, 5708 JZ, Helmond, The Netherlands

‡ The author is with the department of Mechanical Engineering, TU Eindhoven, 5600 MB, Eindhoven, The Netherlands

The paper is organized as follows. First, work related to RRT motion planning is discussed. This is followed by our contributions. Next, the motion planning problem is introduced. Preliminaries for the proposed method are discussed, followed by the method itself. Lastly, the proposed method is tested by means of simulations with static and dynamic obstacles.

A. Related work

The RRT [9] can quickly search high-dimensional spaces that have both algebraic constraints (arising from obstacles) and differential constraints (arising from nonholonomy and dynamics). This makes the algorithm extremely useful for real-time applications. The algorithm incrementally builds a tree of paths by connecting randomly sampled configurations. This connection is made with either exact or approximate steer functions, which are usually based on vehicle kinematics or dynamics. The advantage of approximate steer methods is that they avoid the need for solving a Boundary Value Problem (BVP) for each sample, which can be a computationally expensive operation for differentially constrained systems [10]. Therefore, planning motion real-time with a probabilistic sampling-based planner with a nonholonomic or kinodynamic vehicle model is often restricted to the use of approximate steer functions only.

Although the RRT has many benefits with respect to traditional combinatorial path planning methods, a shortcoming can still be identified: the algorithm often converges to a sub-optimal solution [11]. This sub-optimality may express itself in the path quality by e.g. producing a meandering path.

Early variants of the algorithm enhance the solution quality by adding an optimization heuristic during nearest neighbor selection [7], [12], [13], or prune the path after a solution is found [14]. However, this does not guarantee asymptotic optimality. Therefore, the RRT* introduces a rewiring operation that can guarantee asymptotic convergence towards the optimal solution [11], [15], [16]. Due to the number of BVP that must be solved during rewiring, the real-time implementation may be restricted to the use of exact steer functions only (e.g. [17]).

An approach that avoids the need of solving BVP, is to use the Closed-Loop RRT (CL-RRT) [7], [18]. This variant samples in the controller's input space (e.g. a path and reference velocity) instead of the vehicle inputs (e.g. steer angle and velocity). The controller inputs are used for simulating a closed-loop trajectory of the vehicle controlled by lateral- and longitudinal controllers.

Traditional RRT algorithms sample directly the vehicle inputs and have a steer function with short prediction horizon. Usually far less than a second. Therefore, these methods require many samples (and thus path segments) to reach the goal. In contrast, the CL-RRT can generate path segments of several seconds. Therefore, this variant requires far less samples to reach the goal, and may suffer less from path meandering issues than traditional variants. Therefore, the proposed method is based on the CL-RRT.

Due to the piece-wise linear nature of the controller inputs, the CL-RRT is not capable of accurately following the curved roads. This issue remains to be solved before it can be applied to high speed driving on structured roads.

B. Contributions

In this article, we present the following contributions:

- A generic implementation of a RRT [9] based motion planning algorithm that can plan in structured and unstructured environments
- Combine the Closed-Loop RRT [18] variant with a path bending method that improves path quality in structured environments

II. Problem formulation

Several assumptions are used during the problem definition:

- Obstacle positions are known
- Vehicle states are known
- Goal positions are provided by a mission planner
- Lane marking coefficients are provided by a lane detection system

A. Vehicle representation

The Autonomous Vehicle (AV) operates on a plane. Hence, its workspace is $W \in \mathbb{R}^2$. The dynamics of the AV are represented by a set of nonlinear dynamic equations:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \quad \mathbf{x}(0) = x_0 \tag{1}$$

where $\mathbf{x}(t) \in \mathbb{R}^{n_x}$ and $\mathbf{u}(t) \in \mathbb{R}^{n_u}$ are the vehicle states and inputs respectively and $\mathbf{x}(0)$ is the initial state of the vehicle. The model used in the presented implementation consists of seven states ($n_x = 7$), which are further introduced in section III-A3. The body of the AV is modeled with a 2-Dimensional Oriented Bounding Box (2D-OBB) that is aligned with the vehicle heading. The region of the state-space that the body of the AV occupies is denoted as $\mathcal{A}(\mathbf{x}) \in X$.

B. Obstacle constraints

Static and dynamic obstacles are considered. Both are modeled as 2D-OBB, of which the heading is aligned with the road (or the vehicle when no road is detected). The region in the state-space that violates the collision constraints is denoted with X_{obs}. Future positions of dynamic obstacles are predicted by linear extrapolation of their velocity.

C. Dynamic constraints

Several additional constraints arise from vehicle stability criteria, dynamic limitations and actuator saturation. The set of states that violate these constraints are denoted with X_{dyn}.

D. Feasible trajectory

Let $\mathcal{T} \in \mathbb{R}^{N \times n_x}$ be a trajectory obtained by advancing the vehicle model (1) N steps. A feasible trajectory satisfies obstacle constraints, dynamic constraints and its inputs lay within the set of admissible inputs:

$$\begin{aligned} &\mathcal{A}(\mathbf{x}) \in X_{free} = X \setminus (X_{obs} \cup X_{dyn}) \\ &\mathbf{u}(t) \in U \qquad \forall t \in [0, T] \end{aligned} \tag{2}$$

E. Motion planning problem

The objective is to find a feasible trajectory from an initial state to the goal region of the state-space X_{goal}, while minimizing the cost:

$$\begin{aligned} \arg\min \quad & J(x(t)) \qquad\qquad \text{subject to} \\ & (1), (2) \\ & x(T) \in X_{goal} \end{aligned}$$

III. Preliminaries

A. Closed-loop RRT

The CL-RRT incrementally builds a tree of feasible trajectories towards the goal. During each tree expansion step (see algorithm 1), first a random sample is generated in the workspace (line 1). Tree nodes are then sorted using heuristics (line 2). Multiple nearest nodes are selected (line 3), which in turn, are expanded until a feasible trajectory is generated (line 5-9). If a new node is added, a goal biased expansion is attempted (lines 10-12). If this expansion is feasible (line 13), it is added to the tree (line 14).

Algorithm 1: ExpandTree($\mathcal{T}$) [18]

```
1  Get random sample $S \in \mathbb{R}^2$.
2  Sort nodes $n$ using heuristics.
3  Select $N_{near}$ nearest neighbors.
4  for all $n \in N_{near}$ do
5      Form a reference command from $n$ to $s$
6      Obtain a trajectory $x(t) \in [t_1, t_2]$ by doing a
         simulation until the end of reference is reached.
7      if $x(t)$ is feasible $\forall t \in [t_1, t_2]$ then
8          $\mathcal{T}$.add_node
9          break
10 if a node was added then
11     Form a reference command from $S$ to $goal$.
12     Obtain a trajectory $x(t) \in [t_2, t_3]$ by doing a
         simulation until the end of the goal reference is
         reached.
13     if $x(t)$ is feasible $\forall t \in [t_2, t_3]$ then
14         $\mathcal{T}$.add_node
15 return
```

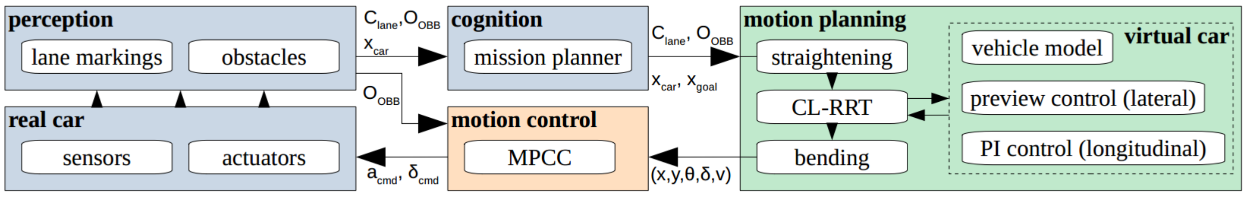


Fig. 1: Schematic overview of the proposed method: C_{lane}: lane marking coefficients, O_{OBB}: obstacle Oriented Bounding Box, x_{car}: vehicle states, x_{goal}: goal states, a_{cmd}: acceleration command, δ_{cmd}: steer angle command.

1) Node sorting: Two heuristics are used for sorting the nodes: exploration and optimization. The exploration heuristic (3) sorts in ascending order based on the distance to the sample. The optimization heuristic (4) sorts based on total time to reach the sample. One of these heuristics is selected probabilistically.

$$J_{exp} = L_{Dubins}(P, S) \tag{3}$$

$$J_{opt} = J_{opt,parent} + \frac{L_{Dubins}(P, S)}{v} \tag{4}$$

Here, $S \in \mathbb{R}^2$ is the sample, $P \in \mathbb{R}^2\mathbb{S}$ is the parent node pose, and L_{Dubins} is Dubins path length from the parent node to the sample, and v is the sampled speed. Equations 3 and 4 are only used for estimating costs for connecting the nodes. The actual cost is updated after doing a closed-loop prediction of the vehicle trajectory.

2) Closed-loop prediction: During closed-loop prediction, a virtual car is driven along a generated reference. This reference consists of a path and velocity profile, and is further discussed in section IV-A2. The prediction produces trajectory, which is added to the tree if it satisfies (2). The prediction requires a vehicle model, which is introduced in the next section, and lateral- and longitudinal controllers, which are introduced in section IV-A5.

3) Vehicle model: To constrain the algorithm complexity, a kinematic vehicle model is applied. It has been previously demonstrated that this model can be consistent for planning feasible trajectories as long as its lateral acceleration remains small [19]. The vehicle model (5-7), is extended with understeer behavior (7), steer dynamics (8), acceleration dynamics (9, 10), and actuator constraints (11-13).

$$\dot{x} = u\cos(\theta) \tag{5}$$

$$\dot{y} = u\sin(\theta) \tag{6}$$

$$\dot{\theta} = \frac{u\tan(\delta)}{u^2\frac{K_{us}}{g} + L} \tag{7}$$

$$\dot{\delta} = {T_d}^{-1}(\delta_{cmd} - \delta) \tag{8}$$

$$\dot{u} = a \tag{9}$$

$$\dot{a} = {T_a}^{-1}(a_{cmd} - a) \tag{10}$$

$$a_{min} \leq a \leq a_{max} \tag{11}$$

$$||\delta|| \leq \delta_{max} \tag{12}$$

$$||\dot{\delta}|| \leq \dot{\delta}_{max} \tag{13}$$

Here, the parameters are defined as follows. (x, y, θ) is the 2-dimensional pose of the vehicle's rear axle center in world coordinates, δ is the steering angle of the front wheels, L is the wheelbase, u and a are the longitudinal velocity and acceleration, K_{us} is the vehicle's understeer coefficient. δ_{cmd} and a_{cmd} are the control commands determined by the lateral and longitudinal controllers. Lastly, T_a and T_d represent the first order lag of the steering system and longitudinal dynamics.

B. Vehicle stability

The vehicle dynamics can be accurately modeled using the linear dynamic bicycle model, when the following constraints satisfied:

- the vehicle operates in the linear tire region
- steer angles remain small and vary slowly
- longitudinal velocity varies slowly

Following stability analysis, it can be concluded that the vehicle remains stable when it shows under-steering behavior ($K_{us} < 0$). By constraining the lateral acceleration, the vehicle operates in the linear tire region and the dynamic bicycle model is valid. Hence, a trajectory is dynamically feasible when $a_y \leq a_{max}$, where $a_{max} = 0.3g$.

IV. Method

A schematic overview of the proposed method is presented in Figure 1. Sensors equipped on the vehicle detect the obstacles and the lane markings. The mission planner uses this data to define a local goal and sends this goal to the motion planner. The motion controller makes the vehicle follow the motion plan through Model Predictive Contouring Control [20]. The motion planning method consists of three steps:

1) Transform the planner inputs from the curved road to a virtual straightened road.
2) Build a tree towards the goal using the CL-RRT.
3) Bend the motion plan back to the curved road.

First, several adjustments to the CL-RRT are discussed (Section IV-A). After this, the specifics on the proposed road transformations are discussed (Section IV-B). Last, the applied real-time planning method is discussed in Section IV-C.

A. CL-RRT modifications

1) Cost function: A cost function is introduced for path selection. This function includes path length, path curvature, and deviation from lane center:

$$J_{sel} = \sum_{n=1}^{n=N}\sum_{i=1}^{i=I}(w_1 u\Delta t + w_2\kappa_{n,i} + w_3 D_{n,i}) \tag{14}$$

In this function the parameters are defined as follows. n are the path segments, i are the iterations of the closed-loop prediction of segment n, $w_k, k = 1, 2, 3$ are the cost weights, u is the longitudinal velocity, Δt is the prediction step size, $\kappa_{n,i}$ is the curvature of the path, and $D_{n,i}$ is the distance to the goal lane.

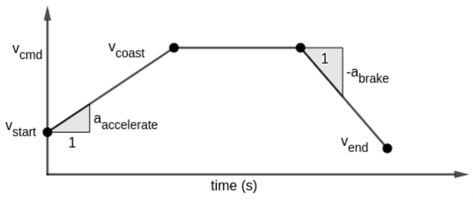


Fig. 2: Velocity profile design

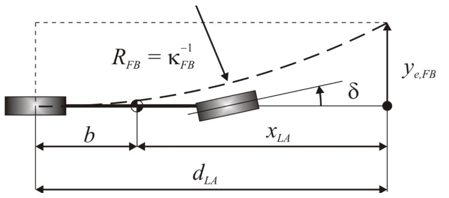


Fig. 3: Generic lateral control concept [21]

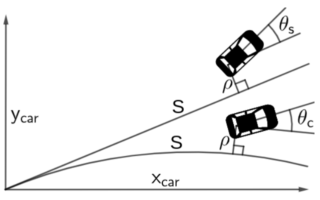


Fig. 4: Road straightening

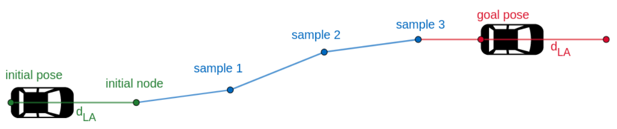


Fig. 5: Reference path generation: from the initial pose, the tree is initialized with a node at distance d_{LA} (green). From here, several regular expansions are performed (blue). Lastly, a goal biased expansion is done extended with look-ahead distance d_{LA}, which is aligned with the goal heading (red)

2) Reference generation: A single reference point consists of a position ($pos \in \mathbb{R}^2$) and a velocity ($v_{cmd} \in \mathbb{R}$). A reference command consists of N points $(pos, v_{cmd})_N$ and a single driving direction.

The first step in reference generation is to construct a path through 2-dimensional space (see Figure 5). A reference path is a linear interpolation between the previous node's reference and the generated sample. A goal biased reference is a reference that is extended with a section to align the vehicle with the goal. The length of this section is coupled to the look-ahead distance of the lateral controller (see section IV-A5). This ensures that the vehicle can always reach the goal before the end of the path is reached.

The second step if to design a velocity profile. For simplicity, a trapezoidal profile is used (see Figure 2). The design procedure of this profile works as follows. The first step is to determine whether the maximum maximum velocity can be reached, while ensuring a minimum coasting time of t_{min}, see (15). Here, the first, second and third terms represent the acceleration, coast and brake distance respectively. D is the total path length. If (15) can be satisfied, the coast velocity (v_{coast}) is set to be the maximum velocity. Else, the coast velocity must be determined in an alternative way. If the end velocity is greater than the start velocity, v_{coast} is set to the end velocity. Else, v_{coast} is chosen such that (16) is satisfied.

$$\frac{v_{max}^2 - v_0^2}{2a_{acc}} + v_{max}t_{min} + \frac{v_{max}^2 - v_{end}^2}{2a_{dec}} < D \quad (15)$$

$$\frac{v_{coast}^2 - v_0^2}{2a_{acc}} + v_{coast}t_{min} + \frac{v_{coast}^2 - v_{end}^2}{2a_{dec}} = D \quad (16)$$

Although the reference command is similar to the one used by [22], the difference is still a small difference. Instead of always stopping at the end of a reference, the proposed profile allows adjustment of the end velocity.

3) Goal region: The goal provided to the planner consists of a 2-dimensional pose and a velocity (x, y, θ, v). These goals would be generated by a mission planner that can scale the goal velocity based on the current road scenario. The goal region is defined in (17), where R_G, H_G, V_G are the goal radius, heading error and velocity error, respectively.

$$\left\| \begin{bmatrix} x - x_G \\ y - y_G \end{bmatrix} \right\| \leq R_G, \quad \theta - \theta_G \leq H_G, \quad v - v_G \leq V_G \quad (17)$$

4) Sampling: In unstructured environments, the planner samples around the vehicle and the goal position (x, y). Whenever road information is available, sampling is done with respect to the road center-line (S, ρ, see Figure 4).

5) Controllers: Lateral control of the virtual vehicle is done with the single point preview controller [21]. The lateral error at the preview point (see Figure 3) is used to calculate the steer command (18). The location of the preview point is velocity dependent (19).

$$\delta_{cmd} = 2\left(\frac{l + K_{us}u^2}{d_{LA}^2}\right) y_{e,FB} \quad (18)$$

$$d_{LA} = b + x_{LA} = b + ut_{LA} \quad (19)$$

The parameters are defined as follows. l is the wheel base, K_{us} is the under-steer gradient, u is the longitudinal velocity, b is the distance from the rear axle to the center of gravity, and t_{LA} the look-ahead time. The look-ahead time varies to enhance maneuverability during parking and curvature smoothness during highway driving.

Longitudinal control is done with the same PI controller (20) as in [18].

$$a_{cmd} = K_p(v_{cmd} - v) + K_i \int_0^t (v_{cmd} - v) d\tau \quad (20)$$

B. Road transformation

A common approach to enhance motion planners for planning on structured roads is to deform the motion plan to follow the shape of the road [2], [3], [8]. Inspired by this work, the proposed method simplifies the planning scenario by planning motion on a virtual straightened road. After defining a motion plan within this frame, it is bent back to the curved road.

Obviously, the piece-wise linear reference path cannot accurately follow the road curvature when motion is planned directly on the curved road. The virtual straight road allows the planner to remain using the piece-wise references, while now being able to construct references that accurately follow the road curvature. Additionally, planning directly on the curved road would result in corner-cutting when using distance-based optimization heuristics. The proposed approach avoids this issue.

Transforming a state ($x \in \mathbb{R}^{n_x}$) from the curved to the straight road involves the transformation of a pose ${}^{\mathcal{C}}(x, y, \theta)$ and the steering angle (δ). The remaining states ($v, a, \dot{a}$) do not change during the transformation.

1) Straightened road definition: First, a second-order spatial parametrization of the road center-line (21) must be obtained. This can be provided by perception hardware such as the Mobileye lane-detection system.

$$^{\mathcal{C}}y_{cl}(x) = c_2x^2 + c_1x + c_0 \tag{21}$$

Next, a straightened road is defined that originates at the y-axis (22) and is tangent to (21). See Figure 4.

$$^{\mathcal{S}}y_{cl}(x) = \frac{d^{\mathcal{C}}y_{cl}}{dx}\bigg|_{x=0} + c_0 = c_1x + c_0 \tag{22}$$

The connection between both roads is established by an intermediate Frénet frame (23). Here, ${}^{\mathcal{C}}(x, y, \theta)$ is the pose on the curved road, ${}^{\mathcal{F}}(S, \rho, \theta)$ the pose in the Frénet frame, and ${}^{\mathcal{S}}(x, y, \theta)$ the pose on the virtual straightened road.

$$^{\mathcal{C}}(x, y, \theta) \Longleftrightarrow {}^{\mathcal{F}}(S, \rho, \theta) \Longleftrightarrow {}^{\mathcal{S}}(x, y, \theta) \tag{23}$$

2) Arc-length parametrization: The Frénet frame link requires an arc-length parametrization for the curved and straightened road. Arc length of a curve is defined as (24).

$$S(x) = \int_0^{x_{max}} \sqrt{1 + \left(\frac{dy}{dx}\right)^2} dx \tag{24}$$

Here, x_{max} is chosen beyond the goal position. Equation 24 can be easily solved analytically for a first order equation. Hence, this approach can be used for the arc-length parametrization of (22). Solving (24) for the second order equation (21) requires solving a fifth order root which is generally too complex to solve analytically. Therefore, instead an approximate method is used for the curved road parametrization. A series of points (S, x) are generated by numerically calculating the arc-length of (21). These points are used for fitting a polynomial to obtain the approximate parametrization (25).

$$S(x) = s_2x^2 + s_1x + s_0 \tag{25}$$

3) Transformation: The transformation starts with locating the closest point on (21). Using (21) and (25), the pose is transformed to the local Frénet frame ${}^{\mathcal{F}}(S, \rho, \theta)$. The inverse of (22) is then used for transforming the pose to the straight road ${}^{\mathcal{S}}(x, y, \theta)$. Remaining is the transformation of δ, which is solved by subtracting the steer angle required for following the road center at S.

The introduced transformations are used for transforming all planner inputs to the straightened road. The CL-RRT then solves the motion planning problem and returns a trajectory ($\mathcal{T}$). Each point in $\mathcal{T}$ is then transformed back to the curved road by doing the inverse of the previously described transformations.

Additional path curvature (κ) is introduced when bending the motion plan to follow the road curvature. This is directly correlated with lateral acceleration: $a_y = u^2\kappa$. Hence, additional lateral acceleration is introduced during bending. This acceleration is included into the dynamics constraints by determining an upper bound. Path curvature is defined as $\kappa(x) = y''/(1+y'^2)^{\frac{3}{2}}$, where y = (21). The upper bound is determined by solving $d\kappa/dx = 0$.

C. Dealing with dynamic environments

The motion planner predicts future positions of obstacles and continuously re-plans the motion. The real-time motion planning framework in Algorithm 2 was implemented. First the planner updates all inputs (line 1). These inputs, together with the previously committed path, are transformed to the straightened scenario (lines 2-4). Committed path that has been passed is removed (line 5), and the tree is initialized (line 6). While time is available, the tree expands (line 7-8). The planner selects the best path and commits to a small section of the path (lines 9-11). Lastly, the motion plan is transformed to world coordinates and sent to the controller (lines 12-15).

Algorithm 2: Plan motion

```
1  Update $x_{car}$, $x_{goal}$, $X_{obs}$, $\mathcal{R}oad$
2  Transform motion plan ($\mathcal{MP}$) from World ($\mathcal{W}$) to Car
   ($\mathcal{C}$)
3  if $\mathcal{R}oad$ exists then
4      Transform $x_{goal}$, $X_{obs}$ and $\mathcal{MP}$ from $\mathcal{C}$ to $\mathcal{S}$
5  Delete parts of $\mathcal{MP}$ that have been passed
6  Initialize tree $\mathcal{T}$ with last node of $\mathcal{MP}$
7  while $t < t + \Delta t$ do
8      $ExpandTree(\mathcal{T})$
9  Select best path that reaches the goal
10 if $T_p < T_{commit}$ then
11     Commit to motion until $T_p + T_{new} \geq T_{commit}$
12 if $\mathcal{R}oad$ exists then
13     Transform committed path and $\mathcal{MP}$ from $\mathcal{S}$ to $\mathcal{C}$
14 Transform committed path and $\mathcal{MP}$ from $\mathcal{C}$ to $\mathcal{W}$
15 Send committed path to controller and add it to $\mathcal{MP}$
```

V. Simulation results

The proposed method is validated by evaluating its capability of planning parking and highway maneuvers. Any other environment can be considered as being a combination of the two. The coming sections will discuss the following scenarios:

- V-B: Parking with static obstacles
- V-C: Highway with static obstacles
- V-D: Dynamic obstacle avoidance

A. Simulation environment

Simulations were performed on a mobile workstation with i7-3630QM processor, running at 2.4Ghz. The computer has 8Gb memory. The simulations with static obstacles were programmed in MATLAB R2019b. The simulations with dynamic obstacles use a C++ (ROS) implementation that was optimized for real-time performance. The C++ planner is capable of exploring approximately 1000 nodes per second. It plans motion with an update rate of 5Hz.

TABLE I: Vehicle parameters

Parameter	Value	Unit
δ_{max}	0.52	rad
$\dot{\delta}_{max}$	0.3294	rad/s
T_d	0.3	$-$
T_a	0.3	$-$
a_{min}	-6	ms^2
a_{max}	2	ms^2
L	2.7	m
K_{us}	0.014	$-$
ρ	5.95	m
w_{body}	2	m
L_{body}	4.7	m
l_r	1	m

TABLE II: Planner parameters

Parameter	Value	Unit
a_{acc}	1	ms^2
a_{dec}	-1	ms^2
t_{min}	1	s
$a_{y,max}$	0.3	g
$t_{la,parking}$	1	s
$t_{la,highway}$	1.4	s
K_p	4	$-$
K_i	0.05	$-$
P_{exp}	0.7	$-$
P_{opt}	0.3	$-$

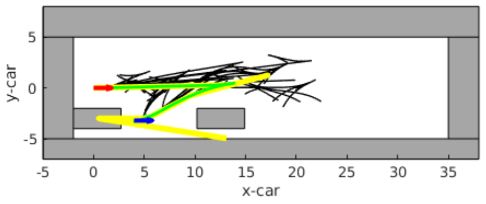


Fig. 6: Parallel parking: results for single motion query with 200 samples. Grey: obstacle bounding box, black: feasible trajectory, green: selected trajectory, yellow: reference of selected trajectory, red arrow: initial pose, blue arrow: goal pose.

The vehicle parameters used during simulation are defined in Table I. Here, w_{body} and L_{body} are the width and length of the vehicle body. l_r is the distance from the rear of the vehicle body to the rear axle coordinate frame.

The applied motion planner configuration parameters are defined in Table II.

B. Urban parking with static obstacles

Parking functionality of the CL-RRT has been previously demonstrated in [18]. Therefore, parking experiment results are discussed briefly only. The planner was tested with parallel and perpendicular parking. During parallel parking, the length and width of the parking spot are 7.5 and 2 meters respectively. For perpendicular parking, the parking spot width is set to be 4 meters.

Figure 6 shows a single motion plan query for the parallel parking scenario. The motion planner can plan trajectories that guide the vehicle into a tight parking spots. Notice here that the vehicle trajectory (green) is used during collision detection. Therefore, the reference (yellow) may cross the obstacle region (grey).

Motion was planned 100 times for each scenario, with the iteration limit set to 200. The planner was able to plan motion for the parallel and perpendicular scenarios with success rates of 71% and 83%, respectively.

C. Highway driving with static obstacle

A constant curvature highway is considered, consisting of 2 lanes. The planning horizon is set to 5 seconds. The tested scenarios were lane following and lane changing. An example of a considered scenario is shown in Figure 7. Notice here that the tree was built on the straightened road. Only the selected trajectory is deformed to follow the follow curved road.

1) Compared algorithms: The proposed method was compared with the standard RRT and CL-RRT, which both plan directly on the curved road. The algorithms all use the same optimization heuristics and collision detection.

The RRT uses a kinematic bicycle model without actuator constraints. During node expansion, a predefined set of steer inputs is simulated. The best input is selected and will not be considered again. The steer angles (δ_n) are linearly distributed over $[-\delta_{max}, \delta_{max}]$, with n = 1...11, and $\delta_{max} = 0.0312$ radians. The simulation horizon is set to 0.25 seconds.

2) Road configurations: The goal is located 150 meters ahead of the vehicle and can be on either of the lanes. The goal is to reach the goal region while minimizing (14). The vehicle has a constant speed of 120km/h.

In total, 20 roads are considered, with the radius linearly distributed between 450 and 5000 m. Each algorithm plans motion 100 times for each road configuration. An obstacle is randomly placed on either one of the two lanes and moves after each query. A total of 2000 motion queries are performed for each algorithm. The computation time for each query is constrained to 4 seconds.

3) Results: Figure 8 presents the main results. Plot a) shows the evolution of the path cost as computation time elapses. Plot b) shows the cost of the selected path versus the road radius. Both plots only consider motion queries that successfully reach the goal at some point during planning. The bold lines represent the means and the shaded areas the standard deviation. Additional results are presented in Table III.

D. Dynamic obstacle avoidance

The capability of dealing with dynamic obstacles is evaluated with a pedestrian avoidance scenario that was realized in the ROS Gazebo simulation environment. During this simulation, the goal is to drive to the end of the road while avoiding collision with two pedestrians (see Figure 9). The pedestrian cross the road at randomized moments.

An additional cost term is added to (14) to increase the distance between obstacles: $\sum_{p=0}^{p=N_p} w_4 exp(-w_5 D_{obs,p})$. Here, w_4 and w_5 are the cost weights and $D_{obs,p}$ the distance to obstacle p.

1) Results: A total of 100 simulations were performed with the MPCC configured for trajectory tracking only. From those simulations, three resulted in a collision, and seven came uncomfortably close to a pedestrian. Often the maneuvers performed for avoiding the pedestrian included a lot steering.

VI. Discussion

The results from the parking scenario show that the planner is able to plan with a relatively high success rate. Although the presented results for the parking scenarios appear to be of good quality, the planner can also produce paths that are sub-optimal. This can result in a trajectory that does more maneuvers than is necessary for parking the car. On structured highway roads, this

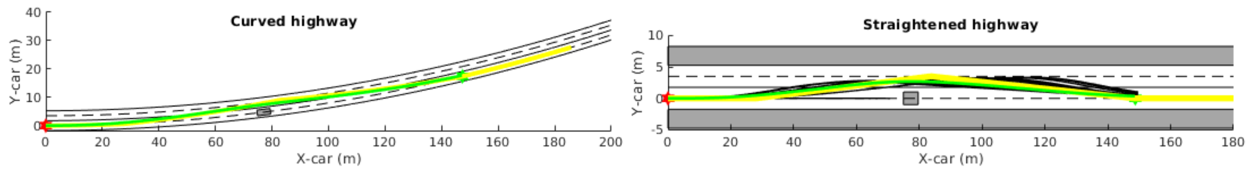
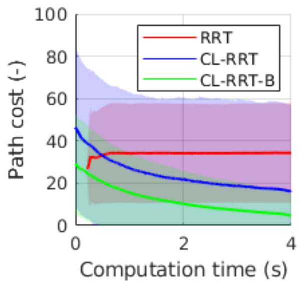


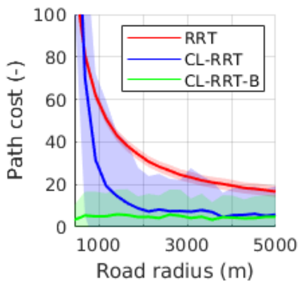
Fig. 7: Planned motion for curved highway. R = 750m, v=120km/h. Left: curved highway. Right: virtual straightened highway. Grey: obstacle bounding box, black: feasible trajectory, green: selected trajectory, yellow: reference of selected trajectory.

TABLE III: Statistic results of motion planning for a constant curvature highway for $R_n \in [450, 5000], n = 1...20$. The radii are linearly distributed. Each algorithm plans 100 times for each road configuration. Considered are 0 and 1 obstacles (randomized location for each query), and scenarios Lane Following (LF) and Lane Change (LC). Here, the proposed method is denoted with CL-B.

config.	% failures			Final cost: mean (Std)			Samples explored: mean (Std)			First goal comp. time: mean (Std)		
	RRT	CL	CL-B	RRT	CL	CL-B	RRT	CL	CL-B	RRT	CL	CL-B
0 obs, LF	13.8	6.0	**0.0**	32 (25)	9 (31)	**1** (2)	351 (27)	35 (6)	35 (7)	0.8 (0.7)	0.5 (0.5)	0.6 (0.3)
1 obs, LF	23.5	18.8	**1.2**	33 (23)	23 (46)	**10** (15)	330 (35)	20 (5)	22 (6)	1.2 (0.9)	1.2 (1.0)	1.2 (0.8)
0 obs, LC	19.4	0.1	**0.0**	36 (23)	10 (37)	**1** (3)	376 (20)	37 (6)	39 (7)	0.8 (0.8)	0.3 (0.3)	0.4 (0.2)
1 obs, LC	28.4	1.5	**0.3**	35 (22)	24 (48)	**7** (11)	363 (28)	20 (4)	23 (6)	1.1 (0.9)	0.9 (0.7)	1.0 (0.7)



a) Cost vs computation time



b) Cost vs radius

Fig. 8: Highway comparison results

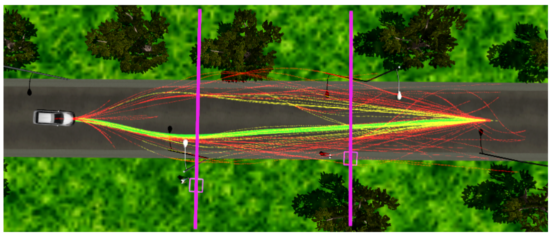


Fig. 9: Pedestrian avoidance simulation: red: feasible trajectory, yellow: feasible trajectory that reaches the goal, green: selected trajectory, pink: pedestrian path.

is less noticeable due to the simplistic geometry and heavily goal biased expansion.

The presented highway results in Figure 8 a) reveal that the proposed method (CL-RRT-B) plans paths of significantly lower cost compared to RRT and CL-RRT. It also shows that while CL-RRT and the proposed method improve the path quality over time, the RRT easily gets stuck in its initial solution. The presented highway results in Figure a) reveal that the proposed method (CL-RRT-B) plans significantly lower cost paths compared to RRT and CL-RRT. It also shows that while CL-RRT and the proposed method improve the path quality over time, the RRT easily gets stuck in its initial solution. The results also showed that the proposed method is able to accurately track the lane center line without any meandering.

Figure 8 b) clearly demonstrates the advantage of the proposed method. On curved roads, the RRT and CL-RRT plan paths with a considerably higher cost than the proposed method. The proposed method is capable of planning low-cost paths for the entire range of road curvatures that were considered. The plot shows that the CL-RRT cost converges to the proposed method as the radius increases. This can be easily explained by the fact that the deformation of the curved to the straight virtual road decreases as radius increases, and thus the CL-RRT-B essentially becomes the same as the CL-RRT. The Dutch directive for highway design states that the highway radius must be at least 750 meters [23]. Therefore, it can be concluded that the proposed method can plan significantly better on roads with realistic curvature, compared to the RRT and CL-RRT.

Table III reveals that the proposed method is more reliable (has fewer failures) and produces cheaper paths on the considered curved roads. Furthermore, the CL-RRT and CL-RRT-B explore significantly fewer samples than the RRT. This is obvious since a single sample leads to a significantly longer path section than the RRT.

Dealing with dynamic obstacles has proven to be challenging for the planner. The deterministic velocity profile can explain this. Because the velocity profile limits the velocity space exploration, the only option may be to drive around the obstacle. Therefore, the planner may resort to a path that requires large steering angles instead of e.g. braking and letting the pedestrian pass.

VII. Conclusion

This article proposed a motion planning framework that is based on the Closed-Loop RRT, and aims to improve path quality in structured environments with curved roads. A generic implementation was realized, which was evaluated in unstructured parking environments and structured highway roads. Simulation results showed that the proposed method significantly improves path quality on curved roads, compared to the RRT and Closed-Loop RRT. Simulations with moving pedestrians have demonstrated that the planner has limitations in its ability of dealing with dynamic obstacles. This is caused by limited exploration of the velocity space, making the planner prefer excessive steering maneuvers over changing its velocity.

To address the identified shortcoming, future research could focus on adapting the algorithm to explore the velocity dimension in a more comprehensive manner. Additionally, further research on the optimality and how to improve it may also be included.

REFERENCES

[1] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, pp. 1879–1884, 2009.

[2] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenét frame," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 987–993, 2010.

[3] M. McNaughton, C. Urmson, J. M. Dolan, and J. W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4889–4895, 2011.

[4] K. Zheng and S. Liu, "RRT based Path Planning for Autonomous Parking of Vehicle," *2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS)*, pp. 627–632, 2018.

[5] F. Esposto, J. Goos, A. Teerhuis, and M. Alirezaei, "Hybrid path planning for non-holonomic autonomous vehicles: An experimental evaluation," in *5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems, MT-ITS 2017 - Proceedings*, 2017, pp. 25–30.

[6] D. Ferguson, T. Howard, and M. Likhachev, "Motion Planning in Urban Environments," *Journal of Field Robotics*, vol. 25, 2008.

[7] Y. Kuwata, G. Fiore, J. Teo, E. Frazzoli, and J. How, "Motion planning for urban driving using RRT," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*. IEEE, 9 2008, pp. 1681–1686. [Online]. Available: http://ieeexplore.ieee.org/document/4651075/

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

[9] S. M. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," Tech. Rep., 1998.

[10] S. Lavalle, *Planning Algorithms*. Cambridge: Cambridge University Press, 2006.

[11] S. Karaman and E. Frazzoli, "Incremental Sampling-based Algorithms for Optimal Motion Planning," *International Journal of Robotics Research*, 5 2010. [Online]. Available: http://arxiv.org/abs/1005.0416

[12] C. Urmson and R. Simmons, "Approaches for Heuristically Biasing RRT Growth," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2, 2003, pp. 1178–1183.

[13] 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. [Online]. Available: http://arc.aiaa.org/doi/10.2514/2.4856

[14] K. Yang, "An efficient Spline-based RRT path planner for non-holonomic robots in cluttered environments," *2013 International Conference on Unmanned Aircraft Systems, ICUAS 2013 - Conference Proceedings*, pp. 288–297, 2013.

[15] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[16] ——, "Sampling-based optimal motion planning for non-holonomic dynamical systems," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5041–5047, 2013.

[17] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime Motion Planning using the RRT*," *IEEE Conference on Robotics and Automation*, 2011.

[18] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105–1118, 2009.

[19] P. Polack, F. Altche, B. DAndrea-Novel, and A. De La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2017, pp. 812–818.

[20] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model Predictive Contouring Control for Collision Avoidance in Unstructured Dynamic Environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4459–4466, 7 2019.

[21] A. Schmeitz, J. Zegers, J. Ploeg, and M. Alirezaei, "Towards a generic lateral control concept for cooperative automated driving theoretical and experimental evaluation," in *5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems, MT-ITS 2017 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., 8 2017, pp. 134–139.

[22] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, and J. How, "Motion Planning in Complex Environments Using Closed-loop Prediction," *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008. [Online]. Available: http://arc.aiaa.org/doi/10.2514/6.2008-7166

[23] Rijkswaterstaat (Ministerie van Infrastructuur en Milieu), "Richtlijn Ontwerp Autosnelwegen 2019," vol. 21, no. november, 2019.

Part II

Supplementary material to the article

Chapter 4

Vehicle stability analysis

4-1 Assumptions

During the derivation of the stability conditions, several assumptions are made that allow the use of the single-track vehicle model:

1. Vehicle roll, pitch and vertical motion are neglected
2. Suspension dynamics are neglected
3. Steer angles remain small
4. Longitudinal velocity is constant (or varies negligible slow)
5. The vehicle operates in the linear region of the tire response

4-2 Dynamic vehicle model

Assumptions 1 and 2 allow the modeling of the vehicle body as a single point mass located at the center of gravity. Furthermore, the wheels on the front and rear axles are merged. This gives the single-track (bicycle) model (see Figure 4-1).

The model linearizes trigonometric equations that are a function of the steering angle, and thus, the model can only be used when the steering angle remains small. Longitudinal dynamics are not included in the model. Hence, velocity must remain constant (or vary negligible slow). The complex nonlinear tire response is simplified by constraining the operation to the linear region only. This reduces the complexity and makes analytic analysis of the stability possible. Using these assumptions, the linear dynamic bicycle model can be derived (4-1).

$$\begin{bmatrix} \dot{v} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} -\frac{C_f+C_r}{mu} & \frac{l_rC_r-l_fC_f}{mu} - u \\ \frac{l_rC_r-l_fC_f}{I_zu} & -\frac{l_f^2C_f+l_r^2C_r}{I_zu} \end{bmatrix} \begin{bmatrix} v \\ r \end{bmatrix} + \begin{bmatrix} \frac{C_f}{m} \\ \frac{l_fC_r}{I_z} \end{bmatrix} \delta \tag{4-1}$$

The parameters are defined as follows. v is the lateral velocity, r is the yaw-rate, δ is the steering angle of the front wheels, C_f and C_r are the front and rear tire cornering stiffness, u

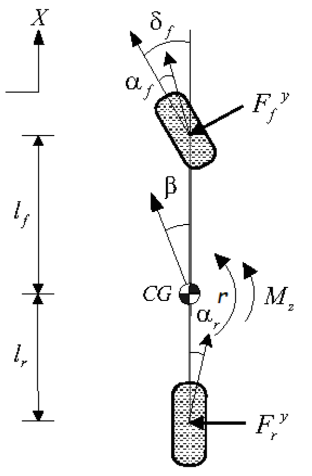


Figure 4-1: Single track vehicle model [36]

is the longitudinal velocity, m is the vehicle mass, l_r and l_f are the distances from the center of mass to the front and rear axle respectively. Lastly, I_z is the inertia around the vertical axis.

4-3 Stability Criterion derivation

The Routh-Hurwitz stability criterion is used for determining when the vehicle becomes unstable. Hence, the characteristic equation of the model must be defined (4-2) $det(sI - A) = 0$.

$$\begin{aligned} s^2 + \left(\frac{C_f + C_r}{mu} + \frac{l_f^2 C_f + l_r^2 C_r}{I_z u}\right)s + \left(\frac{C_f + C_r}{mu}\right)\left(\frac{l_f^2 C_f + l_r^2 C_r}{I_z u}\right) \\ -\left(\frac{l_r C_r - l_f C_f}{I_z u}\right)\left(\frac{l_r C_r - l_f C_f}{mu} - u\right) = 0 \end{aligned} \tag{4-2}$$

According to the Routh-Hurwitz criterion, a second-degree polynomial $P(s) = s^2 + a_1 s + a_0$ is stable if both coefficients satisfy $a_i > 0$, where $i = 1, 2$. Since all vehicle parameters are positive, clearly a_1 is always positive. Therefore, only condition (4-3) remains to be satisfied.

$$\left(\frac{C_f + C_r}{mu}\right)\left(\frac{l_f^2 C_f + l_r^2 C_r}{I_z u}\right) - \left(\frac{l_r C_r - l_f C_f}{I_z u}\right)\left(\frac{l_r C_r - l_f C_f}{mu} - u\right) > 0 \tag{4-3}$$

The first term is always positive because of the vehicle parameters. Simplifying the second part of (4-3) gives the final stability criterion (4-4). Hence, stability can be guaranteed as long as the assumptions are valid and the vehicle shows understeer behavior ($K_{us} > 0$).

$$l_r C_r - l_f C_f > 0 \tag{4-4}$$

4-4 Considering vehicle stability during planning

The vehicle considered in the article has a positive understeer gradient. Therefore, vehicle stability can be guaranteed as long as the linear dynamic bicycle model is an accurate approximation of the complex nonlinear vehicle dynamics. The assumptions in Section 4-1 must be valid.

The proposed motion planning method is constrained to planning under normal operating conditions. Highly dynamic scenarios such as emergency evasion maneuvers can not be planned. For normal operating conditions, the model used in the stability analysis is accurate enough. This is enforced by a lateral acceleration constraint during closed-loop prediction. Additionally, the velocity profile design allows the vehicle to have very limited longitudinal accelerations. Hence, the model can be used for stability analysis.

Chapter 5

Lateral Controller Implementation

The piece-wise linear nature of the reference generation results in non-smooth reference paths. The vehicle is not able to track this path accurately because of the nonholonomic constraints. By using a controller with a preview point, the path is smoothened and obtains continuous curvature. Hence, the vehicle can now track the path when appropriate controllers are applied. Furthermore, the planner has a strong bias to explore the goal region. To reach the goal with success, the vehicle must first align with the goal heading. By using the preview controller, the virtual car sees the goal coming and can align itself with the goal before reaching it.

The proposed algorithm has severe complexity, and computational resources are limited. Thus, to realize a real-time implementation, the algorithm is restricted to using simple controllers. Using complex controllers such as Model Predictive Control (MPC) would result in a too high computational burden. Furthermore, it has been demonstrated in [37] that the preview controller can outperform MPC for lateral control purposes.

The lateral controller requires a single measurement of the lateral error at the preview point, located at a look-ahead distance (D_{LA}) in front of the vehicle. The lateral error is calculated as follows. First, the point on the reference path is located that minimized the distance to the preview point. This point and its two adjacent points are transformed to the car coordinate frame followed by a translation to the preview point. A second-order Lagrange interpolation is then used for calculating the controller error.

The reference path is searched for the closest point during each calculation of the control error. This must be done during every step of the closed-loop prediction. Thus, numerous amounts of searches are performed during a motion query. In our implementation, this was of order 10^4 for a single motion query. The procedure therefore has a major impact on computational efforts during planning. To minimize this issue, the amount of points in a path is kept at a minimum. This is done by scaling path resolution with the vehicle velocity. The spatial resolution of the reference path is calculated with (5-1), where $t_{ref} = 0.2s$ and $res_{min} = 0.2m$.

$$res = max(|u|t_{ref}, res_{min}) \tag{5-1}$$

Chapter 6

Supplementary simulations

The scientific article presented in Part 1 discusses only the main results of the thesis. The purpose of this chapter is to provide specifics on the simulation setup and present additional simulation scenarios.

6-1 Perpendicular parking scenario

The parking scenarios discussed in the article were parallel parking and perpendicular parking. Here, an illustration is provided that presents the perpendicular parking scenario (see Figure 6-1).

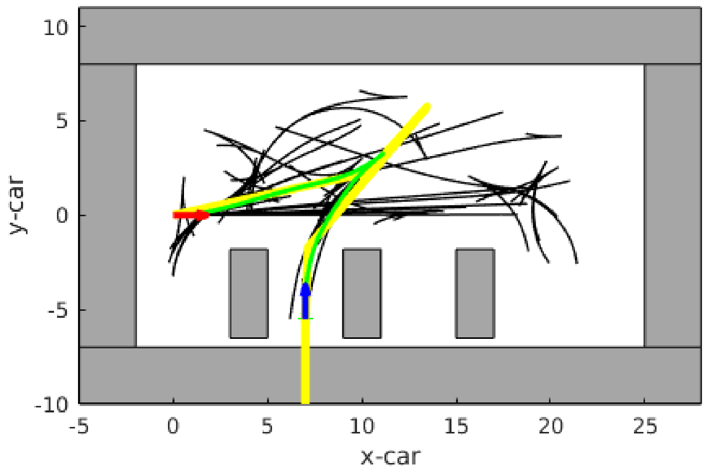


Figure 6-1: Perpendicular parking scenario: the parking spot is 4 meters wide and 5 meters long. Black: feasible trajectory. Green: selected trajectory. Yellow: reference of selected trajectory. Grey: obstacle bounding boxes. Red arrow: initial pose. Blue arrow: goal pose.

6-2 Real-time highway driving with Dynamic Obstacles

This section discusses an additional highway driving simulation that was performed. A MATLAB Simulink model simulates the vehicle dynamics, several perception sensors, and a mission planner. The generated information is sent to the motion planner which is implemented in ROS C++, which sends the motion plan back to MATLAB.

6-2-1 Driving scenario

A double-curved, two-lane highway was designed with the MATLAB Driving Scenario designer (see Figure 6-2). The road was generated with a continuous curvature profile (see Figure 6-4). The ego vehicle must overtake two slower moving vehicles that are driving on the most right lane. Therefore, the vehicle must perform two lane changes. While not performing any maneuvers, the path must follow the lane center as closely as possible. The lane changes are initiated by the mission planner when the vehicle approaches the obstacles. The ego vehicle has a constant speed of 120km/h and the obstacle vehicles 90km/h.

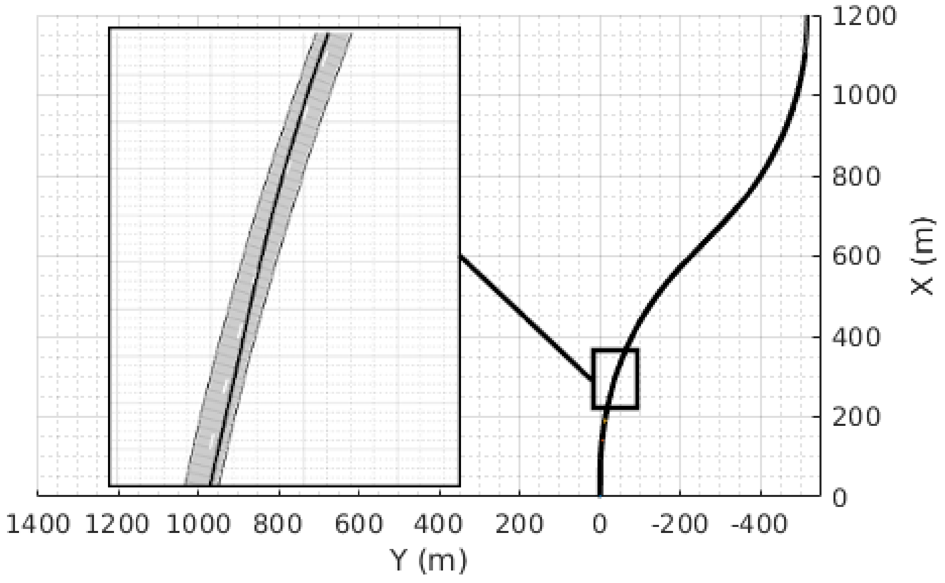


Figure 6-2: Real-time motion planning on double curved highway

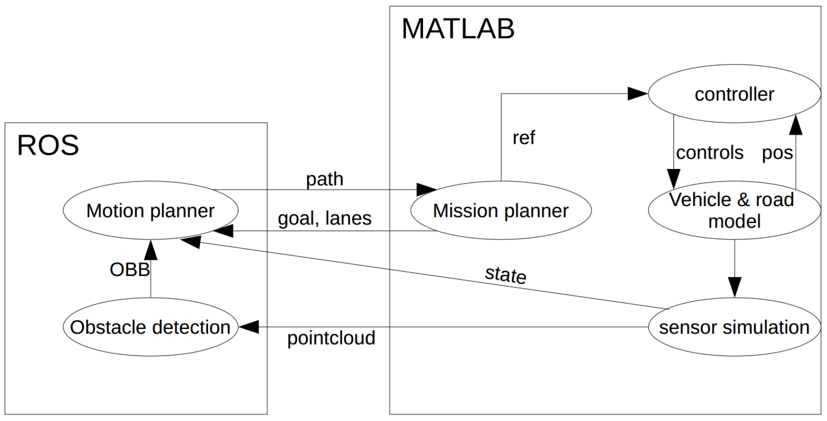


Figure 6-3: Simulation setup for real-time motion planning on highway

6-2-2 Implementation

An overview of the simulation setup is presented in Figure 6-3. Each of the components will be further discussed in the upcoming sections.

Vehicle model
The vehicle was modeled with a dynamic bicycle model. The vehicle operates with limited lateral acceleration. Therefore, the tires operate in the linear region and therefore a linear model is used. Longitudinal dynamics are neglected since the scenario considers constant speed maneuvers only.

Controller
Lateral control of the virtual vehicle is done with the same preview controller as is used during motion planning, see Chapter 5 and the article.

Mission Planner
The mission planner generates local goals for the motion planner. These goals are located 5 seconds ahead of the vehicle. During the first section of the simulation, the mission planner generates goals that are on the right-hand lane. When the vehicle approaches the upcoming obstacles, the mission planner initiates a lane change maneuver. This shifts the goal to the left lane. After passing the obstacle vehicles, the goal shifts back to the right lane. The goals are updated for each motion query.

Sensor simulation
The Simulink model generates sensor measurements that simulate point cloud data. The unprocessed point-cloud data is sent to the obstacle detection. Simulink also generates measurements that represent the lane markings. Lane coefficients are obtained by fitting a second-order polynomial through the center-line. The coefficients are provided to the motion planner.

Obstacle detection
Obstacle detection is implemented in ROS C++. A point cloud processing node is responsible for generating OBB for each obstacle. Obstacles are tracked to generate a prediction of their future positions. More details on the implementation can be found in Appendix A. The collision check itself, which is performed countless times during motion planning, is described in Appendix B.

6-2-3 Results

The results for a single simulation are presented in figure 6-4. The plot presents the curvature of the road and the vehicle. Besides that, it shows the lateral acceleration. The first peak in the curvature indicates the moment that the vehicle first encounters the slower moving vehicles. Here, the first lane change is initiated (see Figure 6-2). The second peak indicates the moment that the vehicle has passed both slower moving vehicles. It then performs the second lane change.

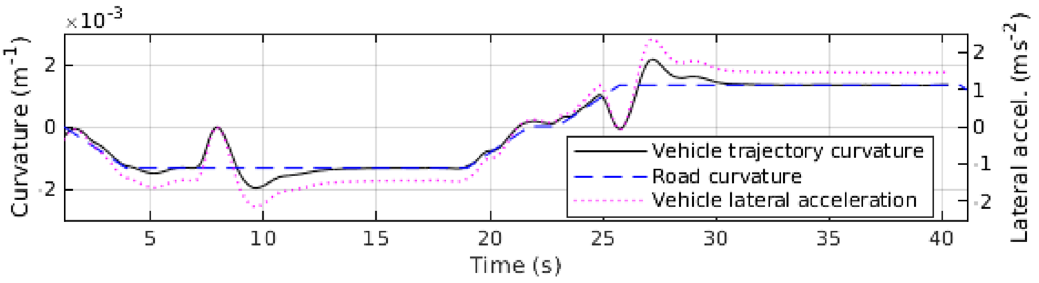


Figure 6-4: Real-time highway simulation results

The online highway driving results demonstrate the capability of planning motion on realistic highway roads. The planner is able to plan motion with smooth curvature, which accurately follows the road curvature during lane following. The results show that the planned motion converges to the road center-line, while avoiding collision with dynamic obstacles and satisfying the stability constraints.

Chapter 7

Setup of the Statistical Analysis

The sampling function that the proposed method uses has a great impact on its behavior. Due to its probabilistic nature, the planned paths can vary a considerable amount between subsequent identical motion queries. Before any conclusions can be drawn on the performance and properties of the algorithm, a comprehensive statistical analysis must be performed. The presented article discusses the results of the analysis. This chapter goes into detail how the setup for the analysis was realized.

7-1 Approximate steering RRT

The approximate Rapidly-exploring Random Tree (RRT) uses a forward simulator based on the kinematic vehicle. The model used during the simulations is presented in 7-1.

Nonholonomic RRT
$\dot{x} = u\cos(\theta)$
$\dot{y} = u\sin(\theta)$
$\dot{\theta} = \dfrac{u\tan(\delta)}{u^2\frac{K_{us}}{g} + L}$

Table 7-1: Approximate RRT vehicle model

The behavior of the planner can be tuned by changing the following parameters: $\Delta t, u, \delta$.

7-2 Parameter selection

The configuration of the planner has a major impact on its performance. Since only constant velocity is considered, the selection of u is straight forward. The selection of Δt should be approached more carefully. When the simulation time is chosen too short, the tree will contain so many nodes that algorithm exploration is slowed. This is due to the number of distance measures performed during node sorting. Furthermore, it can produce a path that shows a lot of meandering. If the time is chosen too long, the algorithm may encounter problems regarding limited exploration. Extensive testing resulted in the final parameters selection presented in Table 7-2.

Parameter	Value
Δt	0.25 s
u	33 m/s
δ_{max}	0.0312 rad
N	11

Table 7-2: Approximate RRT configuration

Parameter selection for the RRT algorithm has proven to be a delicate process. The configuration has a major impact on performance. In contrast, configuring the Closed-Loop RRT (CL-RRT) was quite simple. The algorithm performs rather well with merely a little tuning.

Part III

Appendices

Appendix A

Obstacle detection implementation

The obstacle detection node was implemented using the ROS Point Cloud Library[1]. It uses LIDAR point cloud sensor data for generating 2D Oriented Bounding Box (OBB). First, the ground plane is filtered from the point cloud. The point cloud is then clustered into several groups that represent the obstacles. Lastly, a 2D OBB is constructed around the clustered points. By default, the boxes are aligned with the vehicle heading. When road center-line coefficients are available from the perception part of the vehicle, the bounding boxes are aligned with the road heading.

Obstacle tracking is realized by combining the OBB generator with a Kalman Filter (KF), in order to estimate the internal state of the obstacle. For details on how this method works, the reader is referred to [2]. A C++ implementation[3] is combined with our obstacle tracker. For reproducability, the equations representing the obstacles are given below.

The obstacle of which we estimate its internal state is modeled using equation A-1:

$$\begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}_k = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}_{k-1} + w_{k-1} \tag{A-1}$$

The system is observed using measurement equation A-2:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}_k + v_k \tag{A-2}$$

[1]ROS, "ROS Point Cloud Library", http://wiki.ros.org/pcl

[2]Welch, Bishop, 2006, "An Introduction to the Kalman Filter", http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf

[3]Martiros, H, "Kalman Filter", https://github.com/hmartiro/kalman-cpp

Appendix B

Collision detection implementation

The task of the collision detection module is to determine whether a generated trajectory will lead to collision with obstacles present in its environment. Both the vehicle and obstacles are modeled as OBB. The obstacle detection node estimates the internal state of each obstacle, giving an estimate of its velocity.

Let $\mathcal{T}$ be a vehicle trajectory consisting of N points $(x, y, \theta, \delta, v, a, t)$, representing the state of the vehicle over time, and let $\mathcal{O}$ be the set of all static and dynamic obstacles in the vehicle's environment. The task of the collision module is to determine whether each point in $\mathcal{T}$, leads to a collision with an obstacle.

This problem can be reduced to determining whether a pair of OBB intersect. The process of determining whether a pair of boxes intersect works as follows:

First, we predict the future position of the obstacle that we are checking for collisions. We know the position and orientation of the vehicle, and how long it will take for the vehicle to arrive to that point. Using this time, and a linear extrapolation with the obstacle' velocity, we determine the future position of the obstacle (see Equation B-1).

$$\begin{bmatrix} x \\ y \end{bmatrix}_{prediction} = \begin{bmatrix} x \\ y \end{bmatrix}_{t0} + \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}_{t0} \Delta t \tag{B-1}$$

We then construct an upper bound of the box dimensions, and use it to determine whether we should perform a more complex and accurate collision check. This upper bound is constructed by fitting circles around both boxes. Checking if these two circles intersect is an operation that can be very efficiently done.

Only when the simple test tells us that both circles intersect, we advance to the more complex test. This second test is based on the Separating Axis Theorem (SAT) theorem, and checks whether the boxes themselves intersect. This intersection test is efficiently implemented based on the SAT [1].

[1]Huynh, J. (2009). Separating axis theorem for oriented bounding boxes.

Bibliography

[1] Rijkswaterstaat (Ministerie van Infrastructuur en Milieu), “Richtlijn Ontwerp Autosnelwegen 2019,” vol. 21, no. november, 2019.

[2] S. Lavalle, *Planning Algorithms.* Cambridge: Cambridge University Press, 2006.

[3] S. LaValle and J. Kuffner, “Randomized Kinodynamic Planning,” *International Conference on Robotics & Automation*, vol. 1, pp. 473–479, 1999.

[4] S. M. Lavalle and J. J. Kuffner, “Randomized Kinodyamic Planning,” *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[5] S. Lazard, J. Reif, and H. Wang, “The Complexity of the Two Dimensional Curvature-Constrained Shortest-Path Problem,” *Proceedings of the Third International Workshop on the Algorithmic Foundations of Robotics*, pp. 49–57, 1998.

[6] S. M. LaValle and J. J. Kuffner, “Rapidly-Exploring Random Trees: Progress and Prospects,” *Algorithmic and Computational Robotics: New Directions*, 2000.

[7] S. Thomas, M. Morales, X. Tang, and N. M. Amato, “Biasing samplers to improve motion planning performance,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1625–1630, 2007.

[8] M. Elbanhawi, M. Simic, and R. Jazar, “Continuous-curvature bounded trajectory planning using parametric splines,” *Frontiers in Artificial Intelligence and Applications*, vol. 262, no. July 2015, pp. 513–522, 2014.

[9] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, “Real-time motion planning with applications to autonomous urban driving,” *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105–1118, 2009.

[10] L. Dubins, “On Curves of Minimal Length with a Constraint on Average Curvature , and with Prescribed Initial and Terminal Positions and Tangents,” *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[11] S. M. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," tech. rep., 1998.

[12] R. Pepy, A. Lambert, and H. Mounier, "Path Planning using a Dynamic Vehicle Model," *2006 2nd International Conference on Information & Communication Technologies*, vol. 1, no. 1, pp. 781–786, 2006.

[13] J. A. Reeds and L. A. Shepp, "Optimal Paths for a car that goes both Forwards and Backwards," *Pacific Journal of Mathematics*, vol. 145, no. 2, 1990.

[14] A. Fraichard, Thierry; Scheuer, "From Reeds and Shepps to Continuous-Curvature Paths," *IEEE Transactions on Robotics*, vol. 20, no. 6, pp. 144–149, 2004.

[15] H. Banzhaf, N. Berinpanathan, D. Nienhüser, and J. Marius Zöllner, "From G2 to G3 Continuity: Continuous Curvature Rate Steering Functions for Sampling-Based Nonholonomic Motion Planning," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2018-June, no. Iv, pp. 326–333, 2018.

[16] K. Yang and S. Sukkarieh, "An analytical continuous-curvature path-smoothing algorithm," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 561–568, 2010.

[17] A. Trent, R. Venkataraman, and D. Doman, "Trajectory generation using a Modified Simple Shooting Method," *IEEE Aerospace Conference Proceedings*, vol. 4, pp. 2723–2729, 2004.

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

[19] K. Yang, S. Moon, S. Yoo, J. Kang, N. L. Doh, H. B. Kim, and S. Joo, "Spline-based RRT path planner for non-holonomic robots," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 73, no. 1-4, pp. 763–782, 2014.

[20] M. Reggiani, M. Mazzoli, and S. Caselli, "An experimental evaluation of collision detection packages for robot motion planning," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2329–2334, 2002.

[21] J. Bialkowski, S. Karaman, M. Otte, and E. Frazzoli, "Efficient collision checking in sampling-based motion planning," *Springer Tracts in Advanced Robotics*, vol. 86, pp. 365–380, 2013.

[22] U. Schwesinger, R. Siegwart, and P. Furgale, "Fast collision detection through bounding volume hierarchies in workspace-time space for sampling-based motion planners," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 63–68, 2015.

[23] Peng Cheng and S. LaValle, "Resolution complete rapidly-exploring random trees," *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 1, no. May, pp. 267–272, 2002.

[24] P. Cheng, *Sampling-Based Motion Planning with Differential Constraints.* PhD thesis, University of Illinois at Urbuna-Champaign, 2005.

[25] A. Shkolnik, M. Walter, and R. Tedrake, "Reachability-guided sampling for planning under differential constraints," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2859–2865, 2009.

[26] L. Jaillet, J. Hoffman, J. Van Den Berg, P. Abbeel, J. M. Porta, and K. Goldberg, "EG-RRT: Environment-guided random trees for kinodynamic motion planning with uncertainty and obstacles," *IEEE International Conference on Intelligent Robots and Systems*, pp. 2646–2652, 2011.

[27] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[28] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime Motion Planning using the RRT*," *IEEE Conference on Robotics and Automation*, 2011.

[29] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *Proceedings of the IEEE Conference on Decision and Control*, 2010.

[30] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," *IEEE International Conference on Intelligent Robots and Systems*, pp. 2997–3004, 2014.

[31] J. Denny, M. Morales, S. Rodriguez, and N. M. Amato, "Adapting RRT growth for heterogeneous environments," *IEEE International Conference on Intelligent Robots and Systems*, pp. 1772–1778, 2013.

[32] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, O. Koch, Y. Kuwata, D. Moore, E. Olson, S. Peters, J. Teo, R. Truax, and M. Walter, "A Perception-Driven Autonomous Urban Vehicle," *Journal of Field Robotics*, vol. 25, no. 10, pp. 727–774, 2008.

[33] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, and J. How, "Motion Planning in Complex Environments Using Closed-loop Prediction," *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008.

[34] 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.

[35] O. Arslan and P. Tsiotras, "Use of relaxation methods in sampling-based algorithms for optimal motion planning," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2421–2428, 2013.

[36] K. Nam, H. Fujimoto, and Y. Hori, "Motion control of electric vehicles based on robust lateral tire force control using lateral tire force sensors," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, pp. 526–531, 2012.

[37] Z. Lu, B. Shyrokau, and B. Boulkroune, "Performance Benchmark of state-of-the-art Lateral Path-following Controllers," in *IEEE 15th International Workshop on Advanced Motion Control*, (Tokyo, Japan), pp. 541–546, 2018.