Swarm-Based Trajectory Planning for Autonomous Cars

Dissertation zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)

vorgelegt von

Jan Fritz Ulbrich

am

Fachbereich Mathematik und Informatik
Freie Universität Berlin

June 14, 2022

Erstgutachter

Prof. Dr. Raúl Rojas
Freie Universität Berlin
Fachbereich Mathematik und Informatik

Zweitgutachter

Prof. Dr. Hans-Dieter Burkhard
Humboldt-Universität zu Berlin
Institut für Informatik

Tag der Disputation: 03.06.2022

Selbstständigkeitserklärung

Name: ______________________

Vorname: ____________________

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Dissertation selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht. Diese Dissertation wurde in gleicher oder ähnlicher Form noch in keinem früheren Promotionsverfahren eingereicht.

Mit einer Prüfung meiner Arbeit durch ein Plagiatsprüfungsprogramm erkläre ich mich einverstanden.

________________	________________
Datum	Unterschrift

. . . .

Declaration of authorship

Name: ______________________

First name: ____________________

I declare to the Freie Universität Berlin that I have completed the submitted dissertation independently and without the use of sources and aids other than those indicated. The present thesis is free of plagiarism. I have marked as such all statements that are taken literally or in content from other writings. This dissertation has not been submitted in the same or similar form in any previous doctoral procedure.

I agree to have my thesis examined by a plagiarism examination software.

________________	________________
Date	Signature

Acknowledgments

This thesis results from my work in the AutoNOMOS Labs working group at Freie Universität Berlin. I would like to express my deepest thanks and appreciation to Raúl Rojas, head of the group and my doctoral advisor. He gave me unbelievable amounts of freedom while assisting me whenever I needed direction and guidance. Writing this thesis would not have been possible without his support. Moreover, he enabled me to do things that no one had done before.

Throughout the writing of this dissertation, many of my colleagues at Freie Universität assisted and guided me along the way, and I am grateful to all of them. In particular, I cannot begin to express my thanks to Daniel Göhring, with whom I spend thousands of kilometers driving without anyone touching the steering wheel and many more hours in a car not moving at all. I miss his mental compendium of formulas and our stimulating discussions very much. I must also thank Tobias Langner, who gave insightful feedback on my ideas and fixed my code on many occasions. Special thanks go to Tinosch Ganjineh. I very much appreciated his inspiring spirit in the most incredible adventures we shared.

Summary

This thesis aims to enable efficient trajectory planning for autonomous vehicles without the requirement of a map or prior knowledge of the environment. For this purpose, an approach is presented, which adapts the concept of trail pheromones used by ants as well as the collective animals' flocking behavior to the domain of autonomous vehicles. In this way, the drivers of surrounding vehicles are used as additional input for anticipatory driving, expanding the capabilities of the individual autonomous car: a typical achievement of swarm intelligence.

While map-based trajectory planning has many advantages, there are situations when no map is available or localization in a map is too inaccurate. Also, the actual behavior of road users may differ significantly from the given map. Existing approaches for planning driving maneuvers in urban traffic without a map decouple lateral and longitudinal planning. Usually, knowledge of road geometry is assumed. This thesis presents an approach to overcome those limitations, adopting the established theory of elastic bands to implement swarm-based trajectory planning. The vehicle's dynamic restrictions and the driver's preferences are represented with a comprehensive set of parameterized objective functions. A swarm-based motion prediction algorithm is introduced to predict the surrounding vehicles' trajectories, and a heuristic is presented to choose the best candidate for a leader vehicle based on weighted criteria. The approach is evaluated in simulation, on recorded data, and live field tests in real traffic. The experimental results show that the presented approach, implementing a swarm behavior for autonomous cars, is valid to temporarily compensate for the advantages of map-based planning.

Zusammenfassung

Ziel dieser Arbeit ist, eine effiziente Trajektorienplanung für autonome Fahrzeuge zu ermöglichen, ohne dass eine Karte oder Vorkenntnisse der Umgebung erforderlich sind. Zu diesem Zweck wird ein Ansatz vorgestellt, der das Konzept der Spurpheromone von Ameisen, sowie das Schwarmverhalten der Vögeln und Fischen für den Bereich autonomer Fahrzeuge adaptiert. Die Fahrer der umgebenden Fahrzeuge werden als zusätzliche Informationsquellen verwendet, wodurch die Fähigkeiten für vorausschauendes Fahren des einzelnen autonomen Autos erweitert werden: eine typische Errungenschaft der Schwarmintelligenz.

Hochgenaue Karten bieten viele Vorteile für die vorausschauende Trajektorienplanung. Es gibt jedoch Situationen, in denen keine Karte verfügbar ist oder die Lokalisierung in einer Karte zu ungenau ist. Auch kann das tatsächliche Verhalten der Verkehrsteilnehmer erheblich von der angegebenen Karte abweichen. Bestehende Ansätze zur Planung von Fahrmanövern im Stadtverkehr ohne Karte betrachten die Planung in Längs- und Querrichtung separat. Zudem wird in der Regel die Kenntnis der Straßengeometrie vorausgesetzt. Um diese Einschränkungen zu überwinden, wird in dieser Arbeit eine schwarmbasierte Trajektorienplanung, basierend auf der etablierten Theorie der Elastic Bands, vorgestellt. Die Grenzen der Fahrzeugdynamik sowie Vorlieben des Fahrers hinsichtlich des Fahrverhaltens werden hierbei mit parametrisierten Kostenfunktionen dargestellt. Außerdem wird ein schwarmbasierter Algorithmus zur Vorhersage der Bewegung von Fahrzeugen in der Umgebung eingeführt, sowie eine Heuristik vorgestellt, um den besten Kandidaten für ein Führungsfahrzeug auszuwählen. Der Ansatz wird in Simulationen, auf aufgezeichneten Daten und in Feldtests in realem Verkehr evaluiert. Die experimentellen Ergebnisse zeigen, dass der vorgestellte Ansatz eines Schwarmverhaltens für autonome Fahrzeuge die Vorteile einer kartenbasierten Planung vorübergehend kompensieren kann.

Contents

Summary **v**

1 Introduction **1**
- 1.1 Motivation 1
- 1.2 Swarm-Based Trajectory Planning for Autonomous Cars 4
- 1.3 Thesis Contributions 6

2 Related Work and State of the Art **7**
- 2.1 Swarm-Based Trajectory Planning 7
- 2.2 Elastic Bands For Trajectory Planning 11
- 2.3 Hardware and Software Framework 18
  - 2.3.1 The Automomous Car MadeInGermany 18
  - 2.3.2 The FUB_ROSCAR Software Framework 20

3 Swarm-Based Prediction of Dynamic Objects **22**
- 3.1 Swarm-Based vs. Velocity-Based Prediction 24
- 3.2 Tracking Dynamic Objects 26
- 3.3 Clustering Objects on the Same Road 27
- 3.4 Determining Reference Trajectories 28
- 3.5 Sampling from the Reference Trajectory 31

4 Swarm-Based Trajectory Planning (STEBLE) **35**
- 4.1 STEBLE - Elastic Bands with Fixed Time Interval and Flexible Goal . 37
  - 4.1.1 Elastic Bands for Trajectory Planning 38
  - 4.1.2 Using a Fixed Time Interval and Flexible Goal . . . 39
  - 4.1.3 Trajectory Optimization as Graph Optimization Problem 40
- 4.2 The Objective Functions 42
  - 4.2.1 Non-Holonomic Kinematics and Turning Radius . 44
  - 4.2.2 Forward Driving 47

4.2.3 Limiting Velocity and Acceleration 49
Maximum Velocity 49
Optimal Velocity 51
Longitudinal and Angular Acceleration 52
Centripetal Acceleration 53
4.2.4 Avoiding Obstacles 55
Geometric Representation of Objects 55
Dynamic Longitudinal Safety Distance 58
Summing Up the Objects' Penalties 60
4.2.5 Following Trajectories of Other Vehicles 65
4.3 Selecting A Target to Follow 69
4.4 Initialization of the Elastic Band (CSTT) 72
4.4.1 Pruning the Target Trajectory 74
4.4.2 Sampling With Constant Time Intervals 78
4.5 Validation and Pruning of the Optimized Elastic Band . . . 84
4.6 Simultaneous Optimization of Multiple Trajectories 86

5 Evaluation and Experimental Results **91**
5.1 Description of the Evaluated Data Sets 91
5.1.1 Data Set A 93
5.1.2 Data Set B 95
5.1.3 Data Set C 97
5.2 Evaluation of the Swarm-Based Prediction 99
5.3 Evaluation of STEBLE Trajectory Planning 102
5.3.1 Parameters, Weights and Thresholds 103
General Parameters 103
Weights and Thresholds of the Objective Functions 105
5.3.2 Statistical Analysis of the Objective Function 108
5.3.3 General Validity of the Generated Trajectories . . . 113
5.3.4 Evaluation of the Path Following Accuracy 118
5.3.5 Comparison with Human Drivers' Trajectories . . . 120

6 Conclusions **123**

Appendix A Implementation of Distance Calculations **126**

Bibliography **128**

Chapter 1

Introduction

1.1 Motivation

AutoNOMOS Labs is researching autonomous cars since 2007. Our working group has gathered much experience during the last ten years by testing algorithms in real traffic with three different autonomous vehicles (Sprit Of Berlin, MadeInGermany, and e-Instein). We have tested our cars on roads in four countries (Germany, Switzerland, USA, Mexico). Since all trajectory planning algorithms evaluated by the working group for autonomous driving rely heavily on lane-accurate maps, up-to-date maps with very high precision are indispensable. Another prerequisite is highly accurate localization of the vehicle to match the position in those maps. Both requirements often become a significant problem since both can not always be guaranteed due to several reasons. This thesis aims to provide a fall-back solution for trajectory planning if an autonomous car cannot localize itself on a map. The basic idea is to use other drivers as a source for environment perception, especially their knowledge of local conditions and their skill to interpret unusual situations and thus achieve a solution to the trajectory planning problem, which does not rely on a map, but on social awareness.

As stated above, previous solutions of the working group for trajectory planning for autonomous cars - and nearly all other currently researched solutions - are based on a representation of the environment via a map. Using a map enables the planner to consider properties of the planning area far ahead, such as the drivable space, dynamic and static obstacles that can be expected on the way, or speed limits. This makes it possible to plan with much higher velocities since situations where the car needs to slow down, e.g., curves or crossings roads, can be predicted reliably. Maps for autonomous driving typically describe the road network and its properties with lane-level accuracy. Consequently, it is a very time-consuming process to build those maps, depending heavily on precisely georeferenced data. During tests in several countries around the globe, data from several different runs at different dates was usually needed to compensate drifts in the GPS position due to occlusion or reflection of the satellite signal on buildings and other vehicles. Using high precision real-time kinematic (RTK) GPS systems cannot prevent those effects due to their spatial and temporal locality. Also, the process of connecting the roads and drivable areas correctly, e.g., at complex junctions, is a task that is very difficult to automate. When we recorded data to map about 2.400 km roads in Mexico, we found that already one month later several road sections were undrivable for our autonomous vehicle MadeInGermany, because the traffic was diverted to the oncoming lanes due to construction work. Due to the great expense of maintaining maps, up-to-date detailed and accurate maps will not be available for all areas in the near future.

A critical factor when using maps is localization. The full potential of highly accurate maps can only be used if the car is able to localize itself equally accurately within those maps. Unfortunately, no solution can guarantee the required accuracy for the localization of autonomous vehicles at all times. GPS, the primary source for global localization, relies on an unobstructed line of sight to four or more satellites. The accuracy of GPS can be enhanced to lane-level precision using real-time kinematic (RTK) positioning, which relies on corrections to the satellite's signals derived from local reference stations. Practically this approach does not always work as expected. First of all, the RTK data (i.e., reference stations) must be available for the region with a sufficient resolution. Secondly, the correction signal has to be transmitted to the GPS receiver mounted on the car - either via a 3G connection or by satellite. This process is prone to reception errors. During our tests in different countries, we used various variants of RTK protocols, but none of them provided a completely reliable and stable localization with an accuracy below 1.5 meters. Even if RTK positioning works as expected, the satellite signal is still prone to occlusion or reflection, which cannot be mitigated by information from reference stations. This is especially relevant in an urban environment with tall buildings.

Another approach for positioning, widely used in the domain of robotics, is feature-based localization. Local landmarks are observed by sensors and matched to the corresponding landmarks stored in a map. We have successfully used lane markings, curbs, poles, building corners, and image features such as SURF or ORB for this purpose in our workgroup. Nonetheless, tests involving feature-based localization were always restricted to relatively small areas due to the requirement to build high-quality feature maps and keep them up-to-date. Depending on the feature type, the availability of the features is also a problem: Not all roads have lane markings, buildings, or poles (e.g., trees) or visually unique features. Landmarks could also be occluded temporarily by other traffic participants, weather conditions, or have changed permanently.

In addition to the challenges of maintaining a map and guaranteeing a proper global localization, there is also the case that the lanes actually used by human drivers differ from the officially designated road map. During heavy snowfall in Berlin, we encountered the situation that the human drivers created two lanes in the center of the road - to avoid the heaps of snow at the borders. The original lane markings, which defined three lanes, were invisible under the snow, and our autonomous car was the only one following them.

Apart from situations where the mapped roads are blocked or shifted due to temporal obstructions, there are cases where no valid road map can be created altogether. One example is the roundabout at the Ángel de la Independencia in Mexico City, where no lane markings are available. Because there are no fixed lanes, the human drivers create them instantaneously. Depending on the traffic density, the four lanes entering the roundabout could be merged into two lanes or split up into six lanes. Following the expected four lanes, our autonomous car was often perceived as an obstacle by the local drivers, hindering traffic flow. Those situations demonstrate a need for some kind of swarm behavior for autonomous cars - even if perfect maps and localization may be available in the future.

1.2 Swarm-Based Trajectory Planning for Autonomous Cars

Human drivers have a simple but efficient strategy if they cannot localize themselves on the map they have memorized: They follow other cars, i.e., they adapt to the behavior of other drivers who may have more knowledge of the area. Drivers who are familiar with the local conditions know where they can accelerate safely and where they have to slow down. They also know where and how to drive in unusual situations, such as partially closed or diverted lanes due to construction sites. This knowledge can be used, similar to maps, to plan ahead of the close sensor range and thus enable a more efficient driving behavior. Integrating into the swarm of local drivers also benefits the safety aspect. A driver who adopts the local driving behavior is more predictable for the other road users.

Of course, human drivers do not follow arbitrarily observed cars blindly. They also take into account common sense and their own driving preferences, as well as traffic rules. This leads to a set of preferences and restrictions which may be ambiguous or even contradict themselves. The main challenge is how to weigh those parameters and maintain reasonable safety and comfort properties while integrating as much as possible into the swarm of local road users.

When talking about swarms, the flocking (or schooling) behavior of some birds or fish comes to mind: large groups of individual entities move collectively, according to some unspoken rules, in order to preserve energy, deceive predators, or hunt prey. While it is not possible to directly copy a specific animals' behavior, there are aspects of the flocking behavior which can be transferred to driving cars. Human drivers also adapt to other cars surrounding the ego vehicle. They align with the other cars and adjust their velocity. They also avoid getting too close to other individuals. The aspect of being attracted by the center of the swarm may be useful when trajectory planning can benefit from the knowledge of local drivers in the vicinity. On the other hand, it is not always desirable to be surrounded by other traffic participants. Even when following a specific vehicle, human drivers often choose positions with more free space around them (e.g., the empty neighbor lane) to better perceive the oncoming road or more options to avoid obstacles. Also, human drivers consider many other indicators where and how to drive, e.g., road markings and traffic signs. This is a major difference from the unstructured environment where most animals navigate.

However, when taking the concept of roads into account, another well-documented example of animals achieving swarm intelligence exists: many species of ants use trail pheromones to guide their fellow members to food sources and around obstacles. Based on this approach, the ants are straightening out roads and cut shortcuts to ensure fast access to foraging areas. Some species even maintain “highways” with multiple lanes. The concept of trail pheromones can be used to create a momentary map of the surrounding road geometry: the area where other cars were moving seconds ago should also be drivable for the autonomous car. Similar to the ants’ trails, if more individuals use a specific path, it should be better in some aspect (e.g., safety or efficiency). The virtual pheromones left by observed cars can include additional information on velocity, tracking status, or classification confidence. This information can then be used to filter only the safest and fastest trails to follow.

The objective of this thesis is to enable efficient trajectory planning when no map is available, localization in a map is not possible, or the actual behavior of road users differs from the given map. An approach is presented in the following, which adapts the concept of flocking and trail pheromones to autonomous cars and thus achieves a kind of swarm behavior. Human drivers use many sources of information on the structure and characteristics of the road that influence their driving behavior, e.g., lane markings and traffic signs, weather conditions, and their experience and knowledge of the local practice. Nonetheless, this work focuses on a single high-level type of input for the presented planning modules: objects with associated classification (e.g., car, truck, bicycle, or pedestrian), contour, velocity, and acceleration. In a way, all of the additional information available to human drivers is implicitly encoded in the behavior of the dynamic objects. While non-vehicle objects are regarded solely as obstacles and consequently avoided, the proposed behavior follows the paths of observed vehicles and takes into account their velocity and acceleration. Hence, not only the road geometry is perceived, but also the traffic rules and characteristics (e.g., reasonable speed limits). This can be seen as expanding the capabilities of the individual autonomous car by using the group of other drivers as additional sensors: a typical achievement of swarm intelligence.

1.3 Thesis Contributions

The main contributions of this thesis can be summarized as follows:

- A swarm-based trajectory planning approach for autonomous vehicles combining lateral and longitudinal control is presented.
- The proposed trajectory planning requires no map of the road network or prior knowledge of the environment.
- The vehicle's dynamics are taken into account for the trajectory planning. They are parameterized and can be adapted to the drivers' preferences.
- A swarm-based motion prediction algorithm is introduced to predict the surrounding vehicles' trajectories.
- A heuristic algorithm is presented to choose the best candidate for a leader vehicle based on weighted criteria.
- The approach is evaluated in simulation, on recorded data, as well as with live field tests in real traffic.

Regarding specifically the field of elastic bands, this thesis contributes:

- A comprehensive set of weighted objective functions representing driver preferences and restrictions.
- Objective functions which enable to drop the restriction of a fixed goal position. This enables complex dynamic maneuvers despite fixed time intervals between discrete states representing the elastic band.
- An algorithm for efficiently initializing the states of the elastic band based on the observed trajectories.
- An efficient and stable solution for handling dynamic obstacles during optimization.

The remainder of this thesis is divided into five chapters, starting with a discussion on related work and prerequisites in Chapter 2. The proposed swarm-based motion prediction approach is presented in Chapter 3. At the core of this work is a detailed description of the proposed STEBLE trajectory planning in Chapter 4, followed by an evaluation of the experimental results in Chapter 5, and concluding remarks in Chapter 6.

Chapter 2

Related Work and State of the Art

This Chapter summarizes the state of the art for swarm-based planning approaches, trajectory planning for autonomous cars, as well as the specific concept of elastic bands. Furthermore, an overview is given of the autonomous car MadeInGermany, which was used for the experiments presented in this thesis. The associated FUB_ROSCAR software framework for autonomous driving is described.

2.1 Swarm-Based Trajectory Planning

Swarm-based algorithms have a long history in the context of robotics and optimization. The expression of "swarm intelligence" was introduced in 1988 by Gerardo Beni and Jing Wang relating to cellular robotic systems. Beni found the term appropriate because the group of cellular robots he was researching on "had some special characteristics, which in fact are found in swarms of insects, i.e., decentralized control, lack of synchronicity, simple and (quasi) identical members"[1]. Although autonomous cars are certainly not simple robots, they share many of those aspects. In his article Beni also highlights the mechanism of "stigmergy", i.e., "communication by way of the environment", as a critical concept in the modeling of swarms. This indirect coordination between agents or actions through the environment can also be observed on traffic participants. The similarities of swarming behavior and "automated highway systems" were also noted by Gazi and Passino in the introduction section of their work on swarm robotics control [2].

One of the essential works in the field of swarm-based algorithms was presented in 1987 by Craig Reynolds. The model for flocks of boids (bird-like objects) presented in [3] is based on three simple rules for the behavior of each individual boid, often quoted as "Reynolds' rules".

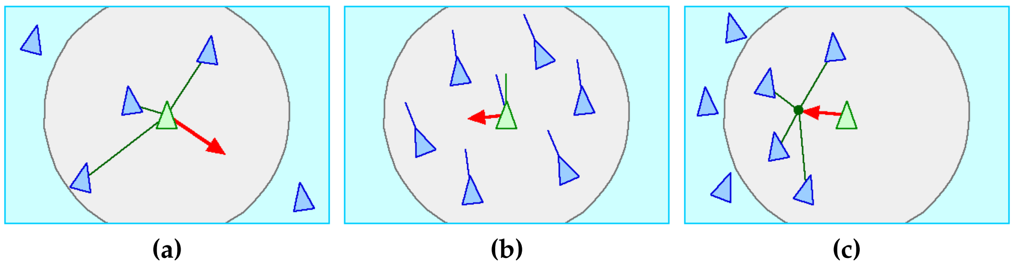


Figure 2.1: Reynolds' rules for modeling flocking behavior. **(a)** Separation: avoid collision with local flockmates. **(b)** Alignment: steer towards the average heading of local flockmates. **(c)** Cohesion: move towards the center of the flock. (Images source: [4])

Reynolds extended his framework in [4], adding different rules for avoiding obstacles or pursuing a path or leader. Also, the concept of velocity alignment is introduced. An early example of the algorithm's ability to bridge the gap between artificial intelligence and natural swarm behavior can be found in [5] where Vaughan et al. use a variation of the boids algorithm to control a robot to interact with a flock of real geese. Based on Reynolds' rules, Olfati-Saber et al. define the flocking algorithm as a set of mathematical equations [6]. The control input for each agent consists of three terms: a gradient-based term, a velocity consensus term, and the navigational feedback. The gradient-based term uses an attractive/repulsive pairwise function to keep the agent at a specified preferred distance from other agents, with the preferred distance smaller than the maximum communication range between agents. The velocity consensus term is a damping force to align the velocities of close flockmates, and the navigational feedback term attracts all agents towards a group objective. Extending those three terms, Semnani et al. present the force-based motion planning (FMP) algorithm for large teams of agents [7]. The approach builds on the concept of "semi flocking", which changes the navigational feedback term in order to enable multiple group targets.

A widely used algorithm with many similarities to Reynolds' boid model is particle swarm optimization (PSO), proposed by Kennedy and Eberhart in 1995 [8]. PSO updates a population (called a swarm) of candidate solutions (called particles) of a given cost function by applying simple rules, which can be interpreted as moving them in the search space. Updating the position of a particle in the search space takes into account the individual particles' current position and velocity, as well as the whole swarm's best-known solution. A major advantage of PSO is that it does not require the optimized problem to be differentiable, as it does not use the gradient of the cost function. On the other hand, PSO does not guarantee that the globally optimal solution is found at all. As an established method for computational optimization, PSO is not exclusively related to the field of trajectory planning. It has been applied to a huge variety of problems [9]. A recent example using PSO to optimize the trajectories of multiple vehicles simultaneously can be found in [10].

Many other optimization algorithms are inspired by the natural swarm behavior of animals, e.g., bees, bacteria, or fireflies. Mavrovouniotis et al. give an extensive overview in their survey [11]. Among the most popular classes of those algorithms is the ant colony optimization (ACO) introduced by Marco Dorigo in 1992 [12]. The algorithm is modeled on ants seeking a path between their colony and a source of food. Information on the currently best solution is stored in the environment in the form of pheromones, a paradigm of stigmergic behavior.

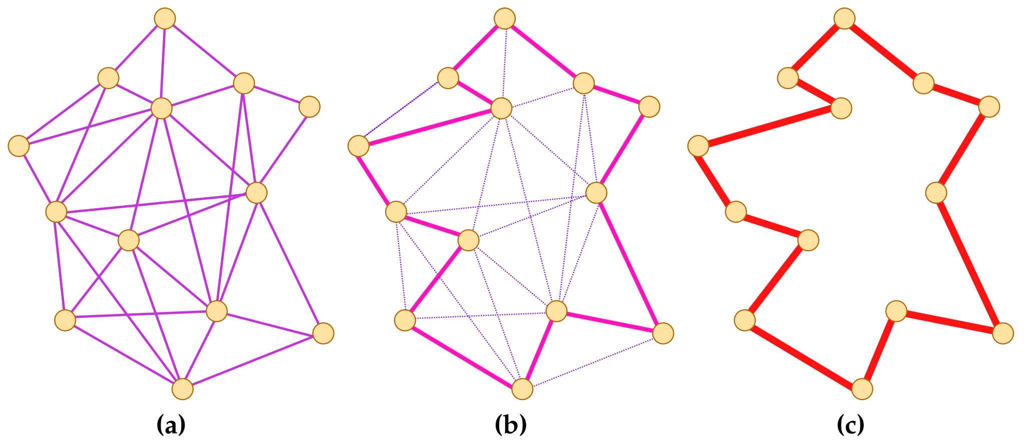


Figure 2.2: The ant colony optimization of the traveling salesman problem. **(a)** Each ant traverses all nodes of the graph. At each node the ants consider distance and the amount of pheromone deposited to probabilistically select the next edge on their tour. **(b)** After having traversed all nodes, the ants deposit pheromones along their traveled path proportional to the quality of the solution. **(c)** Deposited pheromones evaporate over time, leaving only the frequently traveled best paths marked. (Images modified from [14])

While the original ACO algorithm is aiming to solve the traveling salesman problem (compare Figure 2.2), it can be adapted for a large variety of optimization problems. Dorigo et al. give an overview of extensions and applications of ant colony optimization in [13]. Given the origin of the ACO algorithm, i.e., ants finding the most efficient routes, a natural application of the algorithm is urban traffic route planning. Claes et al. highlight two significant advantages of the algorithm for this use case [15]: (1) it can handle dynamically changing cost functions and (2) can find globally coherent solutions based on distributed local information. These features make it a perfect match for the proposed application in an urban traffic environment, where predicted link travel time is constantly changing, and accurate estimation is only locally available. ACO has been applied to path planning for robots in dynamic environments, usually based on a grid network representation. One of the first implementations of this approach is [16]. A more recent example with the focus on autonomous vehicles can be found in [17], where the problem of the discontinuous curvature grid path found with ACO is solved by fitting non-uniform rational B-spline curves (NURBS).

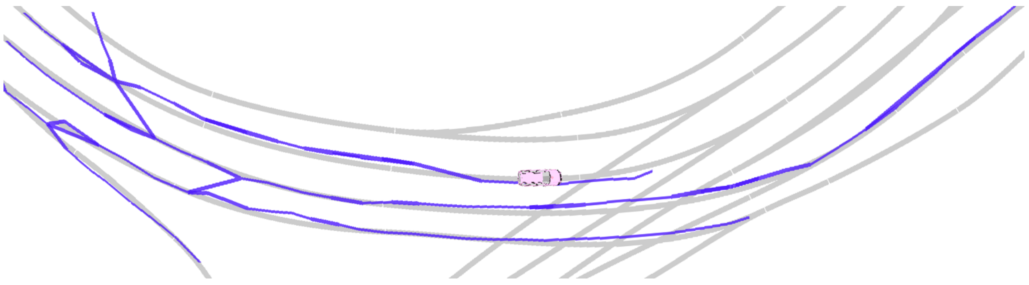


Figure 2.3: Example of a path graph (blue) extracted from observed vehicles trajectories. Gray lines represent expected trajectories from mapped lanes. (Source: [18])

A different approach, also inspired by the pheromone-based stigmergy mechanism of ants, is proposed by Ulbrich et al. to extract a graph of possible paths for an autonomous car from observed vehicles trajectories [18]. Simon Rotter applies the approach to the autonomous driving framework developed at Freie Universität Berlin in his master thesis [19] and evaluates the results on the autonomous car MadeInGermany (which is also the platform used for the experiments throughout this thesis). The complete processing chain from selecting and abstracting sensor data to generating a driveable trajectory is described in detail in [20].

2.2 Elastic Bands For Trajectory Planning

The elastic band theory (as well as Reynolds' boids algorithm) is closely related to the concept of artificial potential fields, introduced by Oussama Khatib for motion planning in 1986 [21]: A point, representing the robot's configuration, is subjected to an artificial force vector. The force vector is calculated from the gradient of a potential function, combining attractive and repulsive potentials, similar to electrostatic potential fields. While the attractive potential guides the robot towards a goal, the repulsive potential repels the robot from obstacles so that the resulting force vector indicates the most promising local direction. The concept of artificial potential fields is a popular approach for implementing obstacle avoidance. It is frequently applied in the domain of robot control, as well as in swarm robotics. In [22] Reif et al. present a study on applying "social" potential fields to distributed autonomous multi-robot control. In the domain of vehicle control, Gerdes and Rossetter present a lane-keeping driver assistance system based on artificial potential fields [23]. They validate their simulated results on a sports car [24]. De Lima et al. implement a dynamic window approach for modifying the field to take moving obstacles into account [25], realizing a path planning system for autonomous road vehicles. Rasekhipour et al. combine the potential fields with a model predictive control approach to include vehicle dynamics [26]. In [27] Boroujeni et al. generate multiple discrete vector fields for different velocities to attract an autonomous car to a target path. The actual force vector for a given pose and velocity is then interpolated between the fields entries.

One of the major advantages of the potential field approach is that its output (i.e., the force vector applied to the robot) can be directly converted into a robot's control signal. It does not need a potentially costly representation of free space, and in most situations, the force vector is guiding the robot safely towards the goal. However, this is not the case in the presence of local minima in the potential field. At a local minimum, the sum of the attractive and repulsive force is zero, resulting in a force vector with magnitude zero (compare Figure 2.4).

The goal of the elastic band approach, introduced by Sean Quinlan and Oussama Khatib in 1993 [29], is to solve this problem by combining the reactive capabilities of the artificial potential field method with global path planning. The authors assume that a coarse collision-free initial path

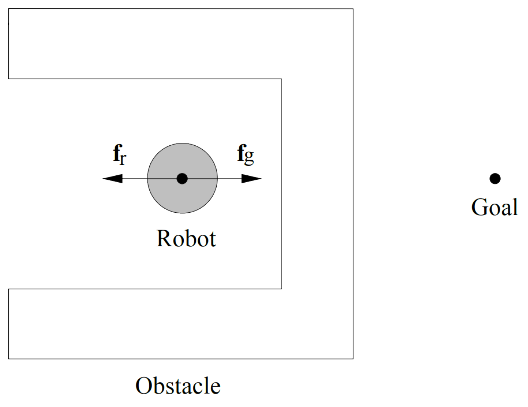


Figure 2.4: A robot trapped in a local minimum of the artificial potential field. The attractive and repulsive forces are balanced, resulting in a force vector with magnitude zero. (Image source: [28])

from the robot's configuration to the goal is provided. Inspired by the tension model in a physical elastic band, this path is then subjected to artificial forces, refining it towards a short and smooth solution while maintaining the initial path's clearance from complex obstacles and global information about how to achieve the goal. Quinlan et al. propose a framework with three levels, each implementing a closed feedback loop. At the topmost and slowest level (with the lowest execution frequency), a world model is used to generate a coarse global solution. This solution is then refined using the elastic band approach on the next level, i.e., the generated path is deformed to handle local changes of the environment in real-time and smooth the path. At the lowest level (with the highest frequency), the control input for the robot to follow the refined path is computed. Quinlan elaborates on the elastic band theory in detail in [28], including the relation to artificial potential fields.

In the original elastic band theory, two artificial forces are used to deform the planned path of the robot: An external repulsion force and an internal contraction force. The repulsion force pushes the path away from obstacles in the vicinity. The contraction force models the tension in a stretched elastic band, smoothing and shortening the path. The total energy of the elastic band is then minimized to find the optimal path. Later Quinlan et al. introduce a third force component [28]. The constraint force ensures that a static configuration of the band exists, but it does not contribute to the band's energy. The elastic band itself is represented as a finite series of points. To ensure that the continuous curve generated from these points is collision-free, Quinlan et al. propose the concept of "bubbles". A bubble is modeling the free space around each point of the band, i.e., an area where the robot can move and is guaranteed not to collide with any obstacles (compare Figure 2.5). In the two-dimensional case, the most simple shape of the bubbles is a circle, but more complex shapes can be used. The size of each bubble is determined by the distance of the corresponding point to the closest obstacles. For the elastic band to be collision-free, the bubbles of subsequent points on the band have to overlap.

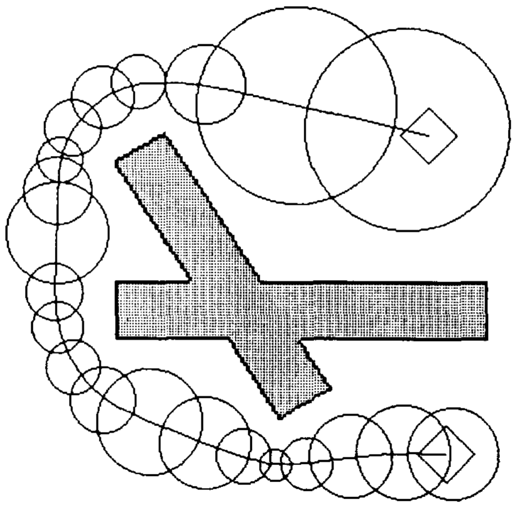


Figure 2.5: Quinlan and Khatib originally describe the elastic band as a series of bubbles, representing the free space around discrete points on the elastic band. (Image source [28])

The external repulsion force on a point of the elastic band is modeled in the same way as the repulsive potential in the artificial potential fields method. It pushes the point away from obstacles, with the force vector corresponding to the gradient of the potential at the respective point. Obstacles can be modeled as arbitrary shapes. The internal contraction force model is inspired by the tension model in a physical elastic band. However, in contrast to the model of an (idealized) linear elastic material, the proposed way to compute the internal force aims to reduce the impact of the stretch of the band. It only depends on the curvature (and not the distance) between two consecutive points. In this way, the distance from the obstacles at which the repulsive force and the contraction force balance each other is much less affected by stretching or contracting the band. Accordingly, local changes in the path due to appearing or disappearing obstacles are less likely to propagate to other regions of the band. The internal force model can be interpreted as a series of springs connecting the points and straightening the band in the absence of obstacles (compare Figure 2.6). By normalizing the force from each spring, a uniform tension along the band regardless of its length (i.e., the number of points) is reflected.

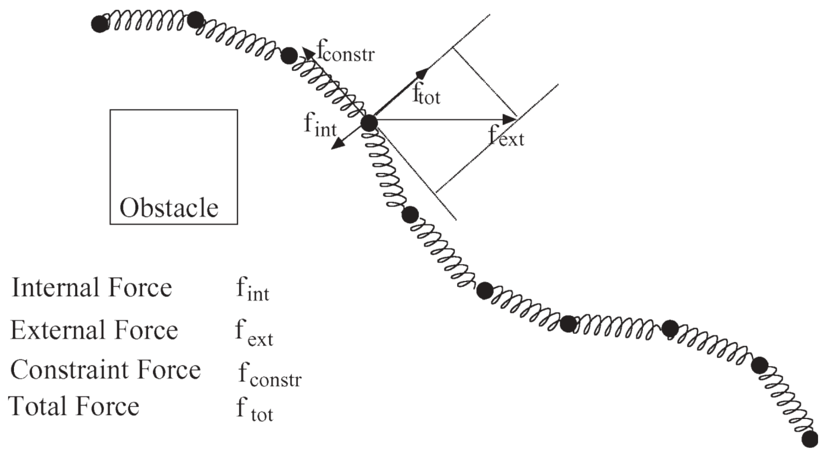


Figure 2.6: The artificial forces deforming the elastic band. The external repulsion force on the elastic band is pushing it away from obstacles. The internal contraction force smooths and shortens the path. It can be interpreted as a series of springs connecting points on the elastic band. The constraint force prevents the band from thinning out at some regions, by restricting motion of points along the band. It is equal and opposite to the external force projected along the elastic band. (Image source [30])

A problem of this model of the contraction force is that it has no component in the tangential direction of the elastic band. Thus the external force could continuously push points along the path, thinning out the band in some regions. To counter this, the constraint force is introduced. It is computed by projecting the external force along the band, preventing the motion of points in this direction. As the stretch of the band can still change due to lateral movement of the points and the internal force model is not entirely independent of the stretch, it is still possible that local changes are reflected in other regions of the band. To further mitigate this, Quinlan et al. add a mechanism to add and remove points to the band, i.e., change the length of the band. This effectively maintains a more or less consistent amount of stretch along the whole elastic band. With this mechanism in place, the constraint force is essential to prevent an infinite sequence of points being inserted, migrate along the band, and being removed at another region.

Quinlan describes an application of the elastic band theory to the motion planning of a system of three Puma 560 manipulator arms [28]. However, the approach has also been applied to the domain of path planning for autonomous vehicles. Hilgert et al. present a method for planning emergency maneuvers of autonomous vehicles [31]. Adopting the original model using only the external and internal forces, they demonstrate the capabilities of the elastic band to refine the planned path of a lane change maneuver dynamically in the presence of other vehicles quickly approaching from behind. The work is extended by Hirsch et al. to include a process called path oriented vehicle state prediction [32]. The process takes into account linear dynamic effects for the calculation of the steering angle. This is achieved by minimizing the distance of a sequence of states (generated from a linear bicycle model) to the optimized elastic band in an iterative process. The approach assumes constant longitudinal velocity of the ego vehicle, and the velocity of obstacles is ignored.

Gehrig et al. present an extension to a vehicle following driver assistance system [30], introducing several modifications to the original elastic bands approach. The system follows the path of a lead vehicle and uses elastic bands to introduce dynamic path modifications when other traffic participants interfere with the leader's path. The observed lead vehicle's path is used as the initial configuration of the elastic band. As the system's goal is to follow this path as closely as possible, the internal forces are offset to penalize any deviation. Furthermore, the initial path cannot be guaranteed to be collision-free, as the lead vehicle may have smaller dimensions or moving obstacles may have interfered. Thus, the concept of bubbles of free space is not valid anymore, and the final equilibrium state of the band is also not necessarily collision-free. To solve this problem Gehrig et al. propose geometrical checks for collision with obstacles on the final path. As a further extension towards the domain of road vehicles, the authors model observed lane markings as virtual obstacles. Modeling the behavior of drivers keeping more distance at higher speed, Gehrig et al. use a potential for obstacles depending on the relative velocities between obstacle and ego vehicle (in addition to the position-based potential proposed by Quinlan). For the position of the obstacles, dynamic parameters are not taken into account (i.e., obstacles are considered static regarding their position). The presented vehicle following system also decouples longitudinal and lateral control, i.e., there is no feedback between the controller adjusting the velocity to follow the lead vehicle and the path planning. Gehrig et al. validate the approach on a real car (a Mercedes Benz E-class 420). The missing non-holonomic constraints on the resulting path are not considered a problem by the authors, with reference to the safety margin around obstacles and the non-holonomic properties of the leader's path. Nonetheless, M. Khatib et al. present an approach to address this problem explicitly, introducing the concept of non-holonomic bubbles to the elastic band theory [33].

Sattel et al. use elastic bands to plan an optimal emergency path for autonomous vehicles [34], implementing a collision avoidance system. They propose two significant extensions of the elastic band approach: First, the (predicted) motion of dynamic obstacles is considered for computing the external force. Furthermore, multiple elastic bands with different initial paths are generated. To realize the inclusion of motion, the points of the elastic band are associated with points in time using a constant planned longitudinal velocity and the approximated traveled distance. Accordingly, the obstacles predicted position at the respective time is used to compute the repulsive force. The prediction of the obstacles also assumes constant velocity and is initialized with sensor data at the beginning of each path planning iteration. The basic idea of planning multiple elastic bands is to generate alternative paths to pass an obstacle, either left or right. This is achieved by shifting points on the initial path in the corresponding direction. Thus the proposed approach optimizes 2^k elastic bands for k observed obstacles. The criterion for evaluating the best path is smallest maximum lateral acceleration. A similar approach, also using motion prediction, is presented by Song et al. [35]. The authors assume constant acceleration and predict the obstacles along the known road geometry.

It has to be noted that all of the approaches mentioned above consider only lateral control, i.e., steering maneuvers. They either assume constant velocity or acceleration of the ego vehicle or do not consider longitudinal control at all in the path planning. This decoupling of path planning and longitudinal control reduces the complexity of the problem significantly, but also makes it impossible to consider dynamic constraints on the robot. This is addressed with the concept of "timed elastic bands", introduced by Rösmann et al. [36]. The timed elastic band method refines trajectories rather than paths. Similar to the methods taking into account obstacle motion, the timed elastic bands approach augments the elastic band with temporal information, i.e., each pair of consecutive points on the band is associated with the duration needed to traverse the segment.

However, those time intervals are also subject to the proposed optimization (along with the spatial configuration of the robot). This enables the approach to consider the temporal aspect of the ego-motion, reflecting dynamic changes in the velocity profile along the path and constraints on the robot, such as maximum velocity and limited acceleration.

Rösmann et al. express the constraints on the timed elastic band in terms of piecewise continuous, differentiable objective functions, reflecting the internal and external forces in the original elastic band theory. The weighted sum of those functions is then minimized using a graph optimization framework. Besides dynamic constraints on the robot (velocity and acceleration) and kinematic constraints enforcing non-holonomic properties, the objectives also implement geometric constraints for clearance of obstacles and attraction to waypoints. The contracting aspect of the elastic band is realized via the "fastest path" objective, minimizing the sum of all time intervals. Since most of the proposed objective functions are used directly or extended in this work, a detailed description can be found in Section 4.2.

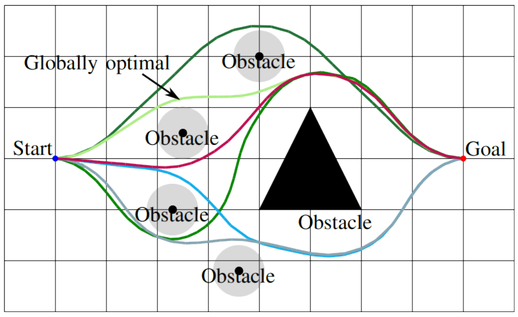


Figure 2.7: Rösmann et al. introduce a method for discovering and maintaining a set of homotopically distinct paths, used to initialize multiple elastic bands passing obstacles on the right and left side. Two paths are homotopic, if one can be deformed into the other without intersecting any obstacles. (Image source [30])

Rösmann et al. extend the approach for planning multiple trajectories simultaneously [37], similar to the work of Sattel et al. described above. The authors introduce a method for discovering and maintaining a set of homotopically distinct initial trajectories (compare Figure 2.7), reducing the computational complexity significantly. Keller et al. augment the approach with objective functions for lateral and longitudinal acceleration as well as jerk [38]. They implement planning emergency maneuvers for autonomous vehicles. In this domain, the distinction between lateral and longitudinal dynamics (and the respective weights of the objective) is of major importance since it allows adjusting the preference between a braking and a lane changing maneuver. Furthermore, Keller et al. rely on an objective function modeling the boundaries of the road, which requires knowledge of the road geometry.

The approach presented in this thesis is aiming to overcome those limitations. It is based on a further adaption of the timed elastic band approach toward the domain of autonomous driving, the stable timed elastic band with loose end (STEBLE). The concept is introduced by Ulbrich et al. in [39]. The approach is described in detail in Chapter 4. An early version of the procedure for initializing the STEBLE trajectories is presented in [40].

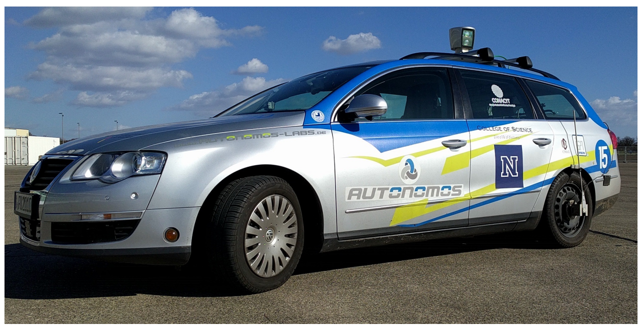
2.3 Hardware and Software Framework

2.3.1 The Automomous Car MadeInGermany

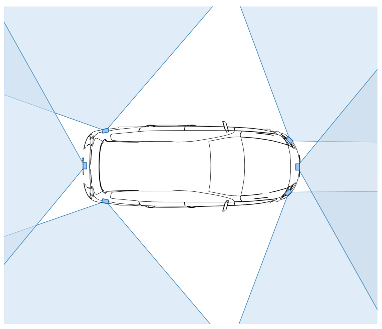
The algorithms proposed in this thesis were tested in live test runs. Furthermore, many of the experiments presented in this thesis are based on data collected in real traffic. The platform used for those experiments and the collection of live data is the autonomous car MadeInGermany (Figure 2.8a), a 2010 Volkswagen Passat Variant 3c.

It is equipped with various sensors and modified so that the drive-by-wire control of gas, brake, and steering can be directly accessed via a CAN bus interface. More detailed information on the sensor setup can be found in [41]. For the experiments throughout this thesis, only the following sensors were used: six Ibeo LUX 4L lidars [42], an Applanix POS LV 510 inertial measurement unit [43], as well as the steering angle, provided on the CAN bus of the vehicle for the integrated Volkswagen lane-keeping system. The Ibeo lidar sensors are fused using an Ibeo electronic control unit (ECU) running the Ibeo Feature Fusion Core Module. The ECU provides the fused point clouds, as well as tracked objects.

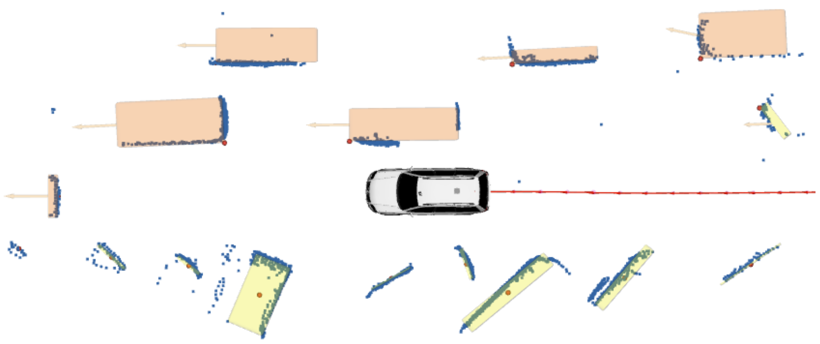
Each of the Ibeo LUX lidar sensors has a horizontal field of view of 110 degrees and a range of up to 200 meters. The six lidars are distributed around the car so that their field of view overlaps (compare Figure 2.8b). Detection and tracking of objects are performed on the fused point cloud



(a)



(b)



(c)

Figure 2.8: The sensor setup of the autonomous car MadeInGermany used for the experiments in this thesis consists of six fused Ibeo LUX 4L lidar sensors and an Applanix POS LV 510 inertial measurement unit. **(a)** MadeInGermany at the test site Flughafen Tempelhof (Image source: Daniel Göhring, 2017). **(b)** The field of view of the lidar sensors. **(c)** The boundary boxes of dynamic (orange) and static (yellow) objects are tracked over time, which mitigates the effect of blindspots and occlusion in the lidar point cloud (blue).

of all sensors, which mitigates the effect of the blindspots in close proximity of the car (Figure 2.8c). The Applanix POS LV 510 inertial measurement unit allows localization with an accuracy in the range of a few centimeters. It has to be noted that the absolute positioning capabilities of the POS LV are not of relevance for the experiments presented in this thesis since no map reference is used. Nonetheless, the Applanix system provides very accurate relative position updates, allowing the evaluation to focus on the validity and convergence of the presented algorithms.

2.3.2 The FUB_ROSCAR Software Framework

The FUB_ROSCAR software framework is the basis for the autonomous driving software stack developed at Freie Universität Berlin. It is built on the robotics middleware ROS [44]. The swarm-based prediction and trajectory planning modules introduced in this thesis are integrated into the FUB_ROSCAR framework via two ROS packages. Besides those two packages, over 50 other packages developed at Freie Universität are involved in the software stack used for the experiments presented in this thesis. (There are many more FUB_ROSCAR packages not used in the context of this thesis.)

The sensors used in the experiments for this thesis, described in detail in Section 2.3.1, are six fused Ibeo LUX 4L lidars [42], an Applanix POS LV 510 inertial measurement unit [43], as well as the steering angle sensor of the integrated Volkswagen lane-keeping system. In the context of this work, the post-processing of the data provided by the respective sensors is limited to decoding and converting the sensor output in ROS messages. The Applanix system provides the odometry of the ego vehicle with six degrees of freedom at 100 Hz, matching the default frequency of the control loop. The steering angle is also provided at 100 Hz. The Ibeo ECU, which is fusing the six individual lidars and performs object detection and tracking, provides object data at a frequency of 12.5 Hz. In addition to the objects' pose and dimensions, each object is associated with a tracking id, enabling the matching of objects between two iterations of the planning loop. The Ibeo object detection also provides a classification into one of the following: car, truck, bike/motorcycle, pedestrian, and unknown.

The data flow in the trajectory planning and control stack of the framework is realized with two closed loops at different frequencies: one planning loop and one control loop (compare Figure 2.9). The planning loop is usually run at 10 Hz, primarily due to the computational cost of the processing (including lidar and camera data). The control loop is run at a much higher frequency of 100 Hz - which is necessary to follow the planned trajectory precisely and avoid oscillations in the steering and velocity control of the vehicle.

There is a direct feedback in each iteration of both the planning and the control loop through the change of the ego vehicle's state and environment detected by the sensors. For the control loop, the feedback data

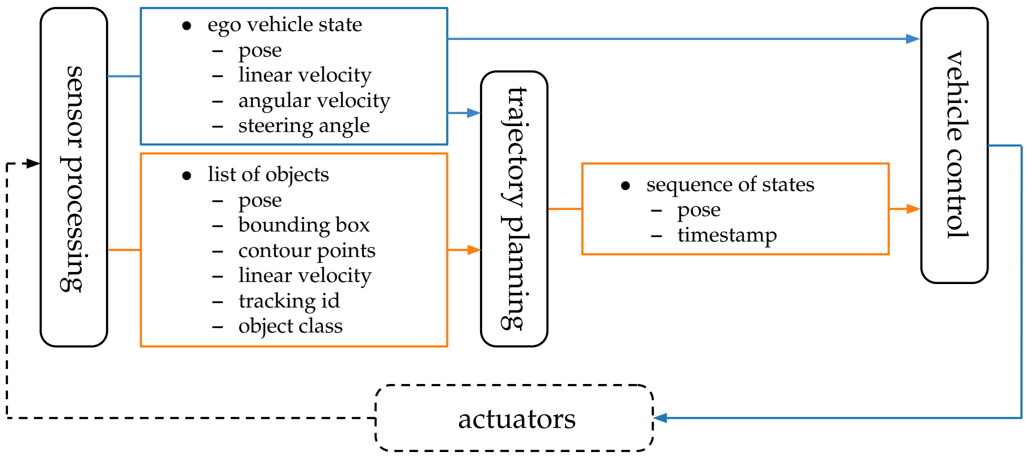


Figure 2.9: Data flow in the trajectory planning and control stack of the FUB_ROSCAR framework. Data in the control loop (blue) is published at much higher frequencies then the data in the planning loop (orange)

includes only the measured ego vehicle's state. For the planning loop also the observed objects are considered. Due to the different frequencies of the loops, the trajectory tracking and the control module are working on the same unchanged planned trajectory for some iterations. This approach can be seen as a hybrid of open and closed-loop control concepts.

Although the existing (map-based) FUB_ROSCAR trajectory planning module was replaced by the proposed swarm-based trajectory planning in the experiments, the control architecture was left as-is: The output of the trajectory planning - a sequence of poses over time - is processed by a trajectory tracking module. This module interprets the sequence of poses as support points of two cubic splines over time. It samples the wanted velocity and steering angle based on a PID controller, considering the actual measured pose and the planned trajectory. The wanted velocity and steering angle are then realized in the control module by two further PID controllers, generating the values for gas and brake pressure, as well as the momentum of the steering wheel. Those values are sent directly to the actuators of the vehicle.

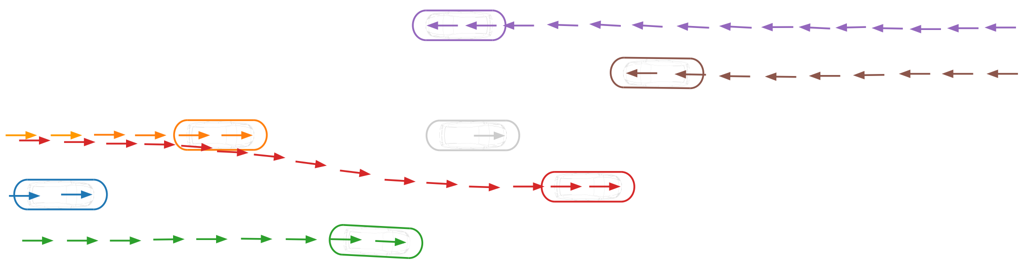
Chapter 3

Swarm-Based Prediction of Dynamic Objects

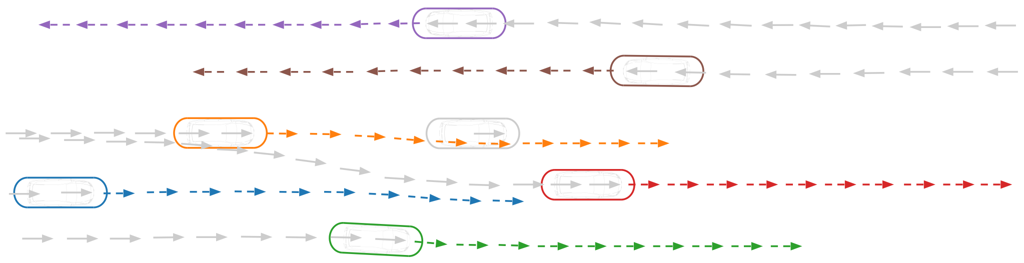
Predicting dynamic objects is one of the essential aspects of trajectory planning since it allows one to anticipate potential collisions and react accordingly in due time. A precise prediction is even more crucial for an approach like STEBLE, described in Chapter 4, where the (partially predicted) trajectories of other vehicles are not only used for obstacle avoidance. In addition, STEBLE also aims towards following them, i.e., the ego vehicle is attracted to the paths of other vehicles. The predicted trajectories also play a significant role in choosing a target to follow and the initialization of the elastic band.

The advantages of the presented swarm-based trajectory prediction over a velocity-based approach are discussed in Section 3.1. In Section 3.2 the tracking of dynamic objects is described, Section 3.3 covers the process of relating the tracked objects to specific roads. The actual prediction of the objects is described in Section 3.4 resulting in a reference trajectory for each object. Section 3.5 deals with re-sampling those trajectories to provide convenient sequences of poses for the algorithms described in Chapter 4.

The input data for the object prediction modules includes the position, orientation, as well as linear and angular velocity of objects surrounding the ego vehicle. The objects are classified as dynamic or static by the perception modules; objects are dynamic when they have been observed moving at some point in time. Each object is identified over time by a unique id. This input data is accumulated for each object $\mathbf{o}_j$ in the tracking step (Section 3.2), resulting in a sequence of observed poses P_j.



(a) For each dynamic object $\mathbf{o}_j$ surrounding the ego vehicle (light gray), the poses and velocities observed by sensor processing modules are tracked over time (solid arrows). The resulting sequences P_j of observed poses are used to attract the ego vehicle to the trajectories of other vehicles (Section 4.2.5). The time interval between stored poses corresponds to the time interval Δt of the STEBLE elastic band (compare Section 4.1).



(b) The objects' poses are predicted for a defined number of discrete time steps Δt (dashed arrows) corresponding to the poses of the STEBLE elastic band (compare Section 4.1). For each object $\mathbf{o}_j$, the predicted poses are appended to the observed poses P_j in a sequence Q_j, which is used for obstacle avoidance (Section 4.2.4), selcting a target to follow (Section 4.3) and generating an initial elastic band (Section 4.4).

Figure 3.1: Output of the tracking and prediction algorithms for dynamic objects.

The poses $P_j = (\mathbf{p}_{j,-n_j}, \dots, \mathbf{p}_{j,0})$ correspond to the poses of the elastic band described in Chapter 4, i.e., each pose $\mathbf{p}_{j,i} = (x_{j,i}, y_{j,i}, \theta_{j,i})$ consists of two coordinates x, y for the two-dimensional position on a plane plus one angle θ for the orientation. The output of the prediction algorithm described in this chapter is a sequence of poses Q_j.

$$Q_j = (\mathbf{p}_{j,-n_j}, \dots, \mathbf{p}_{j,0}, \mathbf{p}_{j,1}, \dots, \mathbf{p}_{j,n+m})$$

This sequence includes the predicted poses of the object $\mathbf{o}_j$, as well as the poses observed in the past, i.e., $P_j \subset Q_j$. Merging the observed and predicted poses has advantages for some of the post-processing algorithms described in Chapter 4. The index i of each pose represents the point in time associated with the pose. The time interval Δt between subsequent poses is constant, i.e., for each pose $\mathbf{p}_{j,i}$ the offset in time to the most recently observed pose $\mathbf{p}_{j,0}$ is given by the product $i\Delta t$.

3.1 Swarm-Based vs. Velocity-Based Prediction

In early iterations of the presented approach, dynamic objects were predicted using the most straightforward method of assuming constant linear and angular velocities [39]. This has the advantage of a very low computational cost: The pose $\mathbf{p}_{j,i}$ of object $\mathbf{o}_j$ at the point in time corresponding to index i of the elastics bands poses can be calculated by simply adding the offset $i\Delta t\vec{v}_j$ to the most recently observed pose $\mathbf{p}_{j,0}$, where $\vec{v}_j = (v_x, v_y, v_\theta)$ is the vector of the linear velocity in x- and y-dimension and the angular velocity. Although this prediction is accurate for straight lanes or in curves with constant curvature, it does not work well for the entrance and exit of curves (compare Figure 3.2). Also, in the planning and construction of roads, segments of constant curvature are generally avoided in favor of a smooth transition between different slopes.

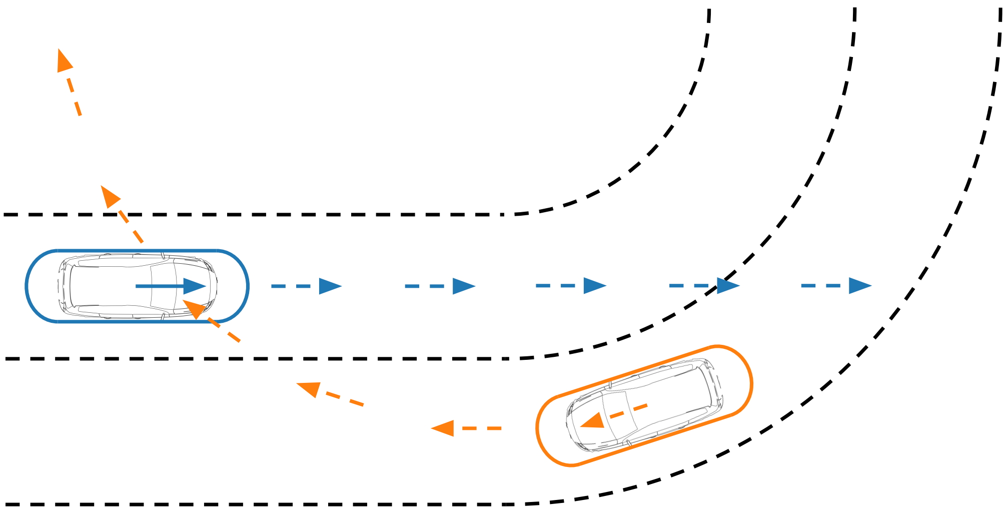


Figure 3.2: Wrongly predicted trajectories assuming constant linear and angular velocity. The error in the prediction is most significant when entering curves (orange vehicle) and leaving curves (blue vehicle).

One advantage of the STEBLE approach is that the exact points in time for which the objects need to be predicted are known in advance. Thus, the objects' predicted trajectories can be computed in advance for those points in time and then be looked up in each iteration of the optimization process. This approach enables a more sophisticated (and computationally expensive) algorithm for calculating the predicted poses without significantly impacting the overall computation cost. The proposed approach assumes that all vehicles follow the same road, i.e., the dynamic object prediction is based on observing all surrounding vehicles instead of assuming constant velocities for individual objects. This principle promotes the idea of swarm-based behavior while also incorporating the structured environment of the traffic infrastructure. Figure 3.3 compares the result of the two approaches for a curvy section of the road. The assumption of constant velocities fails dramatically for all objects in this scenario, but the swarm-based approach predicts the trajectories nearly perfectly (except for the rightmost object, where no observations of vehicles driving ahead are available).

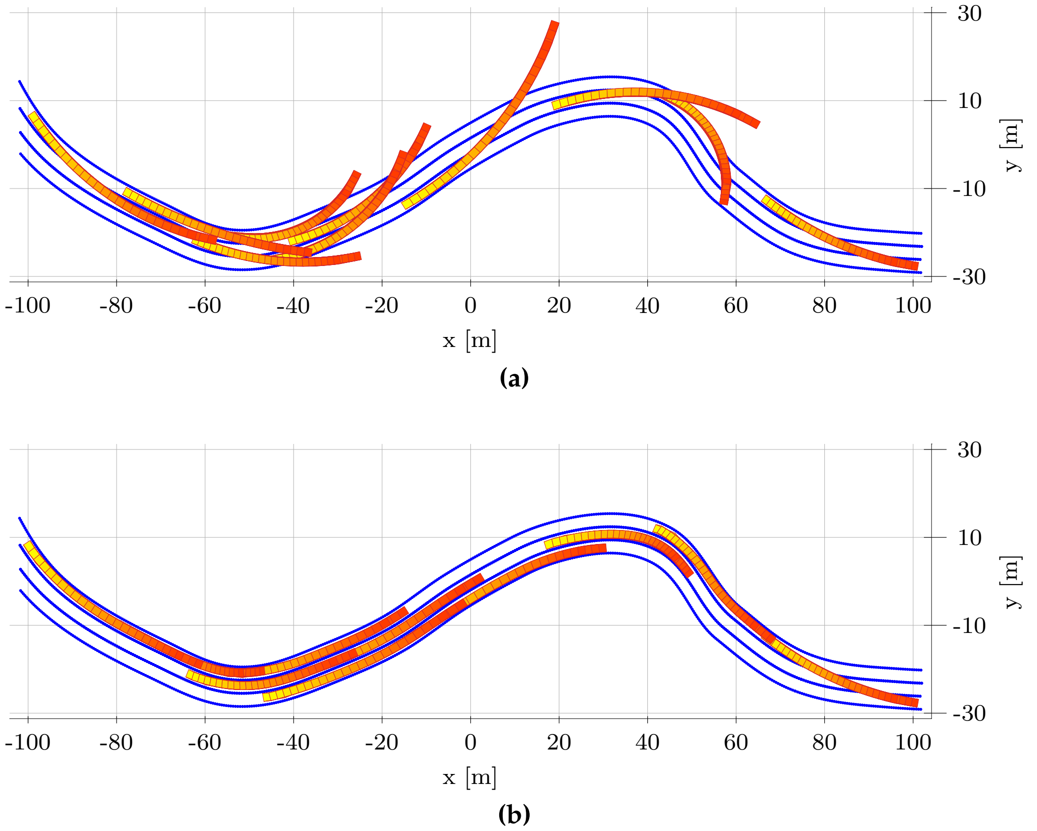


Figure 3.3: Comparison of dynamic object prediction using **(a)** the constant linear and angular velocity and **(b)** the proposed swarm-based approach, taking into account trajectories of other objects driving ahead. The plots show the predicted poses of the objects over time. The objects are predicted for a duration of 5 s with a step size of 0.2 s. The most recently observed pose is shown in yellow, the pose in 5 s is red, the poses in between are colored proportionally. Road boundaries are shown in blue for reference.

This perfect performance is certainly not guaranteed for all scenarios. Although the algorithm presented in the following considers to some degree that drivers do follow other vehicles on shifted paths and with different velocities, the prediction is still prone to individually different behavior. The most important reason for this is that individual human drivers cannot be predicted precisely in general, as too many (unobservable) factors influence their behavior. More specifically, even when restricted by the road networks structure, drivers can still make many choices. Besides opting for different velocity profiles, they can perform lane changes and thus distort the premise of drivers following the lanes. Nonetheless, the evaluation in Section 5.2 verifies that the swarm-based heuristic outperforms the velocity-based prediction substantially in simulation as well as in real traffic scenarios.

3.2 Tracking Dynamic Objects

The perception and tracking of objects are handled in other modules of the FUB_ROSCAR Software Framework, which are not in the scope of this work. In this processing, objects are classified as static or dynamic. The prediction algorithm considers only objects classified as dynamic by the perception modules (i.e., objects observed moving at some point in time). Furthermore, the data passed to the trajectory planning module for each perceived object includes the following information: a unique id assigned by the tracking modules, the center pose, and linear and angular velocities. Also, a label for object classes, such as car, bicycle, or pedestrian, is available.

As a preprocessing step for the swarm-based trajectory prediction described in this chapter, the observed poses are stored for each object $\mathbf{o}_j$ in a queue data structure over time (compare Figure 3.1a). Objects are distinguished by their unique tracking id reported from the sensor processing modules. A user-set parameter limits the length of the queues for performance reasons. The time interval between stored poses corresponds to the constant time interval Δt of the STEBLE elastic band, described in detail in Section 4.1.

The current content of the queues, a sequence of poses P_j for each dynamic object $\mathbf{o}_j$, is the input for the proposed prediction algorithm.

$$P_j = (\mathbf{p}_{j,-n_j}, \ldots, \mathbf{p}_{j,0})$$

The index i of each pose $\mathbf{p}_{j,i}$ represents the point in time associated with the pose. The actual offset in time to the last observed pose $\mathbf{p}_{j,0}$ (which corresponds to the start pose of the trajectory planning described in Chapter 4) is given by $i\Delta t$. As stated above, the number n_j of poses stored for each object $\mathbf{o}_j$ is limited for performance reasons to a respective threshold of 10 s. On the other hand, the number n_j may be smaller than this threshold for objects which have been tracked for a shorter duration.

The observed trajectories P_j are the basis for the swarm-based trajectory prediction described in the following sections. Furthermore, they are used directly by the objective function attracting the ego vehicle to the paths of other vehicles (compare Section 4.2.5).

3.3 Clustering Objects on the Same Road

As stated above, the underlying theory of the proposed approach is that all vehicles follow the same road or, more precisely, drivers generally follow the trajectories of other road users driving ahead. Since the presented approach aims to solve the task of trajectory planning without using a map, the information that the observed vehicles are driving on the same road cannot be derived from such a map. Instead, a very simple heuristic is applied in the first step to divide the observed dynamic objects into two sets of same-directed and oncoming objects by comparing the closest pose of their observed trajectories to the current pose of the ego vehicle (compare Figure 3.4). This heuristic was chosen because, despite its simplicity and low computational cost, it works in the vast majority of scenarios (i.e., when all vehicles going in the same direction as the ego vehicle are actually using the same road). It has to be noted that this heuristic ignores that drivers might be turning into different roads in a similar direction (e.g., at highway exits), or lanes could diverge due to traffic islands. However, such wrongly predicted trajectories do not dramatically impact the proposed swarm-based trajectory planning since it promotes the ego vehicle moving in the same general direction as the majority of close-by vehicles.

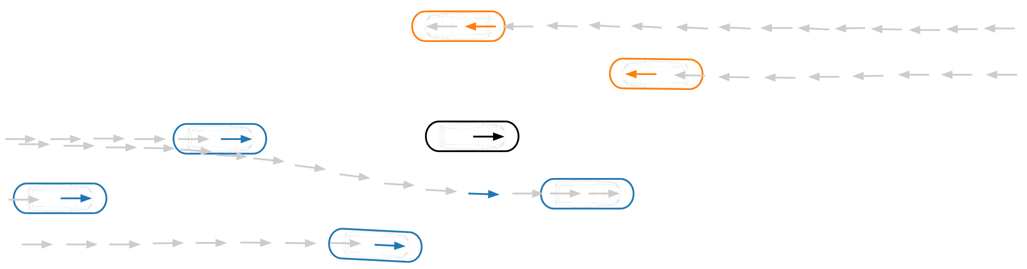


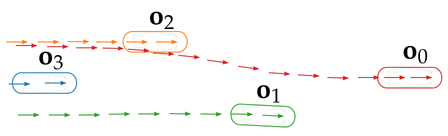
Figure 3.4: The dynamic objects are clustered into two sets for the prediction: One set for same-direction traffic and one set for oncoming traffic. The two sets are processed precisely the same way in the subsequent processing steps, except for an inverted criterion for ordering the elements.

3.4 Determining Reference Trajectories

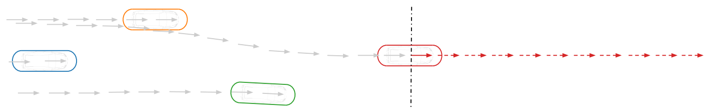
An overview of the subsequent processing step for each set of dynamic objects (same-directed and oncoming) is given in Figure 3.5. The goal of this step is to find the best reference trajectory $Q_{ref,j}$ for each object $\mathbf{o}_j$. This step is intertwined with the next step of sampling the predicted trajectory Q_j from the chosen reference trajectory (described in detail in the following section). The predicted trajectories of previously processed objects are the input for selecting reference trajectories for the remaining ones.

First, the order of processing of the objects within each set is determined. For this, the objects in the two sets are sorted in two respective sequences $O_i = (\mathbf{o}_0, \dots, \mathbf{o}_n)$ by their longitudinal distance to the ego vehicle. Vehicles driving ahead should be processed first, as vehicles driving behind them can use their predicted trajectories. Consequently, the object with the highest longitudinal distance to the ego vehicle is the first element for the same-directed sequence and vice versa for the oncoming sequence.

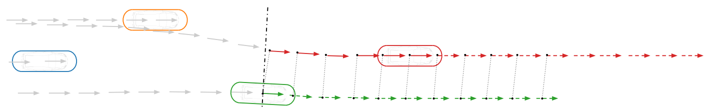
In the following steps, the two sequences O_i of dynamic objects are not distinguished anymore, as they are processed in the same way. For each sorted sequence O_i, the contained objects $\mathbf{o}_j$ are then processed in order of their index. For each $\mathbf{o}_j \in O_i$ a set T_j is maintained, consisting of the predicted trajectories Q_k of previously processed objects (with index $k < j$). As stated above, the objects with lower indices are assumed to drive ahead of objects with higher indices.



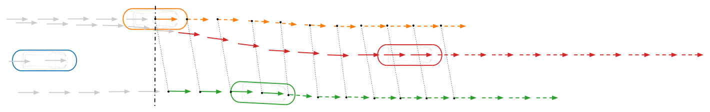
(a) For each set O_i the dynamic objects $\mathbf{o}_j \in O_i$ and corresponding tracked trajectories (solid arrows) are sorted by their longitudinal distance to the ego vehicle from furthest ahead to furthest behind (vice versa for oncoming objects).



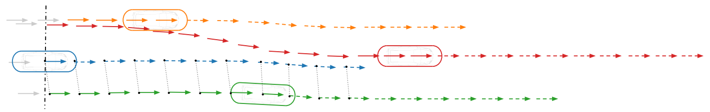
(b) The object $\mathbf{o}_0$ (red), which is the furthest forward, is processed first. No poses of other trajectories are in front of the last tracked pose of $\mathbf{o}_0$. Therefore no reference trajectory is available, the future poses of $\mathbf{o}_0$ (dashed arrows) are predicted from its linear and angular velocity.



(c) The second furthest forward object $\mathbf{o}_1$ (green) is processed next. The only trajectory which has poses in front of the last observed pose of $\mathbf{o}_1$ is the one of $\mathbf{o}_0$ (red). Consequently the valid poses of $\mathbf{o}_0$ are chosen for the reference trajectory $Q_{ref,1}$ (red arrows), regardless of whether they were tracked or predicted in previous steps. The predicted poses of $\mathbf{o}_1$ (dashed green arrows) are sampled from $Q_{ref,1}$



(d) For the next object $\mathbf{o}_2$ (orange) two other (partially predicted) trajectories have valid poses. The valid part of the trajectory of $\mathbf{o}_1$ (green arrows) is selected as the reference trajectory $Q_{ref,2}$, since it has a lower average curvature then the respective valid trajectory of $\mathbf{o}_0$ (red arrows).



(e) The last object $\mathbf{o}_3$ (blue, furthest behind) is processed. The valid trajectory of $\mathbf{o}_1$ (green arrows) is chosen again as reference trajectory $Q_{ref,2}$, since it has the lowest average curvature of all valid trajectories.

Figure 3.5: Selecting reference trajectories from other dynamic objects' poses.

When processing an object $\mathbf{o}_j$, the corresponding previously processed trajectories $Q_k \in T_j$ are checked for validity. A trajectory Q_k is valid if two criteria are true: (a) the distance from the last observed pose $\mathbf{p}_{j,0}$ of $\mathbf{o}_j$ to the closest pose of Q_k does not exceed a threshold d_{max}, and (b) there exists a subsequence $\hat{Q}_k \subseteq Q_k$ of poses which are all in forward direction of $\mathbf{p}_{j,0}$ (where a pose is in forward direction, if the dot product of the orientation vector of $\mathbf{p}_{j,0}$ and the difference vector of their positions is positive). The longest subsequence $\hat{Q}_k$ for each valid trajectory is stored in a set V_j, i.e., after this step, V_j contains all trajectories in a forward direction of $\mathbf{o}_j$.

Next, for all the valid subsequences $\hat{Q}_k \in V_j$ the average curvature $\bar{\kappa}_k$ is calculated by taking the mean of the curvature between all subsequent poses. The equations for estimating the curvature (i.e., the reciprocal of the radius of curvature) between two subsequent poses can be found in the section on the objective function restricting the turn radius (Section 4.2.1). The trajectory $\hat{Q}_k \in V_j$ with the lowest average curvature $\bar{\kappa}_k$ is chosen as reference trajectory $Q_{ref,j}$ for the object $\mathbf{o}_j$. If the set V_j is empty, i.e., there are no valid trajectories, the reference trajectory $Q_{ref,j}$ is also left empty.

The heuristic of selecting the trajectory with the lowest average curvature was chosen above all for its simplicity and the resulting low computational complexity. This is a crucial feature, as the overall number of trajectories to evaluate (with respect to a specific pose) increases quadratically with the number of vehicles. Using the minimal average curvature has two further advantages over other easily evaluated features, such as distance to a reference pose, velocity, or trajectory length and duration. Firstly, a lower curvature, and consequently a lower centripetal acceleration, is generally more comfortable for the passengers. Secondly, the curvature is better suited for detecting lane-change maneuvers. On a straight road, a (partial) lane change increases the average curvature. This is not always the case in curves, but a relatively higher average curvature is still a strong indicator for a lane change. Furthermore, in addition to the average curvature, the distance to the trajectory candidates is considered by limiting valid trajectories to having a minimal distance below a specified threshold.

The output of the processing step described in this section, a reference trajectory $Q_{ref,j}$ for each object $\mathbf{o}_j$, is the basis for generating the corresponding predicted trajectories Q_j. The process of sampling Q_j is described in the following section. As stated above, the two steps of determining the reference trajectory $Q_{ref,j}$ and sampling the predicted trajectory Q_j from it are intertwined: The predicted trajectory Q_j is the input for finding a reference trajectory $Q_{ref,j+1}$ (along with all other previously predicted trajectories $Q_{k<j}$).

3.5 Sampling from the Reference Trajectory

The goal of this processing step is to generate a predicted trajectory for one specific dynamic object $\mathbf{o}_j$. The input for this step are the recently observed poses P_j of $\mathbf{o}_j$ (compare Section 3.2), as well as a reference trajectory $Q_{ref,j}$. The process of determining $Q_{ref,j}$ is described in detail in the previous section. The output is the trajectory Q_j, a sequence of $n + m$ predicted poses appended to the observed trajectory P_j of the corresponding object $\mathbf{o}_j$. The observed poses P_j are included in Q_j (i.e., $P_j \subset Q_j$), since, on the one hand, the reference trajectories are generated from both observed and predicted poses (compare preceding section), and, on the other hand, the objective function for obstacle avoidance needs to access past states in some cases to cover the dynamic longitudinal safety distance (compare Section 4.2.4). The handling of obstacle avoidance is also the reason for extending the number of predicted poses beyond the maximum length n of the elastic band: The number m of surplus poses is a configurable parameter likewise related to the dynamic longitudinal safety distance.

The proposed algorithm for sampling the predicted poses of Q_j from a reference trajectory $Q_{ref,j}$ takes into account two important facts: Firstly, in many cases, drivers do not follow other drivers' trajectories strictly. They may be using different lanes, so their path is shifted by a spatial offset. In curves, also the rotation has to be considered, shortening inner paths and extending outer paths, respectively. Secondly, drivers may follow the road with varying velocities, depending on observed obstacles and driver preferences. The proposed approach takes this into account by shifting the paths and velocity of the predicted trajectories in relation to the recently observed spatial offset $\vec{d}$ and difference in speed Δv.

Figure 3.6 gives an overview of the full process of sampling the predicted poses of $\mathbf{o}_j$ from a reference trajectory $Q_{ref,j}$. First, the offset vector $\vec{d}$ is calculated, pointing from the first pose $\mathbf{p}_0$ of the reference trajectory $Q_{ref,j}$ to the last observed pose of $\mathbf{o}_j$ (i.e. the last element of P_j, compare Figure 3.6b). Also, the difference Δv in the speed (i.e. the magnitude of the velocity in forward direction) at those two poses is stored. For this, the respective speed values are approximated from the distance and the constant time interval Δt between subsequent poses (which is valid for P_j as well as $Q_{ref,j}$).

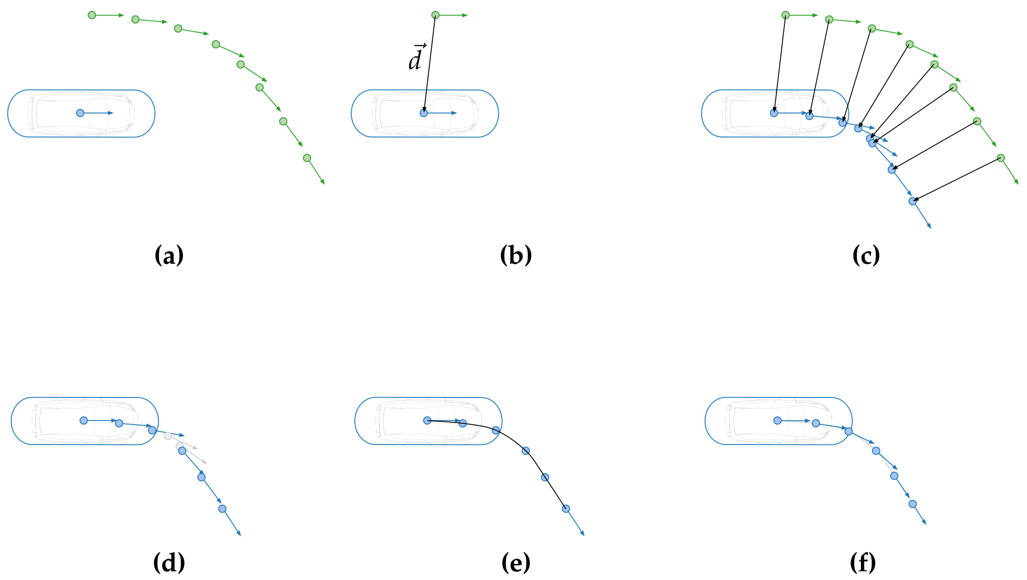


Figure 3.6: Deriving predicted poses from a reference trajectory. **(a)** The blue object's trajectory is to be predicted from the poses of the reference trajectory (green). **(b)** An offset vector $\vec{d}$ is calculated between the first reference pose (green) and the last observed pose of the blue object. Also, the difference Δv in the speed at those two poses is stored. **(c)** The offset vector $\vec{d}$ is attached to each reference pose (green) taking into account the rotation. The resulting sequence of poses (blue circles) is annotated with a sequence of speed values in the respective reference poses. **(d)** The pose sequence is filtered so that the distance between subsequent poses does not fall below a minimum (compare Figure 3.7). **(e)** Using the remaining poses as support points, two cubic splines are constructed for the x- and y-coordinate over time. The respective time intervals between two poses are calculated from the annotated speed values taking into account the initial difference in speed Δv. **(f)** The resulting splines and their first derivatives are then sampled with the desired temporal interval to calculate the predicted poses of the blue object.

The offset vector $\vec{d} = (x_d, y_d)$ is then attached to each pose $\mathbf{p}_i$ of the reference trajectory to obtain a sequence of poses $\hat{\mathbf{p}}_i$. Here, the relative orientation of the reference poses is taken into account, i.e., $\vec{d}$ is rotated by the respective difference $\Delta\theta_i$ in the orientation of p_i to the first pose p_0 of $Q_{ref,j}$ (compare Figure 3.6c).

$$\mathbf{p}_i = (x_i, y_i, \theta_i), \; \Delta\theta_i = \theta_i - \theta_0$$

$$\hat{\mathbf{p}}_i = (x_i + \cos(\Delta\theta_i)x_d - \sin(\Delta\theta_i)x_d, y_i + \sin(\Delta\theta_i)y_d - \cos(\Delta\theta_i)y_d, \theta_i)$$

The resulting sequence of poses $S = (\hat{\mathbf{p}}_0, \ldots, \hat{\mathbf{p}}_n)$ is annotated with a sequence $V = (v_0, \ldots, v_n)$, reflecting the speed in the respective reference poses $\mathbf{p}_i$. Again, the respective speed values are approximated from the distance and the constant time interval Δt between subsequent poses. The pose sequence S is filtered, considering two criteria described in Figure 3.7. The resulting sequence $\hat{S}$ is guaranteed to have a minimum separation of d_{min} between subsequent poses.

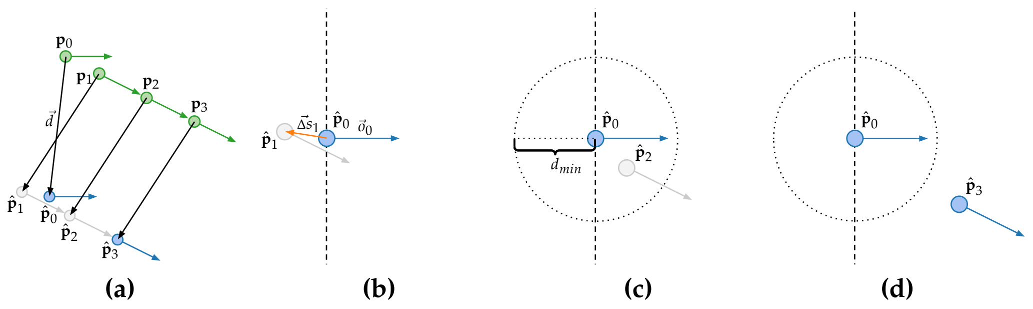


Figure 3.7: Example of filtering a sequence of poses for spline construction. **(a)** The offset vector $\vec{d}$ is attached to each reference pose $\mathbf{p}_i$ taking into account the rotation. The resulting poses $\hat{\mathbf{p}}_i$ are checked in sequence for two criteria. In the example shown, only two poses (blue) are selected as support points for the trajectory spline. The two other poses (gray) are rejected for not satisfying both criteria. **(b)** The pose $\hat{\mathbf{p}}_0$ is selected because it is the first pose in the sequence. The pose $\hat{\mathbf{p}}_1$ is rejected because it is not in forward direction the last selected pose $\hat{\mathbf{p}}_0$. This is determined by evaluating the dot product of the orientation vector $\vec{o}_0$ of $\mathbf{p}_0$ and the difference vector $\vec{\Delta s}_1$ (orange) pointing from $\hat{\mathbf{p}}_1$ to $\hat{\mathbf{p}}_0$. **(c)** The pose $\hat{\mathbf{p}}_2$ is rejected because it is too close to the last selected pose $\hat{\mathbf{p}}_0$. This criterion is evaluated by comparing the euclidean distance between the two poses to a distance threshold d_{min}. **(d)** The pose $\hat{\mathbf{p}}_2$ is selected because it fulfills both criteria: it is in forward direction of the last selected pose $\hat{\mathbf{p}}_0$ as well as further away then the minimum distance d_{min}.

Using the remaining poses $\hat{\mathbf{p}}_i \in \hat{S}$ as support points, two cubic splines with continuous first and second derivatives are constructed for the x- and y-coordinate over time (compare Figure 3.6e). The respective time intervals Δt_i between two poses $\hat{\mathbf{p}}_i$ and $\hat{\mathbf{p}}_{i+1}$ are calculated from the respective annotated speed values $V = (v_0, \ldots, v_n)$ and the arc distance b_i between the two poses. For the definition of the arc distance b_i compare Section 4.2.3. Furthermore, the speed difference Δv in the last observed pose $\mathbf{p}_{j,0}$ of $\mathbf{o}_j$ and the first pose $\mathbf{p}_0$ of the reference trajectory $Q_{ref,j}$ is taken into account by subtracting it from the respective speed values.

$$\Delta t_i = \frac{b_i}{v_i - \Delta v}$$

For the construction of the spline the ecl_geometry ROS package [45] is used, which in turn references [46] for the implementation of the continuous derivative cubic spline. The resulting splines and their first derivatives are then sampled with the desired temporal interval to calculate the predicted poses $(\mathbf{p}_{j,1}, \ldots, \mathbf{p}_{j,n+m})$ of the object $\mathbf{o}_j$. It has to be noted, that the time domain of the spline defined by the support points may be not large enough to sample all required $n+m$ poses. This is the case when no valid reference trajectory is available, but it can also happen when the last observed speed of $\mathbf{o}_j$ is substantially higher than the corresponding speed on the reference trajectory. In those cases the missing elements are calculated assuming constant linear and angular velocity.

Finally, the predicted poses $(\mathbf{p}_{j,1}, \ldots, \mathbf{p}_{j,n+m})$ are appended to the sequence of observed poses $P_j = (\mathbf{p}_{j,-n_j}, \ldots, \mathbf{p}_{j,0})$. Corresponding to the construction of P_j (compare Section 3.2), the index i of each pose $\mathbf{p}_{j,i}$ represents the point in time associated with the pose. The temporal offset to the most recent observation is given by the product $i\Delta t$.

$$Q_j = (\mathbf{p}_{j,-n_j}, \ldots, \mathbf{p}_{j,0}, \mathbf{p}_{j,1}, \ldots, \mathbf{p}_{j,n+m})$$

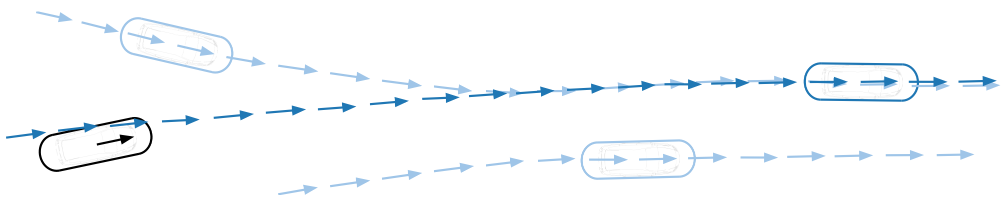
The resulting sequence Q_j, i.e., the predicted trajectory of an object $\mathbf{o}_j$ attached to its actually observed poses, is the output of the processing step described in this section. As stated above, this sequence is then used to generate reference trajectory candidates for subsequently processed objects $\mathbf{o}_{k>j}$ (compare Section 3.4). Additionally, the predicted trajectories are of major relevance for the objective functions to avoid obstacles (Section 4.2.4) and following other vehicles (Section 4.2.5), as well as for selecting the best target to follow (Section 4.3) and the generation of the initial elastic band (Section 4.4).

Chapter 4

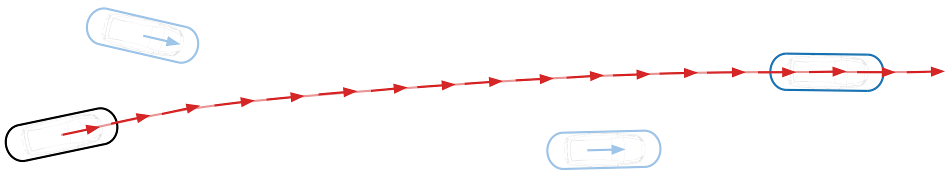
Swarm-Based Trajectory Planning (STEBLE)

The approach presented in this chapter enables efficient trajectory planning without using a map, i.e., when no map is available, localization in a map is not possible, or the observed behavior of road users differs from the given map. To this end, in the following sections, the STEBLE approach for trajectory planning is presented. It transfers many aspects of swarm behavior, like flocking or trail pheromones, to the domain of autonomous driving. Figure 4.1 gives an overview of the proposed process of generating a trajectory.

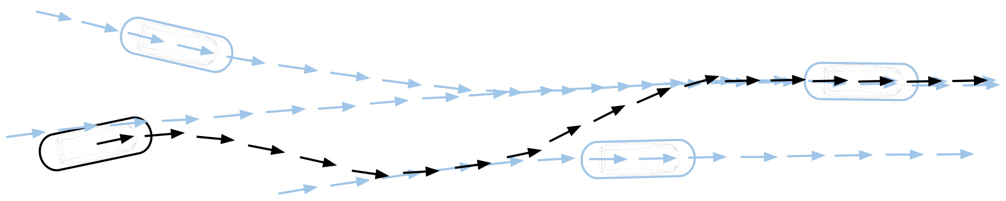
STEBLE cannot work without observing at least one other vehicle. The direction of the generated trajectory is solely determined by selecting a target vehicle to follow, similar to many flocks in nature choosing a (temporary) leader who determines the direction of the swarm. Many other implementation details can be related to characteristics observed on animals moving in swarms or collectives. The set of presented objective functions attract the ego vehicle to other traffic participants (or "swarm members") while simultaneously keeping a safe distance to them. The same behavior can be seen on flocking birds or schooling fish. The objective function for attracting the trajectory to the trajectory of other vehicles is a central part of STEBLE and resembles the trail pheromones many species of ants use to guide their fellow members. The prediction of dynamic objects incorporates aspects of both flocking and trail pheromones, as it assumes that drivers follow in the shifted tracks of other vehicles.



(a) A target vehicle is chosen from a set of candidates (blue). The ego vehicle (black) is supposed to follow the target's trajectory.



(b) A sequence of initial poses (red) is sampled from a cubic spline based on the target vehicle's trajectory. A constant time interval is assumed between subsequent poses.



(c) The planned trajectory of the ego vehicle (black) is optimized using the elastic band approach. The optimization is based on several objective functions, taking into account the ego vehicle's dynamics, avoiding obstacles and attracting the band to observed vehicles' trajectories.

Figure 4.1: Overview of the proposed trajectory planning.

STEBLE extends the established trajectory planning method of elastic bands. Section 4.1 presents the principal aspects of the approach, as well as the representation as a graph optimization problem. The corresponding objective functions, presented in Section 4.2, are the basis for the optimization of the trajectory. Section 4.3 presents a heuristic to determine the optimal target vehicle to follow. In Section 4.4 a procedure to initialize the elastic band is proposed. Section 4.5 presents an approach to prune invalid states from the elastic band, to prevent the optimization from being stuck in a local minimum.

4.1 STEBLE - Elastic Bands with Fixed Time Interval and Flexible Goal

The method described in this chapter utilizes the concept of timed elastic bands to generate locally optimal trajectories. The actual timed elastic band is composed of a sequence of three-dimensional poses and an associated sequence of time intervals between those poses. All constraints, e.g., minimum distance to obstacles, maximum velocity, or acceleration, are formulated as objective functions. The optimization step then adjusts all poses and time intervals to minimize the weighted sum of these functions.

An essential aspect of timed elastic bands is that they consider temporal aspects of the whole trajectory, i.e., they can optimize dynamic constraints, such as acceleration, for the whole trajectory in addition to avoiding obstacles. The poses at the end of the trajectory restrict the poses at the beginning. In this way, it is possible to smoothly plan for more extended maneuvers such as overtaking and lane changes. Furthermore, elastic bands enable planning in continuous space and are still efficient compared to searching a huge discrete state space.

The concept of elastic bands for trajectory optimization is also ideally suited for the task of following other vehicles' trajectories, as an objective function that attracts the band to those trajectories can be easily formulated. This following of trajectories can be seen as swarm behavior, such as ants following the pheromone tracks of their fellow species.

However, existing trajectory planning solutions using elastic bands require a fixed start and goal position between which the band is stretched. This, in turn, requires dynamic time intervals between the individual poses; otherwise, the average velocity on the planned trajectory would also be fixed. As shown in the following sections, using fixed time intervals between the poses of the band dramatically reduces the complexity of the optimization problem. A critical contribution of this thesis is the concept of stable timed elastic bands with loose ends (STEBLE), an approach eliminating the need for a fixed goal position while still preserving the requirement to follow other vehicles' trajectories.

Nevertheless, the concept of elastic bands has two inherent drawbacks: 1) the optimization step can only find a local optimum and 2) it is not possible to formulate hard constraints with the objective functions, i.e., the optimal solution can be a sequence of poses that violates some of the constraints. For example, the band could be trapped between two obstacles, which results in the respective objective function pushing the band in opposite directions and thus canceling the influence of both obstacles altogether. The STEBLE approach presented in this thesis provides strategies to solve this problem by pruning invalid states from the elastic band. Another vital factor in mitigating this effect is providing a reasonable estimate for the initial state of the band, which is another notable contribution of this thesis.

4.1.1 Elastic Bands for Trajectory Planning

The proposed trajectory planning with stable timed elastic bands with loose ends (STEBLE) primarily builds on the concept of timed elastic bands (TEB) introduced by Rösmann et al. [36]. The TEB framework was implemented and published by Rösmann as the central part of the ROS package teb_local_planner [47]. TEB extends the original elastic band approach proposed by Quinlan et al. [29] with regard to temporal aspects. Compare Section 2.2 for related work on the elastic band theory.

This original elastic band is described as a sequence Q of poses $\mathbf{p}_i$ in a three-dimensional workspace with two coordinates x_i, y_i for the position on a plane plus one angle θ_i for the orientation.

$$
\begin{aligned}
Q &= (\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_n) \\
\mathbf{p}_i &= (x_i, y_i, \theta_i)
\end{aligned}
$$

Timed elastic bands (TEB) augment Q by an additional sequence of time intervals, which denote the time needed by a robot to transit between adjacent poses. As discussed in detail in the next section, the approach presented in this thesis (STEBLE) omits this sequence of time intervals and instead defines one global time interval Δt. This Δt is valid for all pairs of adjacent poses, i.e., denotes the time needed by the ego vehicle to transit from each pose $\mathbf{p}_i$ to the next pose $\mathbf{p}_{i+1}$. That is, for STEBLE, the elastic band is described entirely by the sequence of poses Q and a constant time interval Δt.

$$B = (Q, \Delta t)$$

For timed elastic bands (i.e., TEB and STEBLE), the association of time intervals (or a single time interval, respectively) with the poses allows formulating objective functions regarding dynamic aspects, such as velocity, acceleration, or moving obstacles.

The timed elastic band B is optimized by minimizing a global weighted multi-objective function $f(B)$.

$$f(B) = \sum_k \gamma_k f_k(B)$$
$$B^* = \underset{B}{\operatorname{argmin}} f(B)$$

$f(B)$ calculates the weighted sum of several objective functions f_k with corresponding weights γ_k. The individual components f_k are described in detail in Section 4.2. The resulting timed elastic band B^* is optimal in terms of the poses $\mathbf{p}_i$ with regard to $f(B)$.

4.1.2 Using a Fixed Time Interval and Flexible Goal

The central concept of STEBLE - stable timed elastic bands with loose ends - is that the time interval Δt associated with the transition from pose $\mathbf{p}_i$ to pose $\mathbf{p}_{i+1}$ is constant. Thus, there is no need for nodes representing the time intervals between poses in the hyper-graph to optimize. This concept results in a much smaller search space for locally optimal solutions compared to TEB since it is restricted to spatial configurations.

Although this can be seen as going a step back to the original elastic band approach, all the objective functions with respect to Δt are still available in STEBLE, so that the expression "timed" elastic band is still justified. Dynamic changes of the velocity throughout the band are still possible by varying the euclidean distance between poses. Besides the reduced complexity of the optimization problem, it is generally a preferable feature to have constant time intervals on the planned path, as many calculations can be simplified (e.g., objects' predicted poses can be computed in advance of the optimization for discrete points in time).

The other vital aspect of STEBLE is that the last pose of the band (i.e., the goal) is not set to a fixed position and orientation during the optimization process. Since the total number of poses is constant after initialization, the length of the band regarding the temporal aspect is also fixed. Thus, a spatially fixed goal pose would imply the unnecessary constraint of being at a specific time at a specific position. Instead, to follow other cars, it is sufficient to be somewhere on their trajectory at a given time. STEBLE uses the objective function f_{path} to constrain the band's poses to other vehicles' trajectories to achieve such a behavior (compare Section 4.2). In [39] STEBLE and TEB are compared based on their performance when planning a simple merge into traffic maneuver. The respective experiments have shown that f_{path} provides enough stability to the planned paths so that it is feasible to loosen the end of the elastic band.

All objective functions and the weights and parameters used for STEBLE are described in detail in the following subsections.

4.1.3 Trajectory Optimization as Graph Optimization Problem

This optimization problem can be represented as a hyper-graph (compare Figure 4.2). This generalization of the problem can be solved using established methods for graph optimization. For the implementation in the context of this thesis, the g2o library [48] was used. g2o is an open-source C++ framework for optimizing graph-based nonlinear error functions. The algorithm to solve the underlying nonlinear least-squares problem can be configured from a range of standard methods. For the experiments in this thesis, the Levenberg-Marquardt solver provided by the g2o framework was selected.

The ego vehicle's poses $\mathbf{p}_i$ over time are represented as nodes. The individual objective functions f_k correspond to hyper-edges of the graph. Hyper-edges can relate to any number of nodes. In the presented implementation, there are unary, binary, and ternary edges associated with one, two, and three nodes, respectively. An example of a unary edge

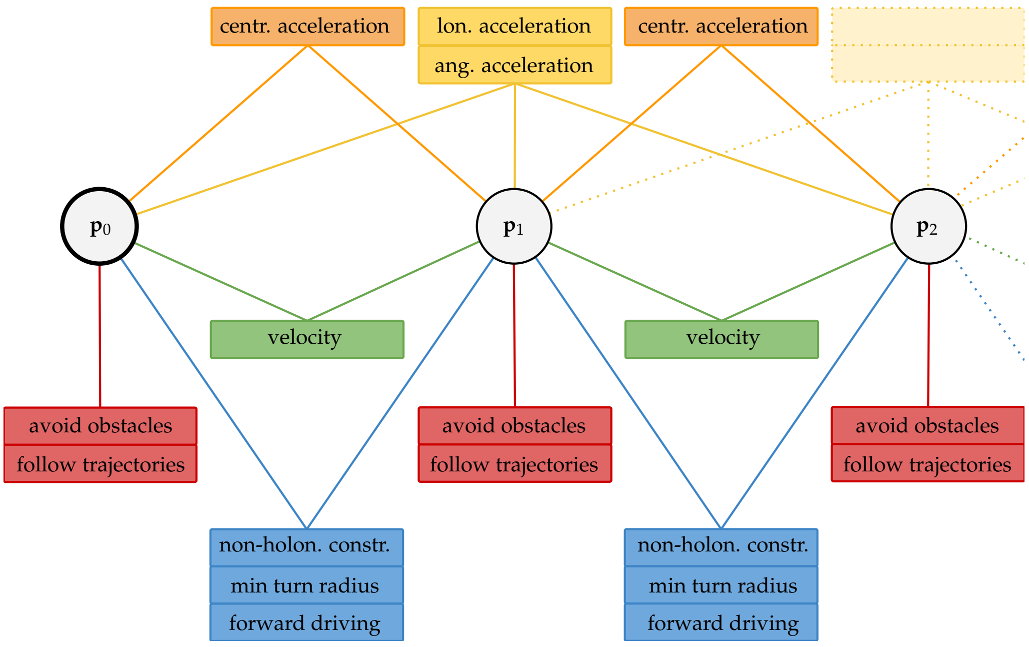


Figure 4.2: Representation of the timed elastic band optimization problem as hyper-graph. The ego vehicle's poses $\mathbf{p}_i$ over time are represented as nodes (circles). Hyper-edges (annotated with squares) can relate to any number of nodes. In the context of timed elastic bands, there are unary (red), binary (blue, green, orange), and ternary (yellow) edges, corresponding to the respective objective functions f_k. Only the node corresponding to the first pose $\mathbf{p}_0$ of the trajectory has a fixed value. The values of all other nodes, i.e., the poses $\mathbf{p}_{i>0}$, can be modified during the optimization process to minimize the overall cost for all edges.

is the objective function for obstacle avoidance. The edges for limiting velocity are binary, as two nodes (poses) are needed for the calculation. Edges related to acceleration are mostly ternary. A detailed description of the respective objective functions associated with the hyper-edges can be found in Section 4.2.

Furthermore, the first pose is permanently fixed during the optimization process, i.e., it cannot be changed. This makes sense since we generally want to start the trajectory from a specific pose. While in TEB also the last pose is fixed, this property is disbanded for the STEBLE approach (compare Section 4.1). All nodes (poses) except the first can be modified during the optimization process.

4.2 The Objective Functions

The final timed elastic band is optimized in terms of spatial configurations (poses). At the core of this optimization process is the weighted multi-objective function $f(B)$. It calculates the weighted sum of several objective functions f_k with corresponding weights γ_k.

$$f(B) = \sum_k \gamma_k f_k(B) \tag{4.1}$$

Each function f_k represents an objective or a constraint that is associated with one or more consecutive poses $\mathbf{p}_i$. Rösmann et. al. introduced in [36] a piece-wise continuous, differentiable cost function e_Γ (Eq. 4.2) that rewards objectives and penalizes the violation of a constrained value x beyond a threshold x_r, respectively.

$$e_\Gamma(x, x_r, \epsilon, S, n) \simeq \begin{cases} (\frac{x-(x_r-\epsilon)}{S})^n & \text{if } x > x_r - \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

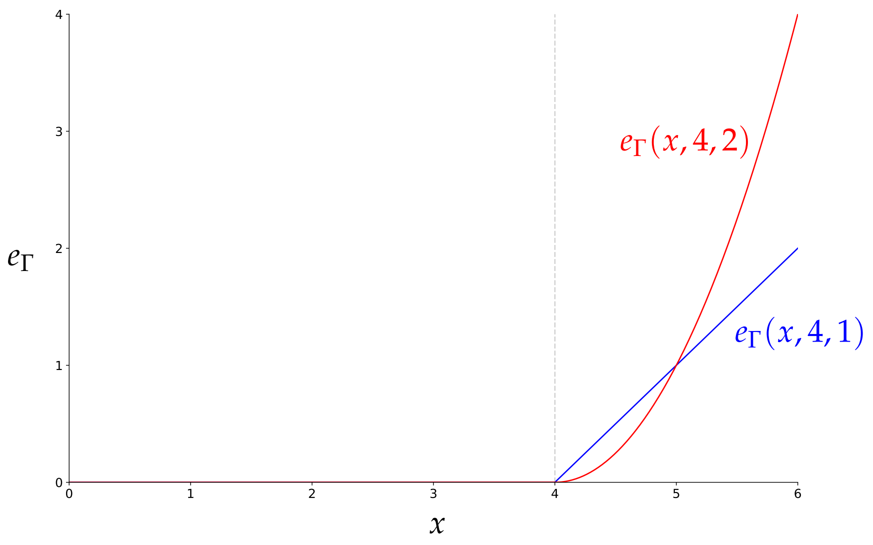


Figure 4.3: The piece-wise continuous, differentiable cost function e_Γ (Eq. 4.3).

In the context of this work the simplified version (Eq. 4.3) of e_Γ was used for all objectives, setting the scaling parameter $S = 1$ and $\epsilon = 0$. For all objectives, the polynomial order n was set to either one or two.

$$e_\Gamma(x, x_r, n) \simeq \begin{cases} (x - x_r)^n & \text{if } x > x_r \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

Based on e_Γ, Rösmann et al. proposed objective functions restricting maximum velocity, maximum longitudinal and angular acceleration, and enforcing non-holonomic kinematics [36], which are used in STEBLE. Furthermore, existing functions from the implementation in [47] are used to penalize backward-driving, falling below the minimum turning radius, and maintaining a specific velocity at the start and goal pose.

Objective Function		Description
f_{nhk}	non-holonomic kinetics	These functions ensure that all poses on the band follow the principle of non-holonomic kinetics.
f_{turn}	min turning radius	
$f_{forward}$	forward driving	Enforces a "only forward driving" policy for the whole trajectory.
f_{v_max}	max velocity	Limits the maximum velocity on the band to the specified threshold.
f_{v_opt}	optimal velocity	Penalizes any deviation from the specified optimal velocity. Used to modify the follow distance.
f_{a_lon}	max longitudinal acceleration	These functions restrict the maximum acceleration on the band to the specified respective thresholds.
f_{a_cen}	max centripetal acceleration	
f_{a_ang}	max angular acceleration	
f_{ob}	avoid obstacles	Repels the band from obstacles closer than the specified threshold.
f_{path}	follow trajectories	Attracts the band to the trajectories of other vehicles.

Table 4.1: Overview of the individual objective functions f_k contributing to the global weighted multi-objective function $f(B)$

The most important addition for the STEBLE approach is an objective function to attract the trajectory to other vehicles' trajectories. Furthermore, an objective function regarding the centripetal acceleration was added, and the existing function for handling dynamic obstacles was substantially modified. Also, an objective function is proposed to prefer a specific velocity (instead of limiting the maximum).

An overview of all objective functions used for STEBLE is given in Table 4.1. They are described in detail in the remainder of this section; the respective weights and thresholds are discussed in Section 5.3.1.

4.2.1 Non-Holonomic Kinematics and Turning Radius

The most crucial objective for optimizing the trajectory is that the vehicle must be able to follow it. The related objective functions f_{nhk} and f_{turn} ensure that all poses on the band follow the principle of non-holonomic kinetics and the curvature of the trajectory does not exceed the physical limitations of the vehicle, respectively.

The basic idea of the related objective functions f_{nhk} and f_{turn} is to approximate a suitable path by assuming arc segments between two consecutive poses, i.e., each pair of adjacent poses $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ is required to be on a common arc segment of constant curvature and radius. It has to be noted that this assumption can only partially enforce driveable trajectories, as the transition between arc segments with different curvatures would require an instantaneous change in the steering angle, i.e., an infinitely high angular acceleration. This is taken care of by the objective function f_{acc_theta} discussed in the respective section below, which implicitly constrains that the change in curvature from one arc segment to the next is limited.

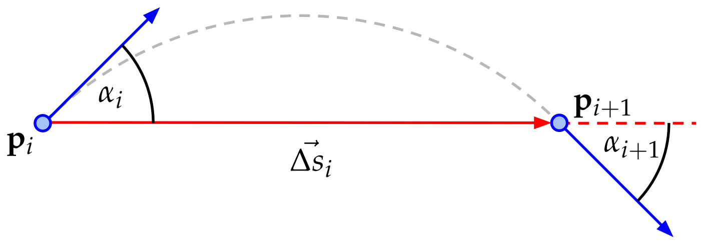


Figure 4.4: To restrict the trajectory to follow the principles of non-holonomic vehicles, each pair of adjacent poses $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ is required to be on a common arc segment of constant curvature. This is realized by comparing the angles α_i and α_{i+1} between the respective orientation of the poses and the vector $\vec{\Delta s}_i$.

$$\mathbf{p}_i = (x_i, y_i, \theta_i),\ \mathbf{p}_{i+1} = (x_{i+1}, y_{i+1}, \theta_{i+1})$$
$$\vec{\Delta s}_i = \begin{pmatrix} x_{i+1} - x_i \\ y_{i+1} - y_i \end{pmatrix} = \begin{pmatrix} \Delta x_i \\ \Delta y_i \end{pmatrix}$$
$$\theta_{\vec{\Delta s}_i} = \measuredangle(\vec{\Delta s}_i)$$
$$\alpha_i = \theta_i - \theta_{\vec{\Delta s}_i},\ \alpha_{i+1} = \theta_{i+1} - \theta_{\vec{\Delta s}_i}$$

The function f_{nhk} penalizes the deviation of the position and orientation from such an arc segment. This is realized by comparing the angles α_i and $-\alpha_{i+1}$ between the respective orientation of the poses and the orientation $\theta_{\vec{\Delta s}_i}$ of $\vec{\Delta s}_i$. $\vec{\Delta s}_i$ is the difference vector of the positions of $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$. The relation of α_i and α_{i+1} depends on the driving direction (compare Table 4.2).

$\alpha_i = -\alpha_{i+1}$	forward driving
$\pi - \alpha_i = -(\pi - \alpha_{i+1})$	backward driving
$\alpha_i = -(\pi - \alpha_{i+1})$	change driving direction from forward to backward
$\pi - \alpha_i = -\alpha_{i+1}$	change driving direction from backward to forward

Table 4.2: Relation of α_i to α_{i+1} depending on the driving direction.

$$\alpha_i = -\alpha_{i+1} \bigvee \pi - \alpha_i = -(\pi - \alpha_{i+1}) \bigvee \pi - \alpha_i = -\alpha_{i+1} \bigvee \alpha_i = -(\pi - \alpha_{i+1})$$
$$\Updownarrow$$
$$\begin{pmatrix} \cos\theta_i \\ \sin\theta_i \\ 0 \end{pmatrix} \times \begin{pmatrix} \Delta x_i \\ \Delta y_i \\ 0 \end{pmatrix} = -\left(\begin{pmatrix} \cos\theta_{i+1} \\ \sin\theta_{i+1} \\ 0 \end{pmatrix} \times \begin{pmatrix} \Delta x_i \\ \Delta y_i \\ 0 \end{pmatrix} \right)$$
$$\Updownarrow$$
$$\begin{pmatrix} \cos\theta_i \\ \sin\theta_i \\ 0 \end{pmatrix} \times \begin{pmatrix} \Delta x_i \\ \Delta y_i \\ 0 \end{pmatrix} = \begin{pmatrix} \Delta x_i \\ \Delta y_i \\ 0 \end{pmatrix} \times \begin{pmatrix} \cos\theta_{i+1} \\ \sin\theta_{i+1} \\ 0 \end{pmatrix}$$
$$\Updownarrow$$
$$\begin{pmatrix} 0 \\ 0 \\ \cos\theta_i \Delta y_i - \sin\theta_i \Delta x_i \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \sin\theta_{i+1} \Delta x_i - \cos\theta_{i+1} \Delta y_i \end{pmatrix}$$
$$\Updownarrow$$
$$\cos\theta_i \Delta y_i - \sin\theta_i \Delta x_i = \sin\theta_{i+1} \Delta x_i - \cos\theta_{i+1} \Delta y_i$$

$$f_{nhk}(\mathbf{p}_i, \mathbf{p}_{i+1}) = \left\| \frac{(\cos\theta_i + \cos\theta_{i+1})\Delta y_i - (\sin\theta_i + \sin\theta_{i+1})\Delta x_i}{\|\vec{\Delta s_i}\|} \right\|^2 \tag{4.4}$$

Rösmann et al. have shown a definition of an objective function for this purpose in [36], using the cross product of the respective orientation vectors and $\vec{\Delta s_i}$. This has the advantage of taking implicitly care of the cases when one or both of the the orientation angles θ_i and θ_{i+1} are rotated by 180°. Thus, it does not penalize traversing the arc segment in reverse direction or inverting the direction between the two poses.

For the proposed definition of f_{nhk} (Eq. 4.4) the definition by Rösmann et al. was extended by a normalizing term. By dividing through $\|\vec{\Delta s_i}\|$, the cost for violating the non-holonomic constraint is prevented from being influenced by the distance between poses.

As stated above, the curvature of the trajectory also needs to be constrained to the vehicle's limits. To that end, f_{turn} (Eq. 4.5) likewise uses the assumption of $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ being on a common arc segment and penalizes the radius of this arc segment being below a specified threshold r_{min} (i.e., the minimum turn radius of the vehicle).

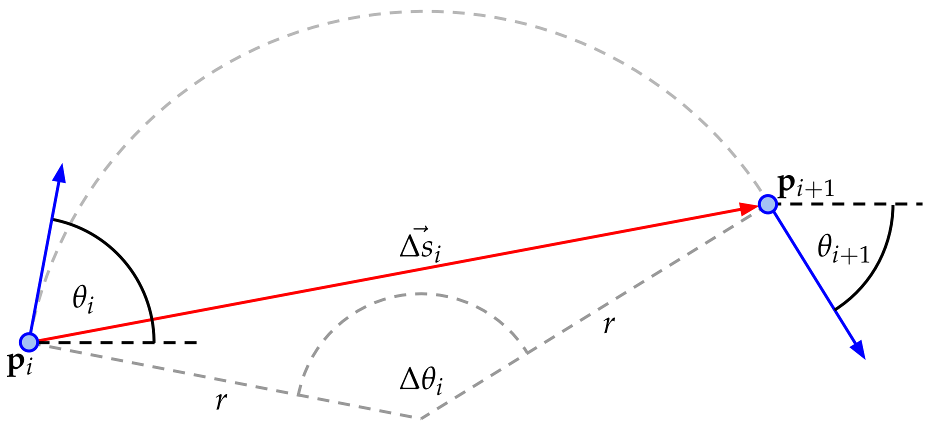


Figure 4.5: The radius r of the arc segment of constant curvature between each pair of adjacent poses $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ can be calculated from the chord length $\|\vec{\Delta s_i}\|$ and the angle $\Delta\theta_i$. $\Delta\theta_i$ is the difference of the respective orientations θ_i and θ_{i+1} normalized to the interval $[-\pi, \pi)$.

$$\Delta\theta_i = \begin{cases} (\theta_{i+1} - \theta_i + 2\pi) & \text{if } (\theta_{i+1} - \theta_i) < -\pi \\ (\theta_{i+1} - \theta_i - 2\pi) & \text{if } (\theta_{i+1} - \theta_i) \geq \pi \\ (\theta_{i+1} - \theta_i) & \text{otherwise} \end{cases}$$

$$r = \begin{cases} \left\| \frac{\|\vec{\Delta s}_i\|}{2\sin\left(\frac{\Delta\theta_i}{2}\right)} \right\| & \text{if } \theta_{i+1} \neq \theta_i \\ \infty & \text{otherwise} \end{cases}$$

$$f_{turn}(\mathbf{p}_i, \mathbf{p}_{i+1}, r_{min}) = e_\Gamma(-r, -r_{min}, 2) = \begin{cases} (r_{min} - r)^2 & \text{if } r < r_{min} \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

It has to be noted that the equations for both f_{nhk} and f_{turn} assume that the poses $\mathbf{p}_i$ are the respective center of rotation for the vehicle (usually the center of the rear axis). This is actually not the case in the context of the FUB_ROSCAR framework. Instead, the origin of the vehicle's coordinate system was set to the center of the front axle, as this is the default for most automotive sensors used in the project. This spatial offset can be taken into account by shifting the x and y coordinates of the poses $\mathbf{p}_i$. In the equations given above, this was omitted for clarity.

Furthermore, both objective functions in this section do not penalize reversing the direction of the trajectory between two poses; they ignore a difference of the orientation angles θ_i and $(\theta_{i+1}$ of more than 180° due to the normalization of $\Delta\theta_i$ to the interval $[-\pi, \pi)$. This property was kept intentionally, as future improvements of the proposed approach may include planning reverse driving. In the scope of this thesis, backward driving is effectively prohibited by penalizing it severely with the objective function $f_{forward}$ (Eq. 4.6) described in the following section.

4.2.2 Forward Driving

As mentioned in the previous section, the objective functions f_{nhk} and f_{turn} for ensuring the principle of non-holonomic kinetics and a minimum turn radius do not penalize inverting the direction between two consecutive poses. With no further constraints, this may produce trajectories that reverse the driving direction. This property may be a desirable feature, e.g., for planning parking maneuvers. The objective function related to maximum acceleration takes care of the necessary velocity reduction in those cases. However, in the context of cars, a change in the driving direction has to be penalized, as it usually requires some time to change the gear. In the scenarios evaluated in the context of this thesis, reversing the driving direction was not applicable; thus, the weight for the respective objective function $f_{forward}$ was set so that changes in the driving direction did not occur.

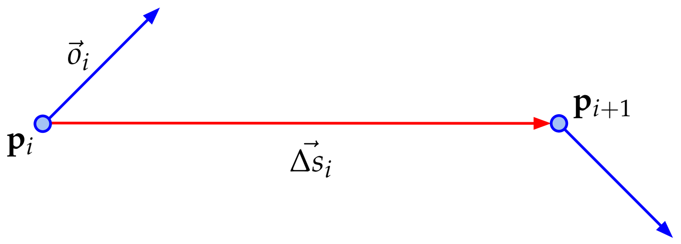


Figure 4.6: To restrict the orientation of pose $\mathbf{p}_i$ to being in forward direction, the dot product of its orientation vector $\vec{o}_i$ and the difference vector $\vec{\Delta s}_i$ of the positions of $\mathbf{p}_i$ and the next pose $\mathbf{p}_{i+1}$ is evaluated.

$$\begin{aligned} \mathbf{p}_i = (x_i, y_i, \theta_i),\ \mathbf{p}_{i+1} &= (x_{i+1}, y_{i+1}, \theta_{i+1}) \\ \vec{\Delta s}_i &= \begin{pmatrix} x_{i+1} - x_i \\ y_{i+1} - y_i \end{pmatrix} \\ \vec{o}_i &= \begin{pmatrix} \cos\theta_i \\ \sin\theta_i \end{pmatrix} \end{aligned}$$

$$f_{forward}(\mathbf{p}_i, \mathbf{p}_{i+1}) = e_\Gamma(-(\vec{\Delta s}_i \cdot \vec{o}_i), 0, 2) = \begin{cases} (\vec{\Delta s}_i \cdot \vec{o}_i)^2 & \text{if } (\vec{\Delta s}_i \cdot \vec{o}_i) < 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

The orientation of pose $\mathbf{p}_i$ is constrained by requiring that the subsequent pose $\mathbf{p}_{i+1}$ is "in front" of it. This is realized by comparing the orientation vector $\vec{o}_i$ of $\mathbf{p}_i$ to the difference vector $\vec{\Delta s}_i$ pointing from the position of $\mathbf{p}_i$ to $\mathbf{p}_{i+1}$. If the dot product $\vec{\Delta s}_i \cdot \vec{o}_i$ is negative, the squared dot product is returned as error.

4.2.3 Limiting Velocity and Acceleration

Another crucial aspect of constraining the trajectory is limiting the velocity and acceleration. The vehicle's dynamics are limited by the laws of physics and cannot exceed some specified maxima. However, the actual limits when driving in (urban) traffic are usually much lower. Additionally, we can have different thresholds for safety and comfort.

In the following, the five different objective functions related to the vehicle dynamics are discussed: f_{v_max} , f_{v_opt}, f_{a_lon}, f_{a_cen} and f_{a_ang}. While the former two functions restrict the maximum velocity and penalize deviation from an optimal velocity, the latter three are related to the acceleration (longitudinal, centripetal, and angular). While in the case of velocity, two (slightly) different functions are needed, for the acceleration functions, it is only necessary to adapt the respective thresholds and weights for a distinction between maximum and optimal values (relating to safety and comfort).

Maximum Velocity

The original objective function f_{v_max} for restricting maximum linear velocity, proposed by Rösmann et al. in [36], is quite simple: The penalty is proportional to the amount of velocity beyond a specified threshold. The velocity is calculated from the euclidean distance between a pair of subsequent poses $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ and the time difference Δt (which is equal for all i). As two poses are needed for the calculation, f_{v_max} represents a binary edge in the hyper-graph to optimize (compare Section 4.1.3) and is associated with each pair of adjacent poses.

The implementation in [47] presents a minor refinement, which was adopted in the context of this thesis: Instead of the euclidean distance $||\vec{\Delta s_i}||$, the arc distance b_i of an arc segment of constant curvature between the poses is used (compare Figure 4.7). This takes into account the vehicle's orientation and approximates the actual path traveled more accurately, especially in the case of a high angular velocity.

$$\mathbf{p}_i = (x_i, y_i, \theta_i),\ \mathbf{p}_{i+1} = (x_{i+1}, y_{i+1}, \theta_{i+1})$$
$$\vec{\Delta s_i} = \begin{pmatrix} x_{i+1} - x_i \\ y_{i+1} - y_i \end{pmatrix}$$

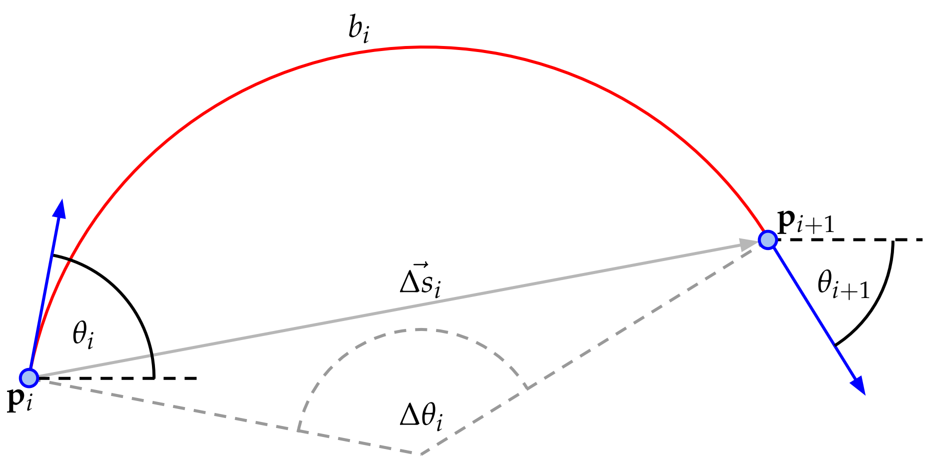


Figure 4.7: The arc distance b_i between each pair of adjacent poses $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ is used for calculating the respective velocity. b_i can be derived from the chord length $\|\vec{\Delta s}_i\|$ and the angle $\Delta\theta_i$. $\Delta\theta_i$ is the difference of the respective orientations θ_i and θ_{i+1} normalized to the interval $[-\pi, \pi)$.

$$\Delta\theta_i = \begin{cases} (\theta_{i+1} - \theta_i + 2\pi) & \text{if } (\theta_{i+1} - \theta_i) < -\pi \\ (\theta_{i+1} - \theta_i - 2\pi) & \text{if } (\theta_{i+1} - \theta_i) \geq \pi \\ (\theta_{i+1} - \theta_i) & \text{otherwise} \end{cases}$$

$$b_i = \begin{cases} \left\| \frac{\Delta\theta_i \|\vec{\Delta s}_i\|}{2\sin\left(\frac{\Delta\theta_i}{2}\right)} \right\| & \text{if } \theta_{i+1} \neq \theta_i \\ \|\vec{\Delta s}_i\| & \text{otherwise} \end{cases}$$

$$v_i = \frac{b_i}{\Delta t}$$

$$f_{v\_max}(\mathbf{p}_i, \mathbf{p}_{i+1}, v_{max}) = e_\Gamma(v_i, v_{max}, 1) = \begin{cases} v_i - v_{max} & \text{if } v_i > v_{max} \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

The threshold for the maximum velocity v_{max} (along with v_{opt} below) is particular in that it is not set to a static value externally, but is determined during the initialization process for each trajectory optimization, i.e., it is set only after the initial band is constructed (compare Section 4.4). This is because the maximum (and optimal) velocity depends on the environment.

While in most cases the maximum velocity is regulated by traffic rules, the approach in this thesis is not to rely on explicit information on the speed limit derived from a map or perceived traffic signs. Instead, we want to adapt to the behavior of other drivers. Therefore the threshold for maximum velocity is set relative to the velocities on the initial band, which incorporates the velocity of the target vehicle. Throughout this thesis, the threshold v_{max} was set to 110 % of the maximum velocity value of the initial elastic band. This allows the ego vehicle to close the distance to a leading vehicle but still penalizes substantially exceeding the velocities of observed cars.

Optimal Velocity

Another function related to the velocity is f_{v_opt}. It penalizes the deviation from an optimal (wanted) velocity. This is necessary since otherwise, the ego vehicle would not increase the velocity at all if the goal velocity (i.e., the velocity of the leading vehicle) is lower than the initial velocity; in other words, the ego vehicle could never get closer to the leading vehicle. As the number of poses is fixed and the time interval Δt between poses is static, we cannot address this with an objective function rewarding a lower total time. A possible alternative would be to reward spatially more extended trajectories, but by adjusting the optimal velocity threshold, the distance kept to the leading vehicle can be influenced implicitly in both directions (see calculation of v_{opt} below).

It also has to be noted that, although f_{v_opt} penalizes velocities higher then v_{opt}, keeping a separate function f_{v_max} and threshold v_{max} is still justified. While f_{v_max} must have a relatively high weight, as it is a safety-critical constraint, the weight for f_{v_opt} can be substantially lower since it should be neglected in favor of more critical constraints.

The implementation of f_{v_opt} is quite similar to f_{v_max}. The central part of calculating the velocity v_i is identical; it only differs in the final calculation of the penalty, where (instead of the amount of the velocity v_i exceeding the threshold v_{max}) the absolute difference of v_i and v_{opt} is returned. Likewise, f_{v_opt} also represents a binary edge and is associated with each pair of consecutive poses $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$.

$$f_{v\_opt}(\mathbf{p}_i, \mathbf{p}_{i+1}, v_{opt}) = e_\Gamma(\|v_i - v_{opt}\|, 0, 1) = \|v_i - v_{opt}\| \qquad (4.8)$$

Another similarity between the two velocity restricting functions is that the threshold v_{opt} is also calculated during initialization. This is necessary since it is implicitly responsible for maintaining a specified distance to the leading vehicle and therefore depends on the observed environment (in contrast to, e.g., thresholds for acceleration, which depend on the drivers' or passengers' preference). As v_{opt} directly influences the velocity on the whole band, it is used to catch up to a selected target vehicle o_t if the distance d_{target} to that vehicle is larger than a specified follow-distance d_{follow}, and fall back if d_o is smaller, respectively. To achieve this, v_{opt} is calculated dynamically based on the current velocity v_{target} of the target o_t (i.e., the vehicle to follow) and the distance d between the ego vehicle and o_t.

$$\begin{aligned} d_{follow} &= \max(5\ m, v_{ego} \cdot 1\ s) \\ v_{offset} &= 0.1\ s^{-1} \cdot (d - d_{follow}) \\ v_{opt} &= \min(v_{max}, v_{target} + v_{offset}) \end{aligned} \tag{4.9}$$

The distance d_{follow} is also calculated dynamically with respect to the current velocity of the ego vehicle. This corresponds to following the target with a temporal distance of 1 s. For low velocities, d_{follow} is limited from below, so it cannot fall below a specified minimum follow-distance of 5 m. Additionally, the value of v_{opt} is limited from above, so that it cannot exceed v_{max}.

Longitudinal and Angular Acceleration

Besides limiting the velocity on the trajectory, we also need to limit acceleration. On the one hand, the maximum acceleration is restricted by the vehicle's physical limits. On the other hand, we need to restrict those values to a range that is acceptable (or even comfortable) for human passengers, which is typically much lower. We also have to distinguish between linear and angular, as well as between longitudinal and lateral acceleration, as those forces are perceived differently.

In the presented approach, the maximum longitudinal linear and the angular acceleration are constrained explicitly. Respective objective functions f_{a_lon} and f_{a_ang} were adopted for this thesis as proposed by Rösmann et al. in [36]. They correspond to ternary edges in the underlying hyper-graph (compare Section 4.1.3) and are associated with each triple of consecutive poses $\mathbf{p}_i$, $\mathbf{p}_{i+1}$ and $\mathbf{p}_{i+2}$. In both cases, the return value is the amount of the respective acceleration exceeding the threshold.

The longitudinal acceleration $a_{lon,i}$ is calculated as the difference of the linear velocities v_i (between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$) and v_{i+1} (between $\mathbf{p}_{i+1}$ and $\mathbf{p}_{i+2}$) over the time interval Δt. The velocities are calculated using the arc distance, as described above for the velocity related functions.

$$a_{lon,i} = \frac{v_{i+1} - v_i}{\Delta t}$$

$$\begin{aligned} f_{a\_lon}(\mathbf{p}_i, \mathbf{p}_{i+1}, \mathbf{p}_{i+2}, a_{lon,max}) &= e_\Gamma(\|a_{lon,i}\|, a_{lon,max}, 1) \\ &= \begin{cases} \|a_{lon,i}\| & \text{if } \|a_{lon,i}\| < a_{lon,max} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{4.10}$$

The calculation of $a_{\theta,i}$ is analogous, using the difference of the angular velocities ω_i and ω_{i+1}, which in turn are calculated from the difference in the orientations θ_i of the respective poses.

$$\omega_i = \frac{\Delta\theta_i}{\Delta t}, \ \Delta\theta_i = \theta_{i+1} - \theta_i$$

$$a_{\theta,i} = \frac{\omega_{i+1} - \omega_i}{\Delta t}$$

$$\begin{aligned} f_{a\_ang}(\mathbf{p}_i, \mathbf{p}_{i+1}, \mathbf{p}_{i+2}, a_{\theta,max}) &= e_\Gamma(\|a_{\theta,i}\|, a_{\theta,max}, 1) \\ &= \begin{cases} \|a_{\theta,i}\| & \text{if } \|a_{\theta,i}\| < a_{\theta,max} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{4.11}$$

Centripetal Acceleration

Unlike longitudinal and angular acceleration, lateral acceleration is not constrained explicitly. Instead an objective function f_{a_cen} was implemented to restrict the centripetal acceleration a_c. Along with f_{a_lon} and f_{a_ang} this implicitly constraints lateral acceleration.

Restricting the centripetal acceleration$a_{c,i}$ has the advantage of also constraining the angular velocity ω_i with respect to the linear velocity v_i, as it is the product of those two values. This allows higher angular velocities at lower linear velocities and vice versa, i.e., it enables sharp turns at low speed while restricting them at higher speeds.

$$a_{c,i} = v_i \omega_i$$

$$f_{a\_cen}(\mathbf{p}_i, \mathbf{p}_{i+1}, a_{c,max}) = e_\Gamma(\|a_{c,i}\|, a_{c,max}, 1) = \begin{cases} \|a_{c,i}\| & \text{if } \|a_{c,i}\| < a_{c,max} \\ 0 & \text{otherwise} \end{cases} \tag{4.12}$$

Another advantage of constraining the centripetal acceleration over constraining the lateral acceleration is that it only needs two consecutive poses (instead of three). The objective function f_{a_cen}, therefore, corresponds to a binary edge in the underlying hyper-graph (compare Section 4.1.3) in contrast to the ternary edges related to longitudinal and angular acceleration.

While the thresholds for the velocity related functions are dynamic and dependent on the observed environment, the thresholds for f_{a_lon}, f_{a_cen} and f_{a_ang} are set externally to static values, which represent the preferences of the drivers (or passengers, respectively).

As the functions regarding acceleration penalize only values above the specified maximum, they were added twice to the global weighted multi-objective function $f(B)$ with different thresholds: Firstly with a threshold reflecting the dynamic limits acceptable for passengers, and secondly with a threshold of zero (which is most comfortable for passengers). The higher thresholds are needed to optimize the band towards a trajectory with reasonable limits for acceleration and velocity. Accordingly, the respective functions have high weights, as this represents a critical feature of the trajectory. In contrast, the functions with thresholds set to zero penalize any acceleration. They are added to favor a smooth and comfortable way of driving. Their corresponding weights are relatively low because we do not want them to interfere with more critical objectives of the optimization.

4.2.4 Avoiding Obstacles

Besides respecting the vehicle's dynamics (i.e., the vehicle must be able to follow the trajectory), the most critical aspect of trajectory planning is avoiding obstacles. The basic idea of the related objective function f_{ob} is straightforward: if the distance to an obstacle is smaller than some safety distance, the penalty is proportional to how much we violate this safety distance. The function f_{ob} corresponds to a unary edge in the graph optimization problem (an edge associated with a single node, compare Section 4.1.3), as it checks the distance to obstacles for every single pose $\mathbf{p}_i$ independently.

Geometric Representation of Objects

However, despite the simple basic idea, the actual calculation of the distance is not trivial for arbitrarily shaped objects. Their contour is approximated by more simple shapes to reduce the complexity. From the sensor processing modules of the FUB_ROSCAR Software Framework objects are passed to the trajectory planning module, including the following information: a unique id for tracking, the center pose and size of the three-dimensional arbitrarily oriented minimum bounding box, a list of points for the estimated contour of the object (two-dimensional on the ground plane), as well as linear and angular velocities for a tracked pose within the object. Furthermore, the objects are classified into static (i.e., non-moving) and dynamic ones. Also, a label for object classes, such as car, bicycle, or pedestrian, is available. For the representation of the objects within the proposed trajectory planning module, a distinction is made between static and dynamic ones (compare Figure 4.8).

For static objects, a single set of line segments $L_s = \{\mathbf{l}_0, \ldots, \mathbf{l}_n\}$ is stored. Each line segment $\mathbf{l}_i = (x_{i,0}, y_{i,0}, x_{i,1}, y_{i,1})$ is formed by connecting a pair of adjacent contour points of an individual object. The line segments stored in L_s are not associated with any specific obstacle anymore, as no further information than the segments themselves is required. An R-tree data structure [49] is used to accelerate the lookup of line segments in the proximity of a specified coordinate.

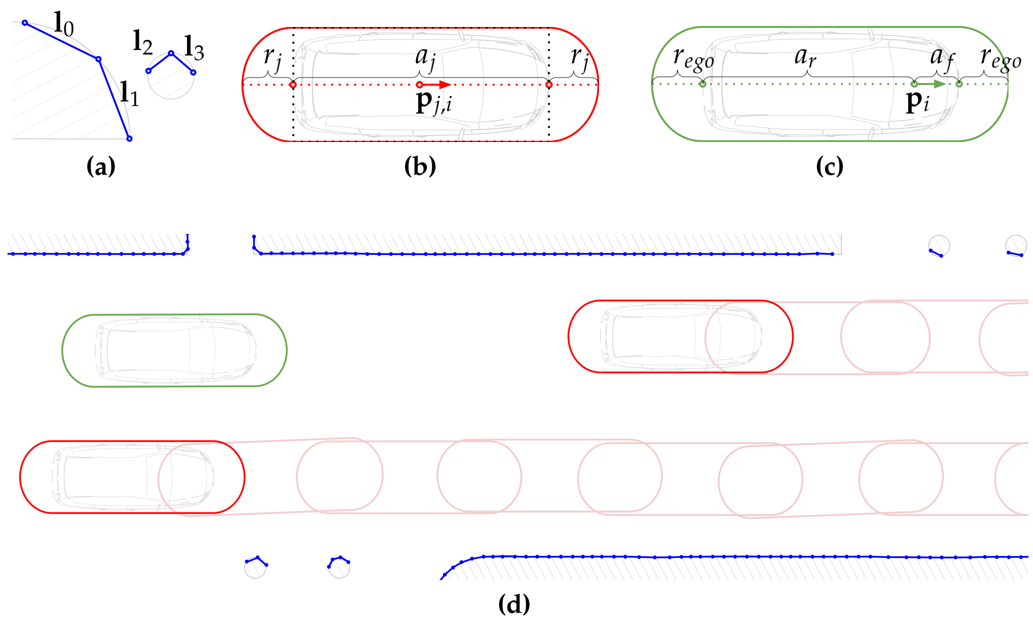


Figure 4.8: Geometric representations of objects. **(a)** All static objects are represented by a single set L_s of line segments $\mathbf{l}_i$ connecting the contour points of the individual objects. **(b)** A dynamic object $\mathbf{o}_j$ is represented by a geometric stadium shape defined by a center pose $\mathbf{p}_{j,i}$, a length a_j and a radius r_j. The values of a_j and r_j are calculated based on the bounding box of the object (black dotted lines). **(c)** The shape of ego vehicle (green) is defined in the same way as those of dynamic objects. As the poses $\mathbf{p}_i$ are not located in the center of the vehicle, but the on the front axle, two lengths a_r and a_f are needed to form the line segment from the rear to the front of the car. The stadium shape is then defined analogously to dynamic objects with the radius r_{ego}. **(d)** The poses of dynamic objects (red) are predicted with a step size of Δt. While the dimensions of the bounding boxes of dynamic obstacles is tracked over time, for static objects only the currently visible contour points (blue) are taken into account.

Using the contour is necessary for static objects since many non-moving obstacles are curved walls or other road boundaries. Such large convex or concave objects cannot be represented by a single bounding box accurately. The minimum distance between two adjacent contour points is 20 cm for the sensor setup used throughout the experiments. The maximum separation of contour points is not limited.

On the other hand, a different representation was chosen for dynamic objects because they require much more complex processing, as their poses change over time. The actually observed poses are tracked and future poses are predicted with an algorithm described in detail in Chapter 3. For each dynamic object $\mathbf{o}_j$ it returns a sequence of poses Q_j, where $\mathbf{p}_{j,i} \in Q_j$ represents the position and orientation of the object at a specific point in time. A time interval of Δt, matching the STEBLE elastic band's time interval, is assumed between two subsequent poses. The pose $\mathbf{p}_{j,0}$ is the currently observed pose of $\mathbf{o}_j$ and corresponds temporally to the start pose $\mathbf{p}_0$ of the elastic band, i.e., the current pose of the ego vehicle. Poses with an index $i \leq 0$ represent the recently observed states, the poses with $i > 0$ are predicted. For the obstacle avoidance a subsequence $\hat{Q}_j \subseteq Q_j$ is selected for each dynamic object $\mathbf{o}_j$.

$$\hat{Q}_j = (\mathbf{p}_{j,(0-m)}, \ldots, \mathbf{p}_{j,(n+m)}) \subseteq Q_j$$

$$\mathbf{p}_{j,i} = (x_{j,i}, y_{j,i}, \theta_{j,i})$$

The parameter m states how many time steps the object is predicted beyond the number of poses n of the initial elastic band. It is set in accordance with the desired longitudinal safety margin (see below in this section).

In addition to the computational cost of the actual prediction, the lookup of dynamic objects cannot be accelerated in the same way as static objects. While the line segment representing the latter can be stored in one single R-tree data structure to accelerate the distance calculation for all poses, the dynamic obstacles would need an individual tree for each time step. This strategy is ineffective unless a very high number of iterations is used for the optimization loop (which proved unnecessary, compare Section 5.3.2).

Thus, to reduce the complexity for dynamic objects, the contour points are entirely ignored. Instead, each dynamic object $\mathbf{o}_j$ is represented by a geometric stadium shape, derived from its arbitrarily oriented minimum bounding box (compare Figure 4.8b). Most dynamic objects in urban traffic can either be approximated quite well with this shape (cars, trucks, motorcycles, bicycles) or are so small that there is no significant difference to using the contour (pedestrians, animals). The geometric stadium effectively adds two half circles to the minimum bounding box. However, it substantially reduces the computational effort of calculating the distance to other objects, compared to directly using the rectangular bounding box, as the distance calculation effectively uses a single line segment. Another major advantage of using the stadium shape is that its definition with a line segment and radius implicitly defines the center of the object. This enables considering the distance to the center in the objective function, so that band poses are pushed outwards of objects (compare Eq. 4.14 and Eq. 4.15).

The stadium shape of dynamic object $\mathbf{o}_j$ at a specific point in time t_i can be calculated from the (predicted) center pose $\mathbf{p}_{j,i}$, a length a_j and a radius r_j (with $\mathbf{p}_{j,i}$ and a_j defining the line segment mentioned above, compare Figure 4.8b). While $\mathbf{p}_{j,i}$ needs to be computed for each step i, the values of a_j and r_j are calculated for each object $\mathbf{o}_j$ only once. They are derived directly from the dimensions of the bounding box of $\mathbf{o}_j$, with a_j being the length and r_j half of the width. Another advantage of using the arbitrarily oriented minimum bounding box over the contour points, besides the much more simple computation, is that the bounding box is tracked over time for each object $\mathbf{o}_j$ (in the sensor processing modules). The reported bounding box's dimensions take into account previous observations of the corresponding object, while contour points are only available for the currently visible portions of the object.

The shape of the ego vehicle is represented in the same way as dynamic obstacles for calculating the distance to objects (compare Figure 4.8c). The contour of the car is approximated quite well by the geometric stadium shape. In contrast to dynamic objects, the poses $\mathbf{p}_i$ of the ego vehicle (i.e., the elastic band) do not describe the center of the object but the center of the front axle. For this reason, two length parameters a_r and a_r are needed to define the length of the stadium shape. Together $\mathbf{p}_i$, a_r and a_r define a line segment, which in turn defines the geometric stadium shape with the radius r_{ego}.

Dynamic Longitudinal Safety Distance

Another critical aspect of obstacle avoidance in traffic is the difference in safety margins for lateral and longitudinal separation from other road users. The lateral distance to objects moving in the same direction can get relatively low (in the range of centimeters for urban traffic). This does not pose an immediate danger, as changes in the lateral velocity are typically much slower than their longitudinal counterparts and can be predicted from the geometry of the lanes (assuming that drivers check for free space when performing lane-change maneuvers). The longitudinal safety margin, on the other hand, is usually much larger. It strongly depends on the velocity of the vehicles. Instead of specifying a fixed value, the recommended safe distance for following another vehicle is often given in seconds (e.g., the German law requires a minimum distance corresponding to 0.9 s). Compare Section 5.3.1 for a detailed discussion on the optimal safety margins.

In the first iterations of the presented approach, the dynamic property and relatively larger size of the longitudinal safety distance was taken into account by simply extending the length a_j by a multiple of the velocity of the object [39]. This strategy works for straight trajectories with simulated measurements free of noise or very small variations of this use case. However, it is not feasible in many real-world cases.

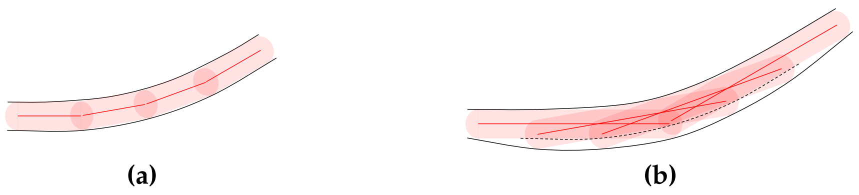


Figure 4.9: Extending objects in longitudinal direction is not a feasible approach for maintaining separate longitudinal and lateral safety margins. Compared to the object predicted with actual dimensions **(a)**, the area occupied by the predicted longitudinally extended object is substantially larger on the outer side of curves **(b)** and thus imposing more lateral distance then necessary.

If the predicted trajectories contain curves or the perception of the orientation of the objects is noisy (which is more or less always the case for actual sensor data), this is not a valid approach (compare Figure 4.9). Instead of extending the objects in one spatial dimension, a much more practical solution is to use multiple predicted poses of the object at different points in time (compare Figure 4.10). This effectively projects the shape of the object along its predicted trajectory.

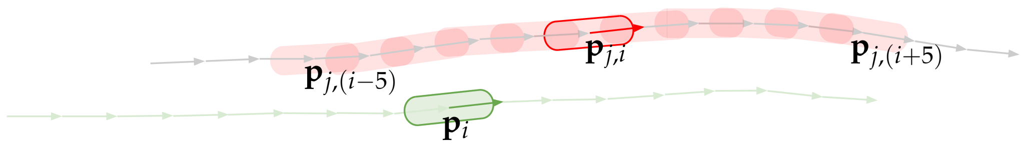


Figure 4.10: Increasing the longitudinal safety margin depending on the object's velocity is accomplished by measuring the distance of the ego vehicle at pose $\mathbf{p}_i$ to the object $\mathbf{o}_j$ at several consecutive poses $\mathbf{p}_{j,i}$. The poses $\mathbf{p}_{j,i}$ represent the objects predicted state at different points in time.

For each pair of a pose $\mathbf{p}_i$ of the elastic band and a dynamic object $\mathbf{o}_j$ a sequence $\hat{Q}_{j,i}$ of poses is selected from all predicted poses $\hat{Q}_j$ of $\mathbf{o}_j$.

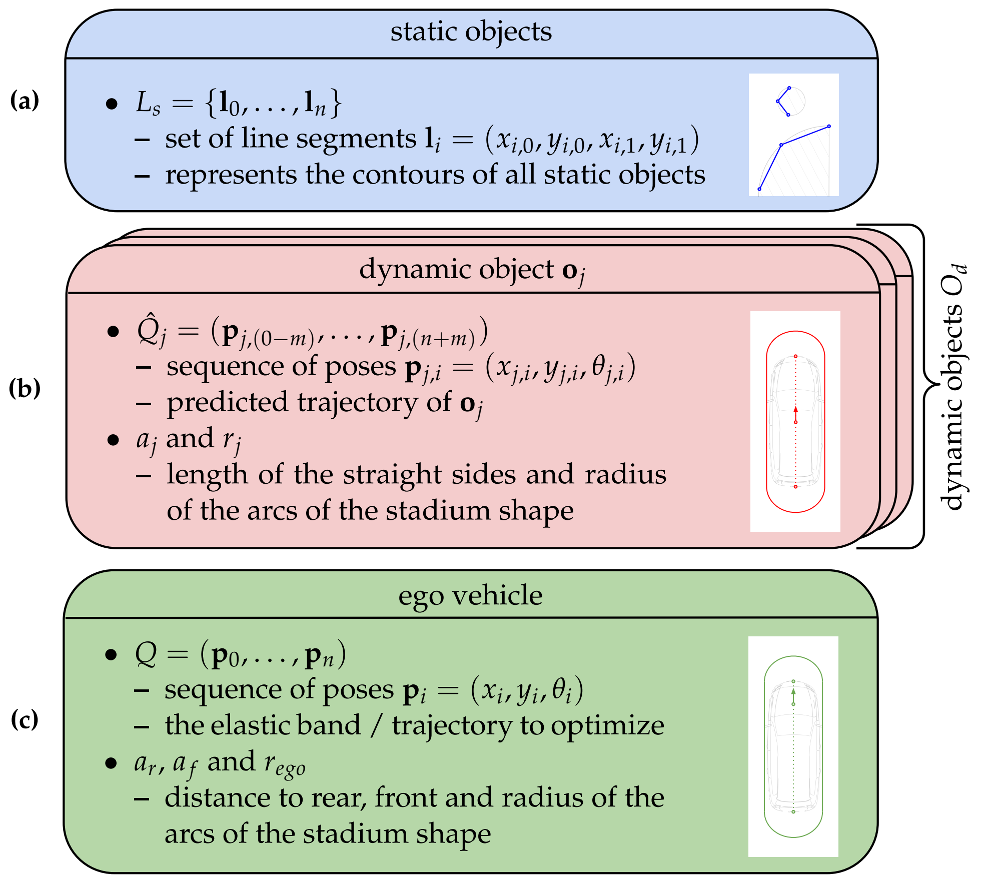
$$\begin{aligned} \hat{Q}_{j,i} &= (\mathbf{p}_{j,(i-m)}, \dots, \mathbf{p}_{j,i}, \dots, \mathbf{p}_{j,(i+m)}) \\ \hat{Q}_{j,i} &\subseteq \hat{Q}_j = (\mathbf{p}_{j,(0-m)}, \dots, \mathbf{p}_{j,(n+m)}) \end{aligned} \tag{4.13}$$

All sequences $\hat{Q}_{j,i}$ are subsequences of object's $\mathbf{o}_j$ predicted trajectory $\hat{Q}_j$. As described above, $\hat{Q}_j$ contains all predicted poses of the object $\mathbf{o}_j$, as well as some recently observed, exceeding the number of poses and respective points in time of the elastic band by m in the past and future. Each subsequence $\hat{Q}_{j,i}$ is centered around a pose $\mathbf{p}_{j,i}$ which corresponds to a pose $\mathbf{p}_i$ of the elastic band. For each $\hat{Q}_{j,i}$ the poses from $\hat{Q}_j$ are selected, where the index is greater than or equal to $(i - m)$ and less than or equal to $(i + m)$. The threshold m is set according to the preferred longitudinal distance depending on the value of Δt.

All of the poses in all sequences $\hat{Q}_{j,i}$ are checked for a violation of the minimum distance against their corresponding elastic band pose $\mathbf{p}_i$ (taking into account the respective shapes of the objects). Of course, this increases the number of distance checks substantially (by a factor of $2m$), compared to simply extending the length of the object at a single point in time. As mentioned above, this is one of the reasons that the shape of dynamic objects is approximated by a stadium shape instead of taking into account the individual contour points (as is done for static objects).

Summing Up the Objects' Penalties

The basic idea of the objective function f_{ob} presented in this chapter is straightforward: For each object closer to the ego vehicle than a threshold distance d_{min}, the amount of the euclidean distance falling below the threshold is squared and summed up as return value. The input parameters for f_{ob} regarding the state of the ego vehicle, as well as static and dynamic objects, are summarized in Figure 4.11. A set of line segments represents static objects. For the ego vehicle and dynamic objects, the combination of poses and a length can also be interpreted as line segments. That is, for both classes of objects, the calculation of the minimum separation to the ego vehicle comes down to calculating the distance between two line segments and comparing it to some threshold. The radii



(a)

static objects

- $L_s = \{\mathbf{l}_0, \ldots, \mathbf{l}_n\}$
  - set of line segments $\mathbf{l}_i = (x_{i,0}, y_{i,0}, x_{i,1}, y_{i,1})$
  - represents the contours of all static objects

(b)

dynamic object $\mathbf{o}_j$

- $\hat{Q}_j = (\mathbf{p}_{j,(0-m)}, \ldots, \mathbf{p}_{j,(n+m)})$
  - sequence of poses $\mathbf{p}_{j,i} = (x_{j,i}, y_{j,i}, \theta_{j,i})$
  - predicted trajectory of $\mathbf{o}_j$
- a_j and r_j
  - length of the straight sides and radius of the arcs of the stadium shape

dynamic objects O_d

(c)

ego vehicle

- $Q = (\mathbf{p}_0, \ldots, \mathbf{p}_n)$
  - sequence of poses $\mathbf{p}_i = (x_i, y_i, \theta_i)$
  - the elastic band / trajectory to optimize
- a_r, a_f and r_{ego}
  - distance to rear, front and radius of the arcs of the stadium shape

Figure 4.11: Summary of the object parameters used in the objective function f_{ob}. Compare Figure 4.8 for a description of the geometric shapes. **(a)** All line segments $\mathbf{l}_i$ derived from the static objects' contours are stored in a single set L_s. **(b)** Each dynamic object $\mathbf{o}_j \in O_d$ is defined by its predicted trajectory $\hat{Q}_j$, a length a_j and a radius r_j. **(c)** The ego vehicle is described similar to the dynamic objects, but with two lengths a_r and a_f.

of the stadium shapes can be taken into account by subtracting them from the threshold. The distance between line segments is calculated quite frequently. The number of calls to this method varies between 100000 and 500000 times per planning iteration with the parameter set used throughout the experiments. Vladimir J. Lumelsky proposed an efficient algorithm for this distance calculation [50].

A specialization of Lumelsky's algorithm was used in this project, the pseudo code can be found in Appendix A.

$$\text{dist}(\mathbf{l}_p, \mathbf{l}_q) = \text{min. distance between } \mathbf{l}_p \text{ and } \mathbf{l}_q \text{ (see Appendix A)}$$

The line segment $\mathbf{l}_{ego,i}$ representing the ego vehicle with rear and front distance a_r and a_f at each pose $\mathbf{p}_i$ of the elastic band can be calculated as follows. It is defined by the quadruple of the x- and y-coordinates of the start and end point.

$$\begin{aligned}
\mathbf{p}_i &= (x_i, y_i, \theta_i) \\
\mathbf{l}_{ego,i} &= (x_i - a_r \cos\theta_i, y_i - a_r \sin\theta_i, x_i + a_f \cos\theta_i, y_i + a_f \sin\theta_i)
\end{aligned}$$

As static and dynamic objects have different representations, the computation of f_{ob} is essentially split into two steps: determining the cost $c_{s,i}$ (Eq. 4.14) for static and $c_{d,i}$ (Eq. 4.15) for dynamic objects.

In the first step the sum $c_{s,i}$ of all penalties $g_{i,k}$ for all static objects violating the minimum distance d_{min} to a specific pose $\mathbf{p}_i$ of the band is computed. To take into account the stadium shape associated with $\mathbf{p}_i$, the actual distance $d_{i,k}$ between the object and the ego vehicle is the difference of the distance between the corresponding line segments $\mathbf{l}_k$ and $\mathbf{l}_{ego,i}$ and the radius r_{ego} of the stadium shape representing the ego vehicle. By subtracting r_{ego} it is possible that $d_i < 0$. This means the penalty is increased the closer the object is to the center of the ego vehicle. That is, the more the objects overlap, the higher the penalty. This is necessary to push the band outwards, if the ego vehicle's shape overlaps an object and the distance of the contours is zero regardless of the extend of the overlap.

$$\begin{aligned}
\mathbf{l}_k &\in L_s \\
d_{i,k} &= \text{dist}(\mathbf{l}_{ego,i}, \mathbf{l}_k) - r_{ego} \\
g_{i,k} = g(\mathbf{l}_{ego,i}, \mathbf{l}_k, r_{ego}, d_{min}) &= e_\Gamma(d_{i,k}, d_{min}, 2) \\
&= \begin{cases} (d_{i,k} - d_{min})^2 & \text{if } d_{i,k} < d_{min} \\ 0 & \text{otherwise} \end{cases} \\
c_{s,i} = c_s(\mathbf{l}_{ego,i}, L_s, r_{ego}, d_{min}) &= \sum_{\mathbf{l}_k \in L_s} g_{i,k} \qquad (4.14)
\end{aligned}$$

It has to be noted that $c_{s,i}$ (as well as $c_{d,i}$ below) is based on a quadratic cost function e_Γ. This is a better fit than a linear cost since the gradient of the quadratic function mirrors a property of obstacle avoidance: major violations of the specified minimum distance are fatal for the safety of the trajectory, while slight breaches can be neglected.

In the second step the penalty for dynamic objects is calculated. For the pose $\mathbf{p}_i$ associated with f_{ob}, the sum $c_{d,i}$ adds up the individual penalties $h_{i,j}$ for each dynamic object $\mathbf{o}_j \in O_d$ violating the minimum distance d_{min}. For the computation of $h_{i,j}$ the sequence of poses $\hat{Q}_{j,i}$ is selected from the sequence $\hat{Q}_j$ of all predicted poses of the respective $\mathbf{o}_j$ (see Eq. 4.13 and corresponding subsection above for a detailed description).

$$\mathbf{o}_j = (\hat{Q}_j, a_j, r_j) \in O_d$$

$$\hat{Q}_{j,i} \subseteq \hat{Q}_j$$

From each sequence $\hat{Q}_{j,i}$ and the length of the object a_j a set $L_{j,i}$ of line segments is derived.

$$\text{lineseg}(x, y, \theta, a) = (x - a\cos\theta, y - a\sin\theta, x + a\cos\theta, y + a\sin\theta)$$

$$L_{j,i} = \{\text{lineseg}(x, y, \theta, \frac{a_j}{2}) : \mathbf{p} = (x, y, \theta) \in \hat{Q}_{j,i}\}$$

The minimum distance $d_{i,j}$ of the ego vehicle at pose $\mathbf{p}_i$ to the object $\mathbf{o}_j$ is then calculated as the minimum of the set of distances to all line segments in $L_{j,i}$. The subsequent calculation of $c_{d,i}$ is analogous to the computation of the cost $c_{s,i}$ of static objects. Again, the radii of the stadium shapes are taken into account by subtracting them from the distance between the line segments (this time including also r_j of the object).

$$\begin{aligned}
d_{i,j} &= \min(\{\text{dist}(\mathbf{l}_{ego,i}, \mathbf{l}) : \mathbf{l} \in L_{j,i}\}) - r_{ego} - r_j \\
h_{i,j} = h(\mathbf{l}_{ego,i}, \mathbf{o}_j, r_{ego}, d_{min}) &= e_\Gamma(d_{i,j}, d_{min}, 2) \\
&= \begin{cases} (d_{i,j} - d_{min})^2 & \text{if } d_{i,j} < d_{min} \\ 0 & \text{otherwise} \end{cases} \\
c_{d,i} = c_d(\mathbf{l}_{ego,i}, O_d, r_{ego}, d_{min}) &= \sum_{\mathbf{o}_j \in O_d} h_{i,j}
\end{aligned} \tag{4.15}$$

The final value of f_{ob} is then simply the sum of the cost $c_{s,i}$ (Eq. 4.14) for static and $c_{d,i}$ (Eq. 4.15) for dynamic objects.

$$f_{ob} = c_{s,i} + c_{d,i} \tag{4.16}$$

There are two reasons why a single objective function f_{ob} for dynamic and static objects is used (although the individual penalties are only merged in the final step). Firstly, both $c_{s,i}$ and $c_{d,i}$ (unlike any other objective) depend on the computation of the line segments $\mathbf{l}_{ego,i}$ representing the ego vehicle. By merging both in a single objective function, this has to be done only once. The second reason is that avoiding static and dynamic obstacles should have the same weight in the global multi-objective function $f(B)$. The two different cost functions are defined to have a fluid transition if an object switches from static to dynamic state, and this property would be invalidated with different weights. Also, when the planned trajectory runs through a narrow space between a static and a dynamic object, the band's poses should end up centered between those objects to minimize the violation of the respective safety margins. Given the balanced property of the cost functions, this also requires uniform weights.

The objective function f_{ob} is one of only two functions represented as unary edge in the hyper-graph to optimize (compare Section 4.1.3). It is assigned to each band pose $\mathbf{p}_i$ (i.e., node of the graph). Each of those edges affects only a single pose of the band (or node in the graph, respectively); all other properties such as the objects' states or the geometric shape of vehicles are not changed during the optimization process. Furthermore, f_{ob} is the only objective function presented, which directly uses the index i of its associated pose $\mathbf{p}_i$. As described above, the index is related to a specific point in time and thus needs to look up the corresponding states of dynamic vehicles. The states of the objects can be calculated once in advance of the optimization process, due to the property of having a fixed time interval Δt between the band poses all relevant points in time are multiples of Δt.

Furthermore, the implementation of f_{ob} uses an R-tree data structure [49] to accelerate the lookup of line segments in the proximity of the corresponding pose's coordinate so that the actual distance calculation only have to be performed on a subset of all line segments. As stated above, this is only done for the line segments representing static objects. For dynamic objects, an individual set of line segments is needed for each band pose $\mathbf{p}_i$, as they represent different points in time. This substantially increases the number of line segments for dynamic objects and is the main reason for their comparatively simpler geometric representation. As using this simpler shape effectively reduces the number of line segments to one per dynamic object per time step, there is no benefit in constructing multiple R-trees compared to simply calculating the distance to all line segments (unless the number of iterations of the optimization process is much higher than in the presented approach).

4.2.5 Following Trajectories of Other Vehicles

A central aspect of the STEBLE approach is the objective function f_{path}. It penalizes the separation of the ego vehicle's trajectory (i.e., the elastic band's poses) to other vehicles' paths. On the one hand, this reflects the intention of driving on a trajectory close to the paths of other vehicles - one of the basic ideas of the presented approach. If other cars were driving on a specific path recently, there is a high chance of this being a safe and comfortable path for the ego vehicle. On the other hand, the function f_{path} is critical for another aspect of STEBLE: the position of the last pose of the band is not fixed. Without an objective attracting the poses to other vehicles' paths, the optimal path (disregarding collision avoidance) would be a straight line in the direction of the start pose since this would minimize the centripetal and angular acceleration. The objective function f_{path} is the only one, which guides the ego vehicle towards its goal of following other vehicles. It corresponds to a unary edge in the graph optimization problem (an edge associated with a single node, compare Section 4.1.3). The objective function f_{path} is associated with each pose $\mathbf{p}_i$ of the elastic band, except for the first.

Besides the respective pose $\mathbf{p}_i$, the input for f_{path} is a set L_p of line segments, representing the other vehicles' paths. An important decision regarding f_{path} is to choose which objects contribute to this set of possible paths. While only the best target to follow is selected for initializing the elastic band (compare Section 4.3), for f_{path}, it is not necessary to limit the choice to a single vehicle. Depending on the start pose and obstacles blocking the way, it may be beneficial to follow the paths of other vehicles (which are not the current target to follow) for some distance. Nonetheless, the presented approach does not take into account the trajectories of all observed objects. A simple heuristic is applied to filter the candidates.

In this regard, the first criterion is that the object is classified as a vehicle since the ego vehicle should not follow pedestrians or unknown objects. The information on the class of the object is passed to the trajectory planning module from other modules of the FUB_ROSCAR Framework that are not in the scope of this thesis. Secondly, only dynamic objects are considered (i.e., objects that have moved at some point in time), as only those can have a valid trajectory.

The further criteria for objects to contribute to the set of possible paths are based on the object's trajectory. For this evaluation, only the tracked objects' states are considered that were actually observed. To represent those observed states, for each dynamic object $\mathbf{o}_j$ a sequence of poses P_j is maintained. The algorithm for tracking (and predicting) dynamic objects is described in detail in Chapter 3. Similar to the sequence $\hat{Q}_j$ of predicted poses used for obstacle avoidance (compare Section 4.2.4), each pose $\mathbf{p}_{j,k} \in P_j$ represents an observed position and orientation of the object at a specific point in time.

$$P_j = (\mathbf{p}_{j,-n_j}, \ldots, \mathbf{p}_{j,0})$$

$$\mathbf{p}_{j,k} = (x_{j,k}, y_{j,k}, \theta_{j,k})$$

The observed poses' corresponding timestamps are not stored explicitly. They can be derived from the index k due to a constant time interval of Δt between two subsequent poses. It has to be noted that the time interval Δt for the observed poses corresponds to the time interval for the elastic band and the predicted poses. This is not mandatory since, for f_{path}, only the spatial information is used. On the other hand, it has advantages for the algorithm predicting dynamic objects (compare Chapter 3). The number of observed poses n_j can be different for each object $\mathbf{o}_j$. It depends on the duration for which the object has been observed and is limited from above to some threshold for computational efficiency.

Given the observed trajectory P_j of an object $\mathbf{o}_j$ (which is classified as a vehicle and has moved at some point in time) and the pose $\mathbf{p}_0$ of the ego vehicle (i.e., the start pose of the elastic band), two conditions are evaluated to decide if $\mathbf{o}_j$ contributes to the set of line segments L_p representing the paths to follow:

- P_j contains at least 2 poses in forward direction of $\mathbf{p}_0$. This is evaluated using the dot product of the orientation vector of $\mathbf{p}_0$ and the difference vector of the positions similar to $f_{forward}$ in Section 4.2.2.
- The absolute difference between the orientation of $\mathbf{p}_0$ and the orientation of the pose in P_j which is closest to $\mathbf{p}_0$ is below 90°. This ensures that trajectories of vehicles in the opposite direction are ignored.

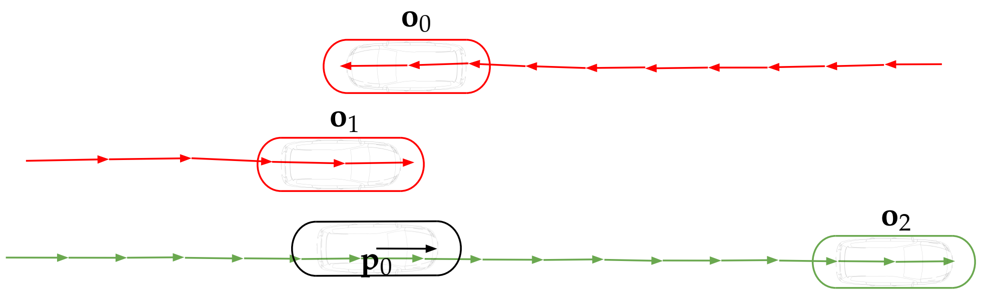


Figure 4.12: The path of all dynamic objects classified as vehicles are checked for two conditions to be included in the set of valid paths to follow. The path of $\mathbf{o}_0$ is not taken into account because its closest pose is in the opposite direction of the ego vehicle's pose $\mathbf{p}_0$. The path of $\mathbf{o}_1$ is not considered because fewer then two poses are in the forward direction of $\mathbf{p}_0$. The path of $\mathbf{o}_2$ fulfills both conditions.

All trajectories P_j fulfilling the above criteria are added to a set P. The set L_p is then generated from P, consisting of a line segment between each non-equal pair of subsequent poses. The individual line segments are defined by the quadruple of the x- and y-coordinates of the start and end pose.

$$\text{lineseg}(\mathbf{p}_p, \mathbf{p}_q) = (x_p, y_p, x_q, y_q)$$

$$L_p = \{\text{lineseg}(\mathbf{p}_{j,k}, \mathbf{p}_{j,(k+1)}) : (\mathbf{p}_{j,k}, \mathbf{p}_{j,(k+1)}) \subseteq P_j, \mathbf{p}_{j,k} \neq \mathbf{p}_{j,(k+1)}, P_j \in P\}$$

Having obtained the set L_p of line segments representing possible paths to follow, the remaining task of determining the minimum distance of those line segments to the respective pose $\mathbf{p}_i$ associated with f_{path} is simple.

Calculating the distance of a two-dimensional position (x, y) to a line segment $\mathbf{l}$ was implemented in a straightforward way using the dot product of two vectors. The pseudo code can be found in Appendix A.

$$\text{dist}(x, y, \mathbf{l}) = \text{min. distance between point } (x, y) \text{ and line segment } \mathbf{l}$$

The cost value returned by f_{path} is then simply the minimum $d_{min,i}$ of the distances to all line segments $\mathbf{l} \in L_p$.

$$\mathbf{p}_i = (x_i, y_i, \theta_i)$$

$$f_{path}(\mathbf{p}_i, L_p) = d_{min,i} = \min(\{\text{dist}(x_i, y_i, \mathbf{l}) : \mathbf{l} \in L_p\}) \tag{4.17}$$

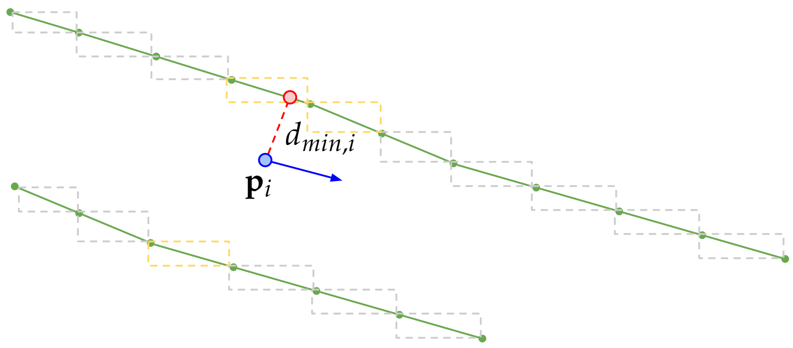


Figure 4.13: The objective function f_{path} evaluates the minimum distance $d_{min,i}$ of its corresponding pose $\mathbf{p}_i$ to the closest line segment. The axis aligned bounding boxes (AABB) of the line segments are stored in a R-tree for faster lookup. The distance to the AABB may be smaller then the distance to the actual line segment. Therefore, the exact distance to the line segment is calculated for the k closest AABB (yellow), where k is the number of trajectories (i.e., distinct objects) contributing to L_p plus one.

The distance calculation can be accelerated by first filtering the candidates using an R-tree data structure [49] to determine the k nearest neighbors. Because the R-tree only takes into account the axis-aligned bounding boxes, the k nearest neighbors need to be evaluated for the exact distance (compare Figure 4.13). When k is the number of distinct trajectories contributing to L_p plus one, the algorithm is guaranteed to find the closest line segment.

Another way to reduce the computational complexity of f_{path} is to limit the number of trajectories that contribute to the set of possible paths. It is also feasible to use only one trajectory. In the vast majority of evaluated planning iterations, there is no difference in the calculated cost using only the trajectory of the selected target to follow versus using all observed trajectories. This is due to the ego vehicle usually being on or very close to the trajectory of the car it is following. However, in the more challenging scenarios, especially a merge into traffic maneuver of the leading vehicle (compare Section 5.3), the availability of alternative paths on other lanes is quite helpful for providing more stable planned trajectories in subsequent iterations. For this reason, the proposed implementation takes into account all valid trajectories, as described above.

Furthermore, it has to be noted that the above cost function assumes that both, the poses $\mathbf{p}_i$ of the ego vehicle and the poses $\mathbf{p}_{j,k}$ of the tracked objects, are assumed to be the center of rotation for the respective vehicle (which is usually at the center of the rear axis). The actual center of rotation for the tracked objects is unknown; their tracked poses represent the object's center. Due to measurement noise, efforts to calculate the offset to the center of rotation turned out too inaccurate to have any noticeable positive effect. Also, in the context of the FUB_ROSCAR framework, the origin of the ego vehicle's coordinate system was set to the center of the front axle, as this is the default for most automotive sensors used in the project. This spatial offset can be easily taken into account by shifting the x and y coordinates of the poses $\mathbf{p}_i$ to the center of the ego vehicle to match the objects' representation. In the equations given above, this shift was omitted for clarity.

4.3 Selecting A Target to Follow

As described in previous sections, the general idea of the STEBLE approach is to follow other vehicles. The related objective function (compare Section 4.2.5) is designed in a way that it is not necessary to choose a specific target to follow, but instead, the trajectory of the ego vehicle is attracted by a set of trajectories selected from all observed vehicles. Nonetheless, several other aspects of STEBLE benefit from selecting a single target vehicle to follow. Taking into account the trajectory of a specific target for the initialization of the elastic band (compare Section 4.4) leads to much faster convergence of the optimization; a statistical analysis can be found in Section 5.3.2. Furthermore, the velocity of the ego vehicle is influenced to keep a specific distance to the chosen target vehicle (compare Section 4.2.3). Consequently, the ego vehicle matches the speed of a single leading vehicle. This integrates the ego vehicle into the swarm of other road users while still allowing other drivers to overtake without affecting the velocity of the ego vehicle.

The remainder of this section describes the algorithm for selecting a target vehicle $\mathbf{o}_t$ from a set of candidates $O = \{\mathbf{o}_0, \ldots, \mathbf{o}_n\}$. The algorithm heavily utilizes the output (and intermediate results) of the tracking and swarm-based trajectory prediction described in Chapter 3. Consequently, the set O contains the elements of the filtered sequence corresponding to the observed vehicles driving in the same direction as the ego vehicle (compare Section 3.4). For each of the candidate vehicles $\mathbf{o}_j \in O$, a sequence of poses Q_j is available from the post-processing modules, describing the observed and predicted trajectory.

The general idea of the algorithm is to select the vehicle whose state resembles most closely the current state of the ego vehicle. For this task, all candidate vehicles $\mathbf{o}_j \in O$ are evaluated based on four weighted criteria: the difference in distance, orientation, and velocity, as well as the duration for which the ego vehicle was following the vehicle in previous planning iterations. While the first three criteria are directly related to the similarity of the vehicles' states, the duration for which a vehicle was followed can be interpreted as a measurement of trust. It is also used to provide a hysteresis and thus discourage frequent changes of the target vehicle. The distance to the ego vehicle's current position is evaluated twice for each object: once for the position of $\mathbf{o}_j$ most recently observed and once for the spatially closest position on the respective trajectory Q_j. While the latter value is calculated only in accordance with the general idea of rewarding similarity to the ego vehicle's state, the former value rewards a more recent similarity. This is useful for breaking ties when two vehicles have a spatially similar but temporally shifted trajectory.

Table 4.3 gives an overview of the criteria and weights used to evaluate the best target to follow. It has to be noted, that all values $c_{k,j}$ are within the range $[0, 1]$. For the first criterion $c_{1,j}$ - the duration object $\mathbf{o}_j$ was followed - this is realized by clamping the value to a maximum of 1.0 s. For the other criteria, the values are normalized between all objects $\mathbf{o}_j \in O$ in a way that the respective state most similar to the ego vehicle's state is 1 and the one farthest away is 0.

Criterion $c_{k,j}$	Weight ω_k	Description
$c_{1,j}$	0.5	Duration (in seconds) for which $\mathbf{o}_j$ was selected continuously as target to follow in previous iterations. Clamped to a maximum of 1.0 s.
$c_{2,j}$	0.2	Normalized distance from the ego vehicle's current position to the most recently observed position of $\mathbf{o}_j$.
$c_{3,j}$	1.0	Normalized distance from the ego vehicle's current position to the closest position on the trajectory Q_j.
$c_{4,j}$	1.0	Normalized difference in orientation between the ego vehicle's pose and the closest pose on the trajectory Q_j.
$c_{5,j}$	0.2	Normalized difference in velocity between the ego vehicle's pose and the closest pose on the trajectory Q_j.

Table 4.3: Criteria and weights for the heuristic evaluating the best target to follow. Except for $c_{1,j}$, all criterion values are normalized between all objects $\mathbf{o}_j \in O$ in the range $[0, 1]$ in a way that the respective state most similar to the ego vehicle's state is 1 and the one farthest away is 0.

After calculating and normalizing the values $c_{k,j}$, for each object $\mathbf{o}_j$ a weighted sum Ω_j is computed.

$$\Omega_j = \sum_k \omega_k c_{k,j}$$

All candidate objects $\mathbf{o}_j \in O$ are then stored in a priority list O_Ω, sorted by their according Ω_j. The object with the highest priority, i.e., the highest value Ω_j, is then selected as target vehicle o_t. As stated above, having a single target vehicle is important for adapting the velocity of the ego vehicle (compare Section 4.2.3), as well as the process of initializing the band described in Section 4.4. Having all candidates $\mathbf{o}_j$ stored in a priority queue enables easily selecting the next best target. This may be necessary, if for some reason no initial trajectory can be generated for the chosen o_t or if multiple redundant trajectories are planned for different target candidates.

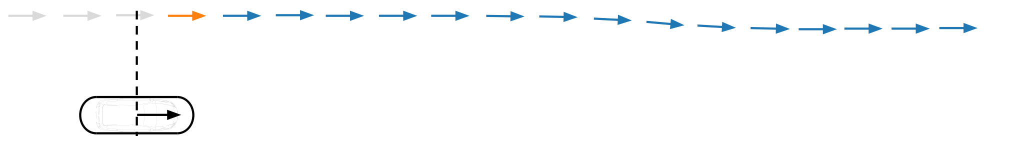
4.4 Initialization of the Elastic Band (CSTT)

One of the most critical factors for the performance of the elastic band approach is the initialization step. If the initial band (i.e., a sequence of initial poses) is based on a good estimate regarding the dynamic constraints for velocity and acceleration, the necessary number of iterations during the optimization step is greatly reduced (compare Section 5.3.2 for statistics). Furthermore, the initialization determines towards which local optimum the optimized band will converge. This, in turn, determines some properties of the optimized trajectory, e.g., if a specific obstacle is passed on the left or right. To provide the initial band $B_{init} = (Q_{init}, \Delta t)$ we have to find initial values for all poses $\mathbf{p}_i \in Q_{init}$.

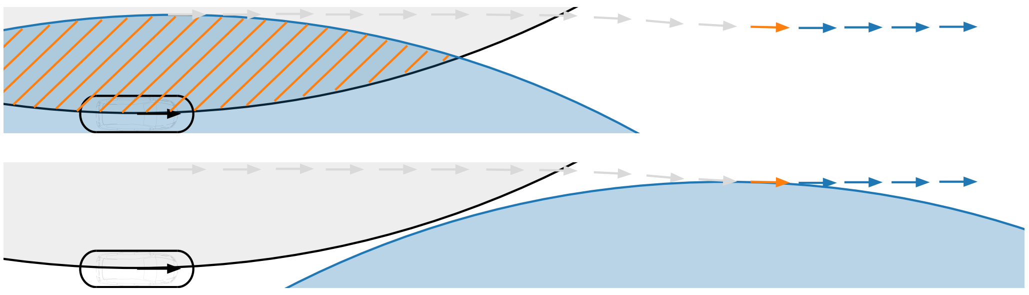
$$Q_{init} = (\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n)$$

The time interval Δt between subsequent poses is constant over all planning iterations and set explicitly to 2.0 s during the experiments throughout this thesis (compare Section 5.3.1). A simple solution - incorporating the vehicle's dynamics - is to sample positions along a straight path according to the ego vehicle's current velocity. However, the primary objective of the STEBLE approach is to follow the trajectories of other vehicles. Towards that goal, in the first experiments presented in [39] the path is aligned with the target vehicle's position: poses are sampled with a constant distance interval (based on the current velocity of the ego vehicle) from the start pose (i.e., the current position of the ego vehicle) to the goal (i.e., the latest predicted position of the target to follow, compare Section 4.3). The orientation for all poses is set to the orientation of the vector from start to goal. As shown in [39], the performance of the approach can be even more improved by providing a more sophisticated initial guess using cubic splines to sample the transitioning poses to the target trajectory, abbreviated CSTT.

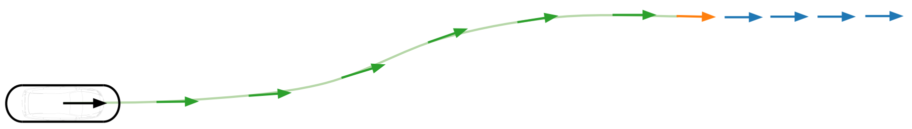
The CSTT initialization process presented in this thesis shares many features of the approach presented by the author of this thesis in [39], although it was improved in some aspects. E.g., the novel process does not take into account the current pose of the target vehicle but instead relies on the swarm-based prediction presented in Chapter 3. Also, cubic functions are constructed in the space domain in an intermediate step to prevent oscillation of the generated trajectories due to minor inaccuracies in the measurements. An overview of the process is shown in Figure 4.14.



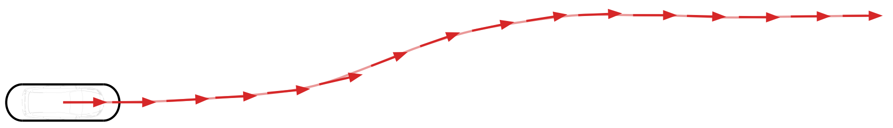
(a) The poses of the target vehicle's trajectory are evaluated in order from oldest to latest. The first encountered pose spatially in front of the ego vehicle (orange) and all later poses (blue) are kept for further processing. Older poses (gray) are discarded.



(b) For each remaining pose two minimum turning circles are constructed. One is attached to the ego vehicle's pose (black) and one to the target vehicle's respective pose (blue). All poses where the circles do intersect (area filled with orange lines) are discarded (gray). The first pose where circles do not intersect (orange) and all later poses (blue) are kept for further processing.



(c) A cubic function is constructed in the space domain for each of the two spatial coordinates of the position, using the current pose of the ego vehicle (black) and the oldest remaining pose of the target's trajectory (orange) as support points. The parameter of the funtions is the arc distance between those two poses. The resulting curves are then sampled for poses (green) using a constant step size (distance). The resulting equally spaced poses are associated with points in time related to the interpolated velocity between the two poses.



(d) A cubic spline is constructed in the time domain for each of the two spatial dimensions of the position, using the current pose of the ego vehicle, the sampled transitioning poses and the remaining poses of the target's trajectory as support points. The length of the splines is the time interval calculated from the velocity and points in time associated with the respective poses. The resulting spline is then sampled with a constant step size (time interval) to generate the poses of the initial band (red).

Figure 4.14: Overview of generating the initial elastic band.

Section 4.4.1 discusses the details of pruning the trajectory of the target vehicle, i.e., removing poses that cannot be reached, taking into account the ego vehicle's dynamics. Section 4.4.2 then elaborates on the process of sampling a trajectory with constant time intervals.

4.4.1 Pruning the Target Trajectory

The overall objective of the initialization process is to generate a trajectory smoothly transitioning from the current pose of the ego vehicle $\mathbf{p}_{ego} = (x_{ego}, y_{ego}, \theta_{ego})$ to the trajectory of a chosen target vehicle o_t (compare Section 4.3). The target's respective trajectory Q_t is obtained from the prediction modules presented in Chapter 3:

$$Q_t = (\mathbf{p}_{t,-n_t}, \ldots, \mathbf{p}_{t,0}, \mathbf{p}_{t,1}, \ldots, \mathbf{p}_{t,n+m})$$

It is represented as a sequence of poses $\mathbf{p}_i = (x_i, y_i, \theta_i)$, assuming a time interval of Δt between subsequent poses (in accordance with the Δt of the elastic band). The respective temporal offset of each pose $\mathbf{p}_i$ to the most recently observed pose $\mathbf{p}_0$ can be calculated by multiplying the index i with the time interval Δt.

The objective of the algorithm presented in this subsection is to prune the trajectory Q_t. In the resulting trajectory Q_t', only those poses should remain, which can be reached by the ego vehicle assuming moderate values for the centripetal and longitudinal acceleration. In general, poses that are not in the forward direction of the ego vehicle should be omitted, but only if they are within a certain distance. If they are far away, the ego vehicle should turn towards the target vehicle's trajectory. On the other hand, the ego vehicle should not be required to turn away from the target trajectory or drive in loops but instead prefer a direct transition, i.e., the distance to the trajectory should continuously decrease. This corresponds to a simple or S-shaped curve in the direction of the target's trajectory (compare Figure 4.15).

The algorithm for determining the final pruned trajectory Q_t'', i.e., deciding which poses of Q_t are feasible, is divided into two steps.

Step 1

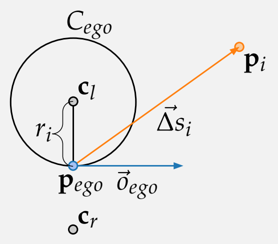
All poses $\mathbf{p}_i$ of the target vehicle's trajectory Q_t are evaluated in order from oldest to latest (i.e., starting with the lowest index i). For each pose $\mathbf{p}_i$ the dot product of the orientation vector of the ego vehicle and the difference vector of the positions of $\mathbf{p}_{ego}$ and the $\mathbf{p}_i$ is evaluated.
When encountering the first pose $\mathbf{p}_k$ spatially in front of the ego vehicle (i.e., the dot product is positive), an intermediate sequence Q'_t is constructed, including $\mathbf{p}_k$ and all later poses:

$$Q'_t = (\mathbf{p}_k, \dots, \mathbf{p}_{t,n+m})$$

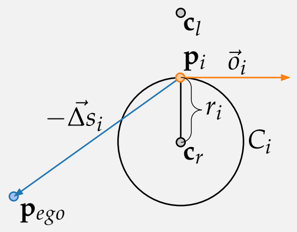
Step 2

All poses $\mathbf{p}_i \in Q'_t$ are evaluated in order, starting with the lowest index i.
For each pose $\mathbf{p}_i$ the feasibility is checked as followed:

- Get the distance d_i between the pose $\mathbf{p}_{ego}$ of the ego vehicle and $\mathbf{p}_i$. Calculate the velocity v_i the ego vehicle would have, when slowing down with longitudinal acceleration a_{long} from current velocity v_{ego} for distance d_i.
$$v_i = \begin{cases} \sqrt{v_i^2} & \text{if } v_i^2 \geq 0 \\ -\sqrt{-v_i^2} & \text{otherwise} \end{cases}, \; v_i^2 = -2a_{max}d_i + v_{ego}^2$$
Calculate the average velocity $v_{avg} = \max(0, \frac{v_{ego}+v_i}{2})$.
- Calculate the radius r_i of a minimum turning circle with centripetal acceleration a_{cen} with respect to v_i:
$$r_i = v_i^2 / a_{cen}$$
- Construct a minimum turning circle C_{ego} with radius r_i. The circle shall be tangential to the orientation vector $\vec{o}_{ego}$ of the pose $\mathbf{p}_{ego}$. Thus, there are two candidates for the position of the center of the circle: $\mathbf{c}_l$ and $\mathbf{c}_r$, orthogonal left and right of $\mathbf{p}_{ego}$, respectively.
Which of the candidates $\mathbf{c}_l$ and $\mathbf{c}_r$ is chosen as center point for C_{ego} depends on the relative position of $\mathbf{p}_i$. To determine if $\mathbf{p}_i$ is on the left or right side of $\vec{o}_{ego}$ (blue), the vector $\vec{\Delta s}_i$ (orange) is constructed, pointing from the position of $\mathbf{p}_{ego}$ to the position of $\mathbf{p}_i$. If the determinant of the matrix $[\vec{o}_{ego} \; \vec{\Delta s}_i]$ is positive, $\mathbf{p}_i$ is left of $\vec{o}_{ego}$ and $\mathbf{c}_l$ is selected as center point for C_{ego}. Otherwise, $\mathbf{c}_r$ is chosen accordingly.



- Construct a circle C_i, analogously to C_{ego}, but with $\mathbf{p}_i$ on the circle perimeter. Respectively, if the determinant of the matrix $[\vec{o}_i \; -\vec{\Delta s}_i]$ is positive, $\mathbf{p}_{ego}$ is left of $\vec{o}_i$ and $\mathbf{c}_l$ is selected as center point for C_i. Otherwise, $\mathbf{c}_r$ is chosen as center point accordingly.



- If the two circles C_{ego} and C_i do not intersect, the pose $\mathbf{p}_i$ is feasible. This means it can be reached from the ego vehicle's current pose, given the current velocity and respective acceleration limits below a_{long} and a_{cen}, without turning away from $\mathbf{p}_i$. Examples for feasible and non-feasible configurations of poses can be found in Figure 4.15.

When encountering the first pose $\mathbf{p}_k$ fulfilling the above criterion of feasibility, the sequence Q''_t is constructed, including $\mathbf{p}_k$ and all later poses:
$$Q''_t = (\mathbf{p}_k, \ldots, \mathbf{p}_{t,n+m})$$

The second step is based on the principle of Dubins path [51] using two circles and a tangent to connect two poses with the shortest curve.

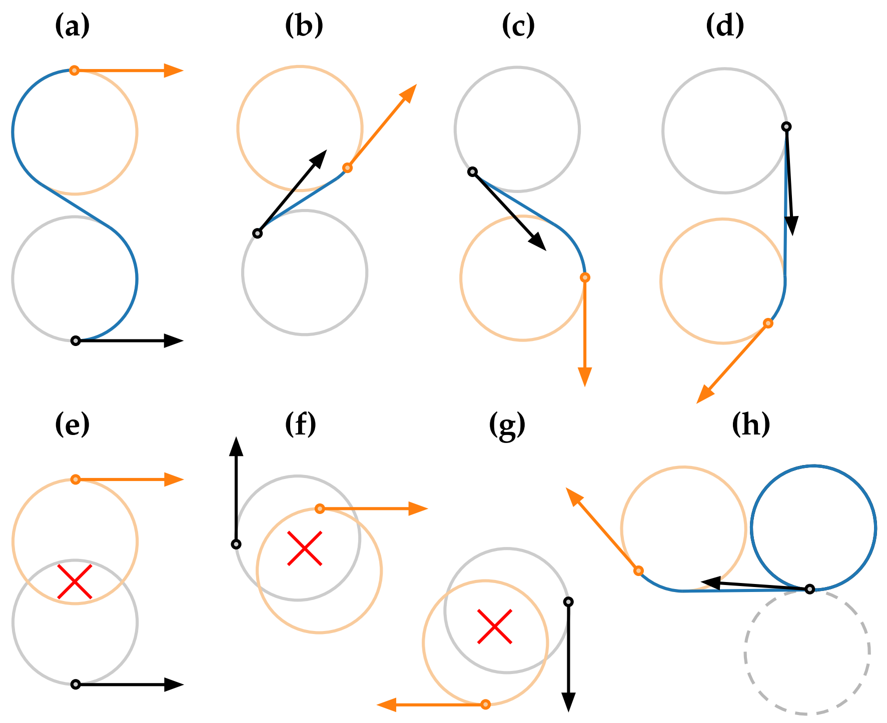
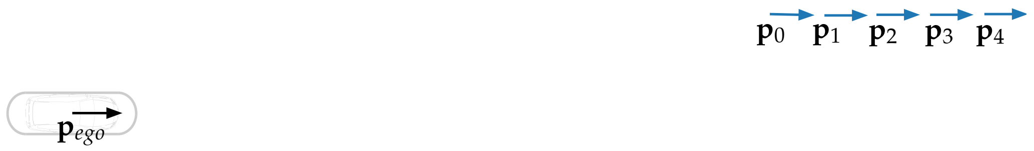


Figure 4.15: Examples for feasible and non-feasible configurations of poses. The pose of the ego vehicle is shown in black, the target pose in orange, the corresponding turning circles are shown in the respective lighter colors. The assumed trajectory transitioning between the poses is blue. **(a)-(d)** The two circles do not intersect, a direct transition without receding from the target pose is possible. The respective target poses are feasible. **(e)**,**(f)** The circles intersect, it would be required to initially move away from the target. The respective target poses are non-feasible and thus omitted from the filtered target trajectory. **(g)** A rare configuration, where the circles intersect, but a direct transition is possible. This false negative is accepted in favor of a more simple heuristic. **(h)** The circles do not intersect, but receding from the target would be required (if following the perimeter of the selected turning circles). This is not a problem, since for all similar configurations the second turning circle candidate (dotted gray) on the opposite side of the start pose can be constructed. This opposite circle also does not intersect with the target's circle and enables a direct transition on its perimeter.

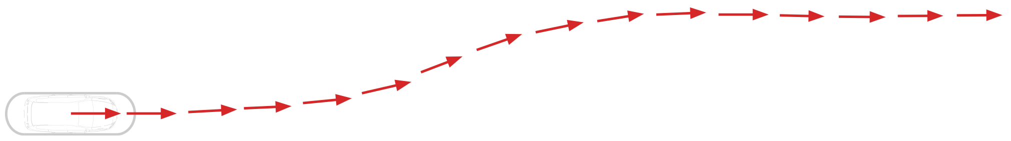
If there are no feasible poses, i.e., Q_t'' is empty, a new target vehicle o_t (i.e., a new trajectory Q_t) has to be chosen, as at least one pose is needed for the subsequent steps of sampling an initial trajectory. In this case, the next object o_i in the priority queue of evaluated target candidates (compare Section 4.3) is selected as o_t and the pruning step is repeated for the new target.

4.4.2 Sampling With Constant Time Intervals

A central aspect of the STEBLE approach is assuming a constant time interval Δt between adjacent poses of the planned trajectory. The trajectory Q_{init} used to initialize the elastic band should reflect this property. As stated above, the initial trajectory Q_{init} starts at the current pose of the ego vehicle and transitions smoothly to the trajectory Q_t of the selected target vehicle. However, for the reason of having enough distance (temporal and spatially) for a smooth transition, there may be a considerable gap between the start pose and the remaining poses of the pruned target trajectory Q_t'' (compare Section 4.4.1). Also, this gap usually does not correspond to a multiple of the constant time interval Δt associated with the elastic band. Thus, it is necessary to (re)sample the transitioning poses as well as the poses of the target trajectory with a constant time interval Δt in between.



(a) The input for the sampling step is the current pose of the ego vehicle $\mathbf{p}_{ego}$ (black) and the poses $\mathbf{p}_i$ (blue) of the pruned target trajectory Q_t''.



(b) The output is a sequence Q_{init} of poses (red), which is used to initialize the nodes of the elastic band. A constant time interval Δt is assumed between subsequent poses.

Figure 4.16: Input and output of the sampling step.

In previous work [40] an algorithm was proposed for this task, using a cubic spline (satisfying continuity requirements up to the second derivative) constructed for each of the two spatial dimensions over time. For this, it is necessary to estimate the time interval between the current pose $\mathbf{p}_{ego}$ of the ego vehicle and the first feasible pose $\mathbf{p}_0$ of the pruned target trajectory Q_t''. In [40] this time interval is derived directly from the arc distance (i.e., the estimated travel distance) and the average velocity between those two poses.

The cubic splines are generated using the *ContinuousDerivatives* class from the *ECL Geometry Package for ROS* [45], which refers to [46] for the algorithm. The splines are of third-degree and satisfy C^2-continuity requirements. Each spline is constructed from two corresponding arrays describing values and parameters of the support points of the spline, as well as the values of the first derivative in the first and last support point (i.e., boundary constraints).

During the experiments presented in Chapter 5 a further intermediate step has proven beneficial: generating additional support points for the spline between the start pose $\mathbf{p}_{ego}$ and the first feasible target pose $\mathbf{p}_0$. The optimization of the elastic band with regard to the vehicle's dynamics tends to generate trajectories with a symmetric transitioning segment, i.e., the inflection point of the transition curve is located around its center. This is primarily due to the acceleration constraints smoothing differences in the velocities and penalizing curvature. However, the estimation of the time interval between $\mathbf{p}_{ego}$ and $\mathbf{p}_0$, as well as inaccuracies in the measurements, lead to a large variety in the spline's appearance in this transitioning segment (compare Figure 4.17). The introduction of more support points dramatically reduces the variation of the spline due to minor inconsistencies in the time stamps related to the observed poses. Moreover, when sampling the additional support points, the orientation of the first feasible target pose $\mathbf{p}_0$ can be taken into account. This mitigates oscillations of the final spline between the target trajectory's poses.

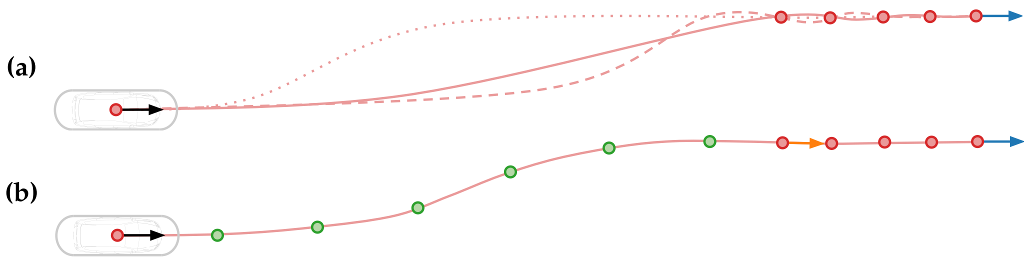


Figure 4.17: A cubic spline (light red) is constructed for each of the spatial dimensions over time to sample transitioning poses. **(a)** The start pose $\mathbf{p}_{ego}$ and feasible target poses $\mathbf{p}_i$ are used as support points (red). The value of the first derivative is set according to the corresponding velocities (black and blue arrows) in the first and last support point. The resulting spline's appearance varies (dashed, dotted and solid curve) with slightly different values for the time parameter of the support points. **(b)** Adding more support points (green) with a constant spatial distance considerably reduces the oscillations. The normalized orientation vector of the first feasible target pose (orange) is taken into account for sampling the additional points.

The result of the optimization of the elastic band is, in the vast majority, a trajectory with a spatially symmetric S-shaped transitioning segment. Thus, it makes sense to initialize the band with such a trajectory. For this, a cubic function (i.e., a polynomial of third-degree) is constructed for each of the two spatial coordinates of the path over the traveled distance (compare Figure 4.18). The functions are constrained by the coordinates of two support points: the start pose $\mathbf{p}_{ego}$ and the first feasible target pose $\mathbf{p}_0$. The orientation of the poses is taken into account by constraining the first derivative in those support points to values derived from normalized orientation vectors. That is, the amount of the velocity is completely ignored for the construction of the cubic functions. The domain of the function is the traveled distance.

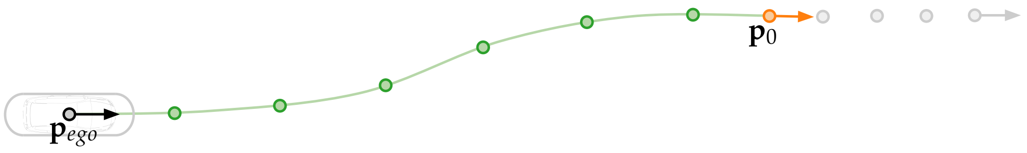


Figure 4.18: Generating additional support points to encourage a spatially symmetric transition segment. A cubic function (light green) is constructed for each of the two spatial coordinates over the traveled distance. The two functions are constrained by the respective coordinates of the start pose $\mathbf{p}_{ego}$ (black) and the first feasible target pose $\mathbf{p}_0$ (orange). Also, the values for the first derivative in the two boundary points are derived from the normalized vectors of the respective orientation (i.e., only the direction of the velocity is taken into account). The additional support points (green) are then sampled with a constant step size (with regard to the traveled distance).

Constructing the cubic function in the spatial domain is crucial in promoting the spatial symmetry of the transitioning curve. Deriving the values of the first derivative in both boundaries from unit vectors yields a perfectly symmetric curve if the orientation for both poses is the same. In practice, the orientation is at least similar in the vast majority of cases, which still reduces the oscillation of the final spline in the time domain considerably.

The domain of definition for the cubic function is the actually traveled distance. This distance is estimated by the arc distance between the two poses. Compared to the arc distance used for the velocity-related objective functions in Section 4.2.3, a slightly different definition of the arc distance is used. This is due to the fact that the two poses might be relatively far away (up to 70 m with a speed of 50 km/h and a planning horizon of 5 s) and often have a very similar orientation.

Thus, instead of assuming a single arc segment tangential to both poses' orientation vectors, the path is estimated by two symmetric arc segments. The combined length of those arc segments is equal to the length of one single arc segment tangential to the orientation vector of $\mathbf{p}_{ego}$ and touching $\mathbf{p}_0$ (compare Figure 4.19).

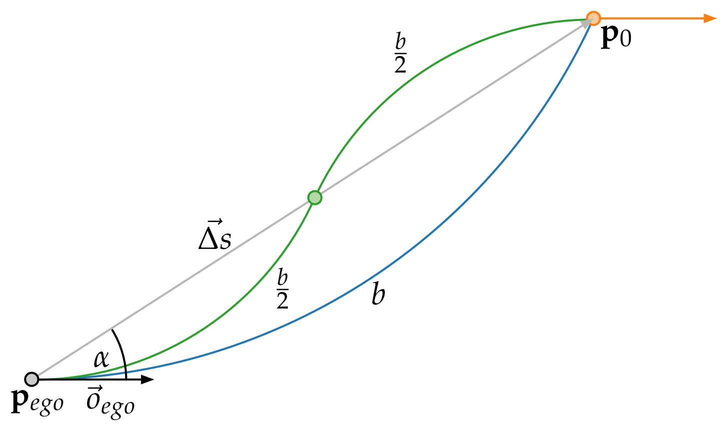


Figure 4.19: The arc length b of the blue arc segment is used for estimating the actually traveled distance between the poses $\mathbf{p}_{ego}$ and $\mathbf{p}_0$. It can be calculated from the angle α between the orientation vector $\vec{o}_{ego}$ and the positions' difference vector $\vec{\Delta s}$. The length of the two green arc segments (which resemble a symmetric trajectory assuming uniform orientation in both poses) is exactly $b/2$.

To calculate the arc length b of this single arc segment two values are needed: First, the norm of the difference vector $\vec{\Delta s}$ of the two poses' positions (i.e., the chord length of the arc segment). Second, the angle α between $\vec{\Delta s}$ and the orientation vector $\vec{o}_{ego}$.

$$b = \begin{cases} \left| \frac{\alpha \cdot |\vec{\Delta s}|}{sin(\alpha)} \right| & \text{if } \alpha \neq 0 \\ |\vec{\Delta s}| & \text{otherwise} \end{cases}$$

The cubic function constructed from the respective coordinates and first derivative of the boundary points $\mathbf{p}_{ego}$ and $\mathbf{p}_0$ over the estimated traveled distance b is then sampled for poses $\mathbf{p}'_i$ with a constant step size of 1 m. The resulting sequence of poses is not associated with a constant time interval. Instead, the time domain is reintroduced by assigning points in time to the spatially equidistant sampled points in space. The point in time t'_i associated with each pose $\mathbf{p}'_i$ is calculated from the speed value v'_i, which is interpolated linearly over the traveled distance between the ego vehicles speed v_{ego} and the speed v_0 in the pose $\mathbf{p}_0$.

$$v'_i = (1 - \frac{i}{b})v_{ego} + \frac{i}{b}v_0 \text{ (with } i < b)$$
$$t'_0 = \frac{1}{v'_0},\ t'_i = t'_{i-1} + \frac{1}{v'_i},$$

The point in time t_0 associated with the first feasible target pose $\mathbf{p}_0$ is calculated from the traveled distance b of the transitioning segment and the average velocity v_b between $\mathbf{p}_{ego}$ and $\mathbf{p}_0$. The time stamps t_i of the subsequent poses $\mathbf{p}_i$ are then computed based on their index i, as the respective poses are associated with the constant time interval Δt.

$$v_b = \frac{v_{ego} + v_0}{2}$$
$$t_0 = \frac{b}{v_b},\ t_i = t_0 + i\Delta t$$

The start pose $\mathbf{p}_{ego}$ is associated with origin of the time frame of the trajectory.

$$t_{ego} = 0\,\mathrm{s}$$

To obtain the final initial sequence of poses, a pair of cubic splines $p_x(t)$ and $p_y(t)$ (one for each spatial coordinate) is then constructed in the time domain.

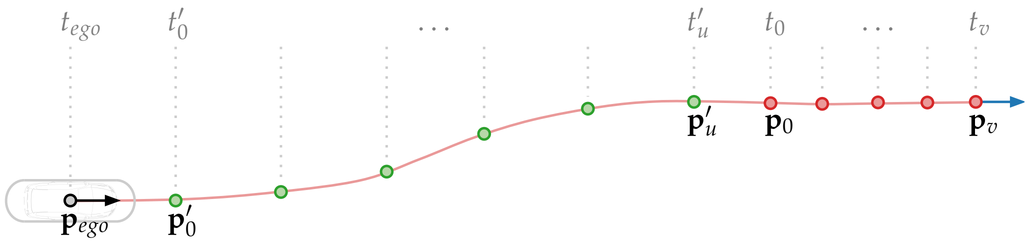


Figure 4.20: The poses of the initial trajectory Q_{init} are sampled from two cubic splines, representing the spatial coordinates in the time domain. The start pose $\mathbf{p}_{ego}$ (black), the transitioning poses $\mathbf{p}'_i$ (green) and the feasible target poses $\mathbf{p}_i$ (red) are associated with respective points in time (gray) to construct the support points of the splines. The first derivative in the splines' boundaries is taken from the respective velocity vectors (black and blue arrows).

For this, the start pose $\mathbf{p}_{ego}$, the transitioning poses $\mathbf{p}'_i$ and the feasible target poses $\mathbf{p}_i$ are merged in a sequence P.

$$P = (\mathbf{p}_{ego}, \mathbf{p}'_0, \dots, \mathbf{p}'_u, \mathbf{p}_0, \dots, \mathbf{p}_v)$$

A second sequence T is constructed, containing the respective points in time t'_i and t_i, relative to the time stamp $t_{ego} = 0$.

$$T = (t_{ego}, t'_0, \dots, t'_u, t_0, \dots, t_v)$$

The sequences P and T represent the support points of the pair of cubic splines $p_x(t)$ and $p_y(t)$. The respective spatial coordinate of the poses in P is the value, the corresponding entry of T is the parameters. The boundary constraints, i.e., the values of the first derivative in the first and last support point, are taken directly from the components of the respective velocity vectors $\vec{v}_{ego}$ and $\vec{v}_v$ in the first and last pose of P. As stated above, the splines are continuous in the first and second derivative by construction.

The sequence of initial poses Q_{init} is then obtained by sampling initial poses $\mathbf{p}_i = (x_i, y_i, \theta_i)$ from $p_x(t)$ and $p_y(t)$ with a step size of Δt. The values for x_i and y_i are the values of the respective coordinate's spline at the parameter $i\Delta t$. The orientation angle θ_i can be calculated in a straight-forward manner from the first derivatives $p'_x(t)$ and $p'_y(t)$ of the splines, as those can be interpreted as the components of the respective pose's velocity vector.

$$\begin{aligned} \theta_i &= \text{atan2}(p'_y(i\Delta t), p'_x(i\Delta t)) \\ \mathbf{p}_i &= (p_x(i\Delta t), p_y(i\Delta t), \theta_i) \\ Q_{init} &= (\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_n) \\ B_{init} &= (Q_{init}, \Delta t) \end{aligned}$$

The temporal horizon of the predicted target trajectory extends the duration of the elastic band (compare Chapter 3). Thus, the domain of the spline is larger than $n\Delta t$ in most cases (i.e., if the ego vehicle does not move substantially faster than the target vehicle), so that all initial poses up to $\mathbf{p}_n$ can be sampled from the splines. If this is not the case, the remaining poses are computed assuming constant linear and angular velocity.

4.5 Validation and Pruning of the Optimized Elastic Band

An inherent problem of the STEBLE approach is that for some configurations of the poses, the different objectives of the optimization may oppose each other. Due to these conflicting objectives, the elastic band may get stuck in invalid states during the optimization process, i.e., the dynamics of the trajectory exceed the given limits or the minimum distance to objects is violated. If this is the case, one of the objectives has to be disregarded, i.e., one constraint is violated. This drawback is reinforced by the fact that the optimization can only converge towards a local optimum. An existing globally better solution may not be found.

This violation of the objectives is accepted within a certain limit, as chosen thresholds do not represent hard constraints. In some cases, it is actually beneficial to exceed the thresholds temporarily to enable convergence towards a globally better solution. On the other hand, the dynamics of vehicles have hard constraints in the real world, which cannot be exceeded. If the final trajectory violates those limits, it may be impossible for the ego vehicle to follow the planned trajectory or the risk of collisions is increased unacceptably. Thus, the final trajectory needs to be checked for validity (i.e., if there are no violations of the "hard" constraints) before it is passed to the vehicle controller.

As stated above, the optimization may get stuck in an invalid state because of opposing objectives. The objective function pushing the trajectory away from objects is always involved in those gridlocks, as it directly constrains the location of the trajectories poses. Enforcing other objectives, such as maximum acceleration or velocity, may require the poses to be in such restricted areas. Two simple cases are depicted in Figure 4.21. There are much more complicated configurations, where more than two objectives are involved: Avoiding obstacles in combinations with any number of the safety-related functions preserving the non-holonomic kinetics, minimum turning circle, maximum acceleration, or velocity may lead to a gridlock situation. It has to be noted that the objective function attracting the trajectory to the other vehicles' trajectories also enforces a direct spatial restriction. However, it is weighted much lower and thus does not cause the problems mentioned above.

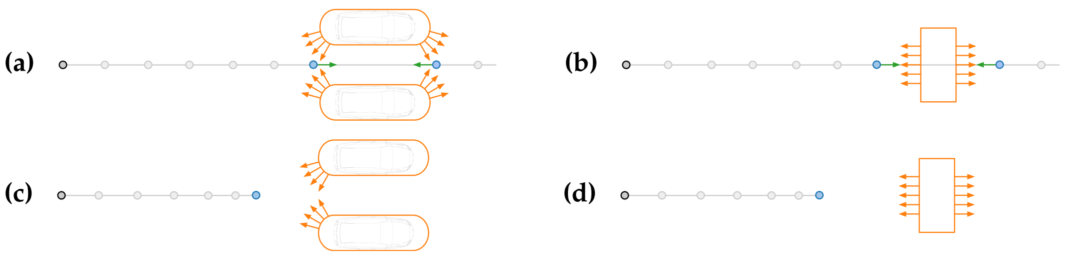


Figure 4.21: (a)/(b) Examples of elastic bands stuck in an invalid state due to opposing objectives. The orange obstacles enclose the trajectory on both sides or are located on the trajectory with a perpendicular alignment, respectively. The objective function repels the blues poses from the obstacles (orange arrows). They are also attracted to each other to preserve max acceleration and velocity limits (green arrows). The blue poses do not move during further optimization iterations, as the repelling and the attracting forces are in balance. The stuck poses prevent the trajectory from slowing down to avoid a collision, as for such a maneuver, it would be necessary to pull more poses closer towards the trajectory's start pose (black). **(c)/(d)** By pruning all poses after the first stuck pose, the remaining poses are freed from the gridlock, as one of the opposing objectives is removed.

Checking the trajectory for validity is done in a straightforward manner: The minimum distance to objects, the curvature, the velocity, and the longitudinal, angular, and centripetal acceleration at each pose (or pair/triple of poses, respectively) are compared to thresholds representing the hard constraints. The actual values of the thresholds can be found in Table 5.2 in Chapter 5. The values at the respective poses do not need to be calculated again, as they are stored in the edges of the optimization graph. Compare Section 4.2 for a detailed description on how those values are calculated.

In most cases, the gridlock of the objectives only affects a minimal number of poses directly. If the constraints on those poses are removed (or the poses themselves), the elastic band recovers with further optimization iterations and converges to a valid state again. Thus, instead of throwing away the whole invalid trajectory, only poses directly involved in violating the given hard limits are pruned from the elastic band. This has the advantage of still providing a valid trajectory in many cases, although this pruned trajectory may be much shorter than the usual planning horizon. This is not necessarily a problem in the context of the framework used for this thesis, as the time interval for re-planning is much shorter than the planning horizon. Nonetheless, a shortened horizon cripples the planning algorithms' ability to plan ahead.

In practice, it is not necessary to prune all poses contributing to an invalid state, but only the latest pose is involved in calculating the respective values. Thus, the following rules are applied to decide where the elastic band is clipped: (1) For violations of the minimum distance to objects, all poses after the first invalid one are removed. (2) If the limits for maximum velocity or centripetal acceleration are exceeded, the latter of the two poses involved in the calculation and all subsequent poses are pruned. The same applies to violations of the minimum turning circle. (3) The longitudinal and angular acceleration take three poses into account; the latest of those and all subsequent poses are removed, respectively.

The check for validity is not performed after every iteration of the optimization to give the optimization process some time to recover to a valid state (and reduce the impact on the computational cost). Instead, the iterations are partitioned into an inner and outer loop. i.e., the optimization is performed in batches. While the range of the inner loop represents the actual iterations of the graph optimization, the range of the outer loop defines how many of those optimization batches are executed. For the experiments in Chapter 5 the number of iterations was set to 10 for the inner loop and 4 for the outer loop, respectively (compare Section 5.3.1). In Section 5.3 the impact of the pruning on the trajectories is evaluated in detail.

4.6 Simultaneous Optimization of Multiple Trajectories

Due to the process of pruning invalid trajectories, the STEBLE approach may return a result with zero length. Even if the horizon of a valid trajectory extends the time interval for re-planning (i.e., 0.2 s in the presented experiments), it may be too short to enable anticipatory driving. This may be inevitable in some cases, but in other cases, a globally better solution exists. This inherent problem of local search algorithms can be dealt with in several ways. A straightforward technique to enhance the chances of finding a globally optimal solution is to start the optimization with different initial values. In the case of STEBLE, this is realized by planning multiple trajectories simultaneously.

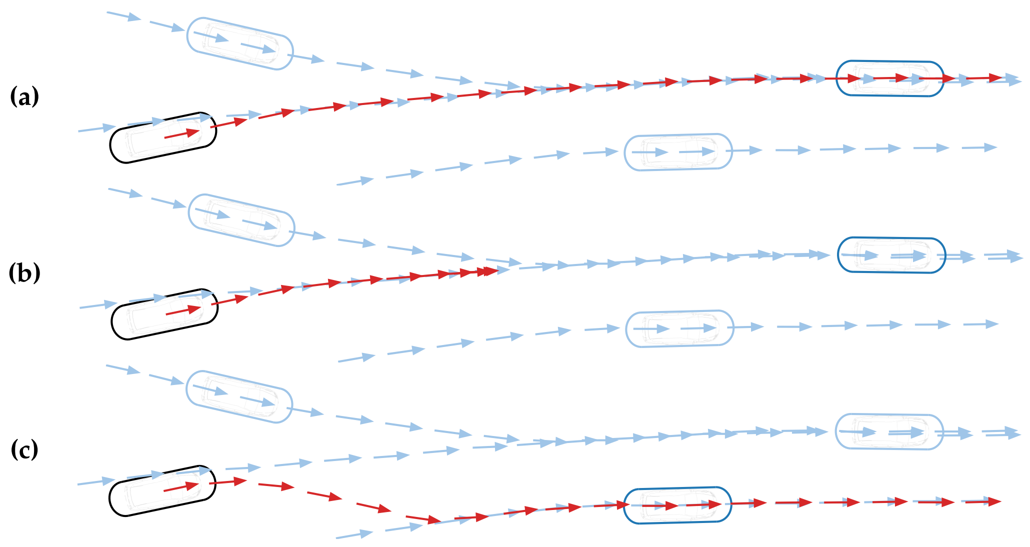


Figure 4.22: Candidates for initial trajectories. **(a)** The initial trajectory (red) is based on the heuristically best target vehicle's trajectory. **(b)** The trajectory's path is based on the heuristically best target's path, but braking with maximum longitudinal acceleration is assumed. **(c)** The initial trajectory is based on the heuristically second best target vehicle's trajectory.

The STEBLE approach uses two ways of providing different initial values (i.e., initial trajectories), depicted in Figure 4.22: First, adapting the velocity on the path following the heuristically best target to follow. Second, choosing a different target to follow.

Both strategies can be implemented in a straightforward manner. The poses of the initial trajectories are sampled from a cubic spline in the time domain (compare Section 4.4.2), adapting the velocity is performed by sampling with adjusted time intervals based on the euclidean distance. For the value of the adapted velocity, an emergency braking maneuver is assumed, i.e., braking with the maximum possible negative longitudinal acceleration of $8\,\mathrm{m/s^2}$. Planning with a different target vehicle is computationally more costly since it requires rerunning the whole process of pruning the target trajectory and constructing a spline in the time domain. On the other hand, some aspects can be shared between the different passes. Most importantly, the dynamic objects are predicted independently from the actual target vehicle (compare Chapter 3). Also, the process of selecting the heuristically best target provides a priority list of candidates, so choosing the next best target is straightforward (compare Section 4.3).

Another essential task when simultaneously planning multiple trajectories is to select the best trajectory for passing on to the software modules controlling the vehicle's actuators. As all candidates are checked for validity and pruned accordingly (compare Section 4.5), the final optimized trajectories are all driveable and collision-free.

An obvious candidate for evaluating the trajectories is the value of the weighted multi-objective function $f(B)$, i.e., the sum of all edges of the graph underlying the optimization process (compare Section 4.2). Since $f(B)$ is the basis for the optimization, lower values represent more optimal solutions. However, this value is not a good choice when comparing trajectories that are already guaranteed to be safe and driveable.

The multi-objective function $f(B)$'s thresholds and weights are tuned to emphasize the properties of the trajectory related to the safety aspect. Even slight violations of the respective thresholds may result in large penalties due to the relatively higher weights. If the safety-related limits are already respected (which is guaranteed by the validation and pruning step), there is no more need to penalize any violations of the respective thresholds. Also, the objective functions for optimal velocity and path following are not directly related to a trajectory's safety and comfort but contribute to following a specific vehicle's paths at a specific distance. Instead, the comfort-related aspects are much more relevant when comparing trajectories that are already guaranteed to be driveable and safe. Those can be reduced to one defining value: the amount of acceleration on the trajectory. It has to be noted that it is not differentiated between negative or positive longitudinal acceleration and centripetal acceleration in this regard. The value used for the comparison is the norm of the vector $\vec{a}$, which represents the combined acceleration in all directions.

Another important aspect when comparing trajectories is the locality of the criteria. Taking into account only an average value on all poses may hide some uncomfortable sections. E.g., an otherwise perfectly smooth trajectory with one large spike in the acceleration values may get evaluated better than a slightly rougher but more balanced trajectory. In the context of both safety and comfort, it is often more meaningful to compare the maximum value of the trajectories' properties than an average.

Another aspect of the trajectories' quality is the (temporal) length: longer trajectories enable more anticipatory driving. As the trajectories may be pruned during optimization to ensure driveability and safety, the total duration ΔT_B of the elastic band B is also used as a criterion.

Furthermore, another temporal element is taken into account: the duration the target vehicle used to initialize the respective elastic band was followed. This is essentially a hysteresis on the target vehicle. It is implemented to prevent frequent changes of the target vehicle in the case of two trajectories with similar acceleration values and length, as those changes often introduce unwanted swerving. This swerving feels uncomfortable even when it is not directly increasing the acceleration.

The above mentioned criteria, i.e., (1) the maximum $|\vec{a}|_{max}$ of the norms of the combined acceleration vectors $\vec{a}_i$ at each pose (in m/s^2), (2) the respective average $|\vec{a}|_{avg}$ over all poses, (3) the total temporal length ΔT_B of the band (in s) and (4) the duration ΔT_t the respective target vehicle was followed in previous planning iterations (in s), are combined in the cost function $c(B)$.

$$\begin{array}{rl|l} c(B) = & |\vec{a}|_{max} & \text{maximum acceleration in range [0,8]} \\ & + |\vec{a}|_{avg} & \text{average acceleration in range [0,8]} \\ & + 0.1 \max(5 - \Delta T_B, 0) & \text{total trajectory duration clipped to [0,0.5]} \\ & + 0.5 \max(1 - \Delta T_t, 0) & \text{duration target followed clipped to [0,0.5]} \end{array} \tag{4.18}$$

Each addend of $c(B)$ is limited from below and above, either by the validation step restricting the respective value during optimization or by the max operator. The individual range of each addend is annotated in Equation 4.18. As can be seen, the acceleration's expressions are most highly weighted; the expressions related to the length of the band and the target hysteresis have much lower weights.

All acceleration values are already computed for each pose (or pair/triple of poses, respectively) on the band in the previous processing step of validation and pruning. The respective minima and maxima for each band B are stored at the same time. The total duration $\Delta T_B = n_B \Delta t$ of the band B can be derived directly from the respective number of poses n_B. The duration $\Delta T_{followed}$ represents the time interval how long the target vehicle forming the basis of the initial trajectory of B was followed previously. It is tracked over subsequent planning iterations based on the object id reported by the object tracking modules.

Selecting the most comfortable of the simultaneously optimized elastic bands is the final step in the proposed trajectory planning algorithm. After calculating the value of $c(B)$ for each candidate, the trajectory of the elastic band with the lowest value is passed to the further processing modules. It has to be noted, there still may be no valid trajectory at all, e.g., if a collision cannot be avoided while still respecting the given limits on the dynamics. This may happen if an object pops up on the planned path close to the ego vehicle. If no candidate with a non-zero length can be found, an empty trajectory is passed to signal this particular case. The empty trajectory can be easily detected and handled appropriately by introducing emergency maneuvers. However, the experiments presented in the following chapter show that this situation is very unlikely in typical traffic scenarios. As stated above, planning multiple trajectories significantly enhances the probability of finding a globally optimal solution with a sufficiently large planning horizon for anticipatory driving maneuvers.

Chapter 5

Evaluation and Experimental Results

This chapter evaluates the proposed swarm-based prediction and trajectory planning algorithms and presents experimental results. Section 5.1 describes the data sets used for the evaluation. Section 5.2 presents statistics on the swarm-based prediction, while Section 5.3 focuses on the STEBLE trajectory planning. Regarding STEBLE, Section 5.3.1 gives an overview of the parameters, weights, and thresholds used in the experiments presented in this chapter to adapt to passenger preferences and physical limitations of the vehicle. Section 5.3.2 gives a statistical analyses of the objective function for trajectory optimization. In 5.3.3 the general validity of the approach is discussed, Section 5.3.4 evaluates the accuracy of the path following. In Section 5.3.5 the trajectory generated by STEBLE are compared to the trajectories of human drivers.

5.1 Description of the Evaluated Data Sets

This section gives an overview of the data sets used for evaluating the proposed STEBLE path planning algorithm. The data was obtained in two different setups: One is a pure simulation approach, used for data set A (compare Section 5.1.1). In the other setup, the simulation environment is combined with recorded live data. This second approach is used for data set B and C (Section 5.1.2 and 5.1.3, respectively).

In the pure simulation environment of data set A, the poses of the ego vehicle and all other objects (i.e., other vehicles) are computed over time by the simulation modules of the FUB_ROSCAR framework. The physical model for updating the poses is based on the observed dynamics of the autonomous vehicle MadeInGermany but disregards complex physical aspects such as inertia and friction. The offset between two subsequent simulation steps is based on the control output of the vehicles' respective controller modules and its linear and angular velocity (compare Section 2.3.2). A pure simulation environment has the advantage that the trajectory of the ego vehicle and the trajectories of all other vehicles are known with perfect accuracy. This setup enables evaluating the path planning algorithm without being affected by the precision of the environment perception (such as sensor accuracy and timing issues). Also, the behavior of the other traffic participants can be directly controlled. This is especially useful for enforcing actions that do not frequently occur in real traffic, such as emergency braking maneuvers.

For the second approach, which is used in data set B and C, the simulation of the ego vehicle is combined with recorded data of the environment. The poses of the ego vehicle are computed in the same way (i.e., from a simulation model) as in the pure simulation environment. However, the information on observed objects is taken from data sets recorded during test driving in real traffic with the autonomous vehicle MadeInGermany. Of course, the simulated pose of the ego vehicle can deviate from the recorded original pose if the parameters of the trajectory planning have changed. Thus, there may be an offset between the simulated pose and the origin of the recorded data. This offset may affect the authenticity of the recorded data; different objects or parts of objects may be visible in the recorded data due to occlusion. To mitigate this effect, the simulated pose of the ego vehicle is reset to the vehicle's original recorded pose if the respective positions deviate more than a specified threshold. For data sets B and C, this threshold was set to a value of 20 m and 30 m, respectively. The combined simulation and recorded data have the advantage of providing a much more realistic impression of the natural environment and the performance of the sensors (compared to pure simulation), including sensor accuracy, occlusion, and delay. However, the combination with a simulated ego vehicle still enables repetitions of the same scenario with different planning parameters.

It has to be noted that there have also been several experiments in a pure live environment, i.e., the STEBLE trajectory planning modules were running on the autonomous vehicle MadeInGermany and providing the basis for the control of the vehicle. However, due to the computational cost of live sensor processing and data recording (mandatory for live test drives), detailed statistics on the generated trajectories had to be collected later on the respective recorded data (using the same parameters for the planning). This resembles the combined approach with a deviation threshold of zero. As stated above, of the three different data sets used to evaluate the STEBLE approach, only one (A) uses the pure simulation approach. The other data sets (B and C) are based on recorded data from different test drives with the autonomous vehicle MadeInGermany. All three data sets are described in detail in the following subsections.

5.1.1 Data Set A

Data set A is a round course of ∼600 m length. It was initially designed for tuning the controller modules of the FUB_ROSCAR framework and is located on the apron of the decommissioned Berlin Tempelhof Airport. It has three lanes in all sections of the course, although the lanes are purely virtual, i.e., no road markings or other boundaries indicate the course in the real world.

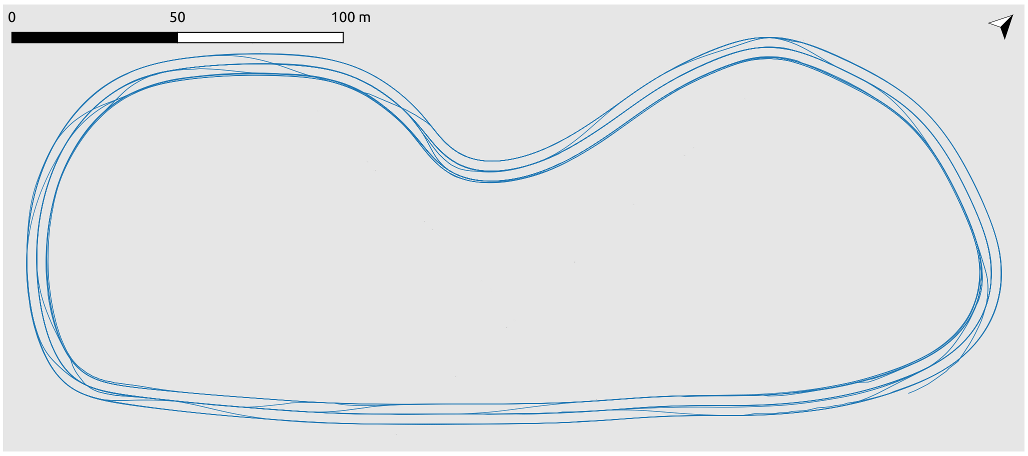


Figure 5.1: The GPS track of the (simulated) ego vehicle following other (simulated) vehicles autonomously for 60 minutes in Scenario A.

The course has sections of varying curvature and one long straight segment of ∼200 m length (compare Figure 5.1). The virtual track was populated with a simulated ego vehicle and ten simulated non-ego vehicles to accumulate data for evaluating the trajectory planning. The corresponding data set was recorded in a 60 minute run of the simulation.

In contrast to the ego vehicle (which used the STEBLE trajectory planning), the simulated non-ego vehicles followed the virtual lanes in a counterclockwise direction using a map-based trajectory planner. The vehicles were spawned at locations distributed randomly along the track; each simulated vehicle also has a randomly chosen preference for one of the three lanes. This lane preference was randomly altered at 10 Hz with a probability of $0.001\overline{6}$, i.e., on average, each vehicle performed a lane change maneuver onto a new preferred lane every 1.5 minutes.

To generate some variance in the driving style, a speed limit was set randomly for each non-ego vehicle between 36 and 54 km/h. Furthermore, the maximum centripetal acceleration was set to a value between 2 and $3\,\mathrm{m/s^2}$, the maximum positive acceleration between 0.6 and $0.8\,\mathrm{m/s^2}$ and the maximum negative acceleration was varied between 0.9 and $1.1\,\mathrm{m/s^2}$. Due to the restriction of the maximum acceleration, the speed limit is only reached on the straight segment. In the curvy parts of the track, the velocity of the simulated vehicles varies between 20 and 40 km/h.

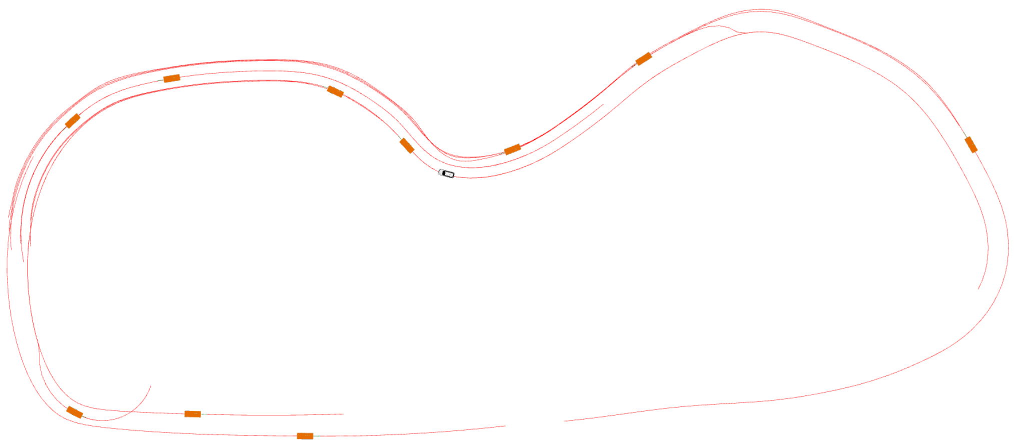


Figure 5.2: A typical scene of the object data recorded in Scenario A. The ego vehicle is shown in white. All simulated vehicles (orange) are visible to the ego vehicle, regardless of the distance or occlusion. Their trajectories (partly tracked and partly predicted) are shown in red. There are no other objects simulated except for the vehicles.

The object information (i.e., pose, velocity and dimensions) of all simulated vehicles is made available to the ego vehicle's trajectory planning modules without considering occlusion and sensor uncertainty. An exemplary scene of the object data (and the respective predicted trajectories) is shown in Figure 5.2. One significant advantage of the pure simulation approach of data set A is that the trajectories of all observed vehicles are known perfectly so that sensor uncertainty can be excluded from the evaluation. The curvy track, designed for tuning the control modules, also provides a challenge for the STEBLE approach due to the high and frequently changing curvature. Furthermore, the simulated vehicles' random lane changes and planning parameters generate a wide variety of different situations.

5.1.2 Data Set B

Data set B was accumulated in a simulation environment combined with recorded live data. The poses of the ego vehicle are computed in the same way as in the pure simulation approach: from a simulation model based on the control output. The control output, in turn, is based on the planned trajectory provided by the STEBLE trajectory planning modules. However, in contrast to the pure simulation, the information on observed objects is taken from a data set recorded during test driving in real traffic with the autonomous vehicle MadeInGermany. To preserve the authenticity of the recorded data in combination with the simulation, the pose of the simulated ego vehicle is reset to the recorded original pose if the two poses deviate more than 20 m.

The object data for data set B was recorded on a test drive in an especially challenging urban traffic environment: the roundabout “Großer Stern” in the center of Berlin, Germany (Figure 5.3). The roundabout connects five roads; the two northern roads each have four lanes, the three other roads have six lanes. Traffic lights control the entering traffic. The number of lanes within the roundabout varies between four and six.

The recorded data set consists of 15 transits of the roundabout. It was recorded on Thursday, January 4, 2018, at 10:54 am (UTC) and has a total length of $\sim$39 minutes. The traffic situation reflects an average morning (after rush hour, but still crowded) on a working day. Samples of the observed objects are shown in Figure 5.4.

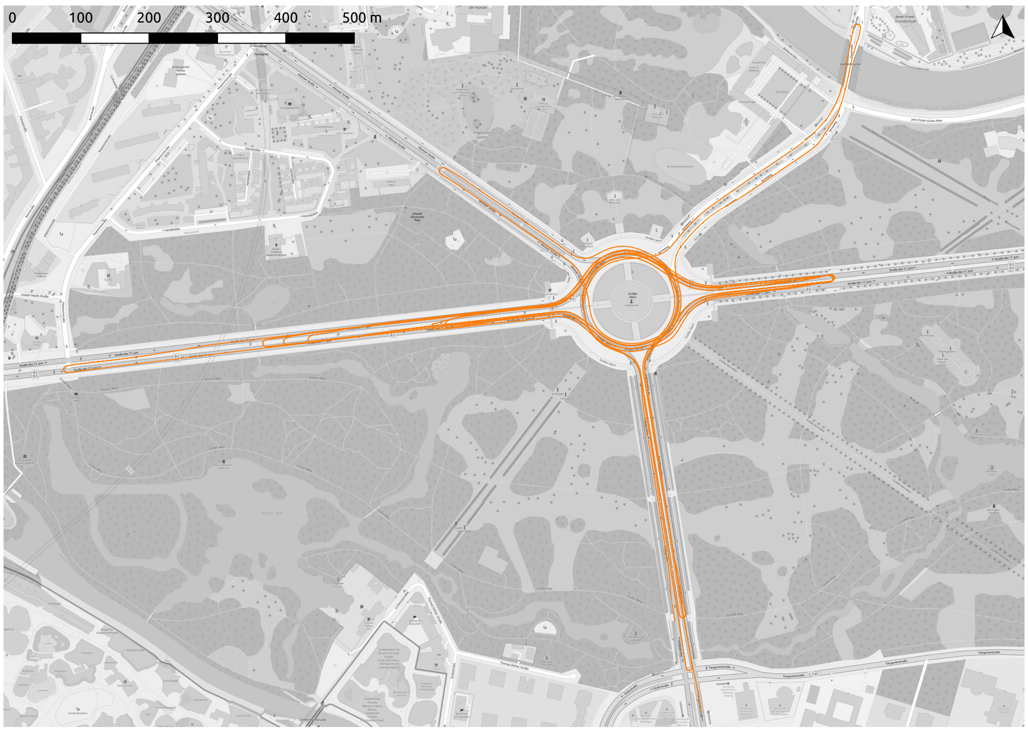


Figure 5.3: The recorded GPS track of the ego vehicle in Scenario B.

Over the whole data set, the ego vehicle perceived at least two other vehicles at all times. A maximum of 18 vehicles was observed simultaneously; on average, the number of perceived other vehicles is six. The speed limit for the whole roundabout and its surrounding roads is 50 km/h. The human drivers transiting the roundabout usually do not exceed this limit due to the high curvature. On the other hand, they do not fall below this limit significantly, leading to relatively high centripetal acceleration values. The human driver of the ego vehicle tried to blend in with the traffic as much as possible (excluding the unusual U-turns to re-enter the roundabout). More statistics on the behavior of the driver in this scenario can be found in Section 5.3.5.

The traffic environment of data set B is a massive challenge for the STEBLE trajectory planning, as it has vehicles changing lanes, sudden changes in the traffic flow due to traffic lights, and a very high curvature in relation to the speed of the other vehicles.

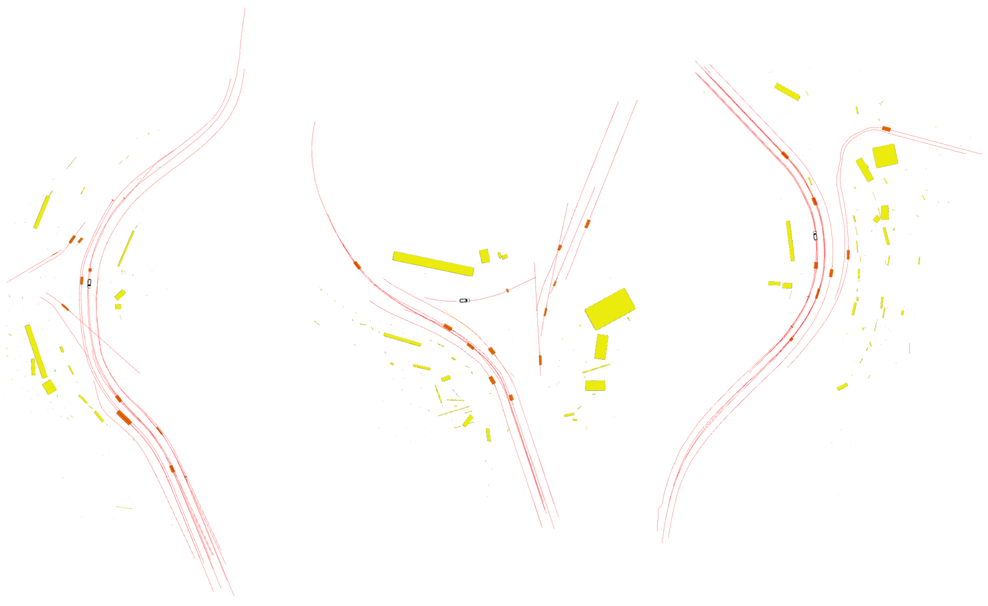


Figure 5.4: Samples of the object data recorded for data set B. The ego vehicle is shown in white. Objects classified as vehicles are shown in orange, their trajectories (partly tracked and partly predicted) red. Static objects are yellow.

5.1.3 Data Set C

Data set C was also accumulated in the setup combining simulation and recorded live data of the autonomous vehicle MadeInGermany. In contrast to data set B, which reflects an urban traffic environment, data set C comprises a highway-only test drive. Another major difference is that a considerable part of the test drive was conducted in autonomous mode, i.e., the control of the vehicle was handed over to the FUB_ROSCAR framework (using the STEBLE trajectory planning).

The underlying data for the objects and original pose of data set C was recorded on Thursday, November 22, 2018, at 9:12 am (UTC). The record has a duration of ∼12 minutes, a total distance of 13.1 km is traveled, a distance of 6.7 km is traveled in autonomous mode. The corresponding GPS track is depicted in Figure 5.5.

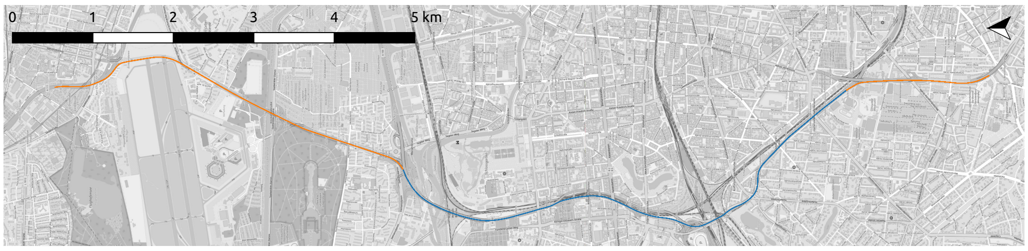
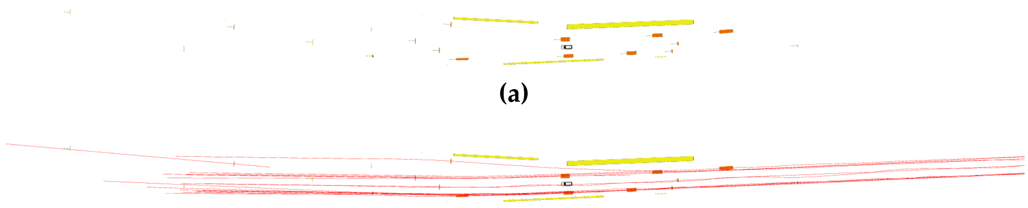


Figure 5.5: The recorded GPS track of the ego vehicle in data set C. In the orange segments of the track, the ego vehicle was controlled by a human driver. In the blue segment, the ego vehicle was driving in autonomous mode, using the STEBLE trajectory planning to follow the trajectories of observed vehicles.

The record starts with the human driver controlling the vehicle on highway A 111 in the southern direction, close to exit "Seidelstraße". When reaching highway A 100, the control of the vehicle was handed over to the FUB_ROSCAR framework. The ego vehicle then followed the other vehicles to the exit "Schmargendorf" (using the STEBLE trajectory planning), where the human driver took over control and exited the A 100, following the "Abzweig Steglitz".



(b)

Figure 5.6: A sample of the object data recorded for data set C. The ego vehicle is shown in white. Objects classified as vehicles are shown in orange, static objects are yellow. In **(a)** trajectories are omitted for improved visibility of the objects, **(b)** depicts the same scene with the tracked and predicted trajectories (red) of the observed objects.

The traffic situation in data set C reflects an average morning on a working day. Samples of the observed objects are shown in Figure 5.6. Despite a speed limit of 80 km/h throughout the whole track, the actual speed of the observed vehicles is much lower due to the heavy traffic. The average speed of the ego vehicle is 64.076 km/h. The human-driven segments of the track have two lanes in the driving direction (excluding exit and entry lanes), the autonomous segment has three lanes. The oncoming traffic is separated clearly by solid walls or fences at all times. Over the whole data set, the ego vehicle perceived at least three other vehicles at all times (five in the autonomous segment). A maximum of 19 vehicles was observed simultaneously; on average, the number of perceived other vehicles is five (seven in the autonomous segment). The highway environment of data set C (with no traffic lights and sudden changes in the curvature) suits the STEBLE trajectory planning. On the other hand, heavy traffic still provides a major challenge, with many vehicles cutting in at significantly varying speeds.

5.2 Evaluation of the Swarm-Based Prediction

The prediction of the movement of dynamic objects is an essential requirement for the STEBLE approach. The concept of following other vehicles benefits significantly from a stable and accurate guess on their future trajectories. For STEBLE, the predicted trajectories are used in two different domains: the avoidance of dynamic objects (Section 4.2.4) and the initialization of the elastic band along a target trajectory (Section 4.4). The performance of the swarm-based prediction, explained in detail in Chapter 3, is evaluated for all three data sets described in Section 5.1: (A) a small round course with heavily varying curvature, (B) an urban roundabout with several lanes, and (C) a highway environment. The swarm-based approach is compared to a simple prediction model, which assumes constant linear and angular velocities and was used in an early implementation of STEBLE (compare [39] for details).

To evaluate the prediction accuracy, the predicted position and speed are compared to the actual state of the object at the respective points in time. This comparison is conducted for different temporal horizons of the prediction; the plots show the results for 1, 2, 3, 4, and 5 seconds. The results for all horizons are only considered for the evaluation if the object (identified by a unique tracking id) is observed in the recorded data set at each respective point in time. While for data set A, all objects' actual position and speed are known with perfect accuracy (due to the pure simulation setup), data sets B and C include the sensor uncertainty in the measurements. The results of the evaluation are shown in Figure 5.7 and Figure 5.8 for the position and speed, respectively. The plots show the resulting error in the prediction for the simple velocity-based and the proposed swarm-based approach.

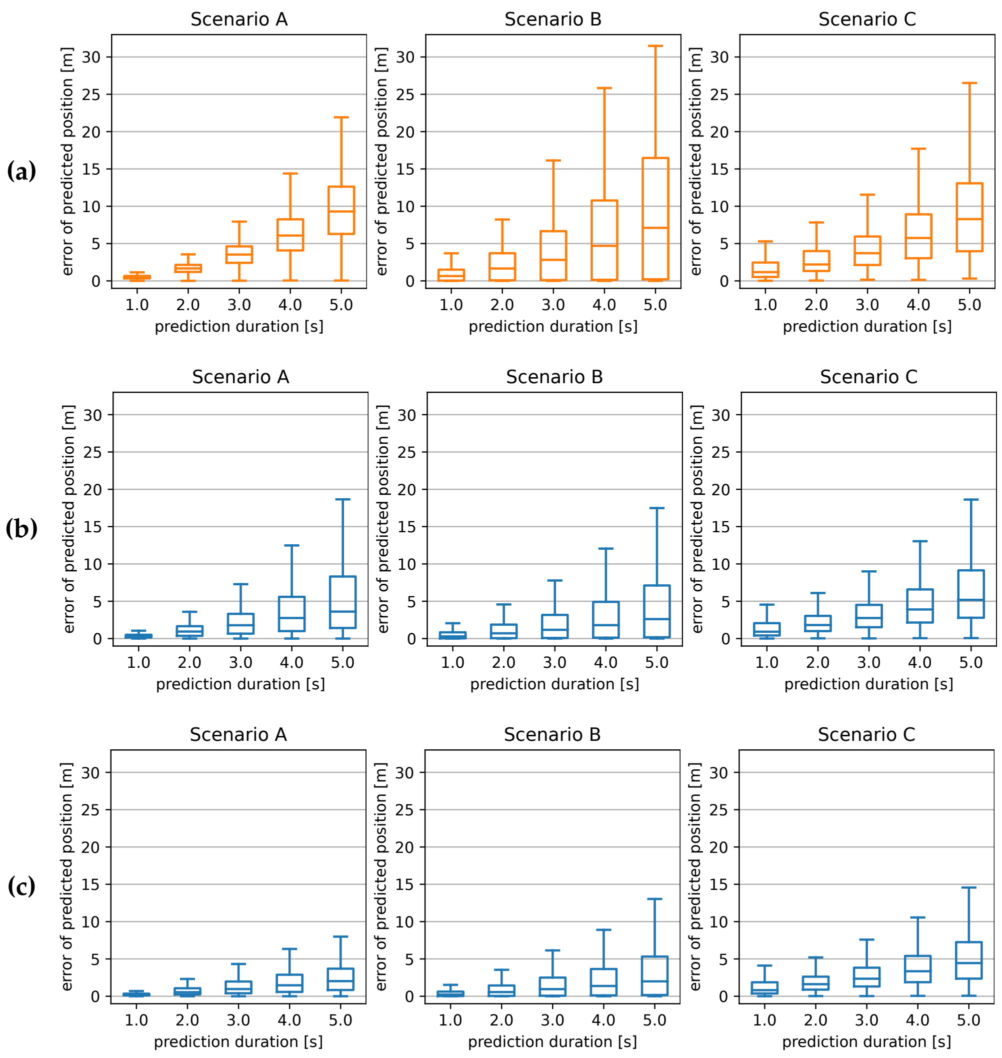


Figure 5.7: Box plots of the error of the predicted object position over prediction duration. The error is the euclidean distance from the predicted position to the actual recorded position of the object at the respective point in time. Each column of the plots shows the results for one of the data sets. Row **(a)** presents the error in the prediction for the simple velocity-based approach, **(b)** for the proposed swarm-based approach. In **(c)** the error of the swarm-based approach is evaluated excluding predictions where the algorithm has to fall back to the velocity-based prediction due to the lack of vehicles driving ahead.

The plots of the resulting errors are broken down according to the data sets. It can be seen that the results are pretty similar for all three scenarios, despite the different environments. Furthermore, the evaluation shows a significant improvement in the accuracy of the proposed swarm-based

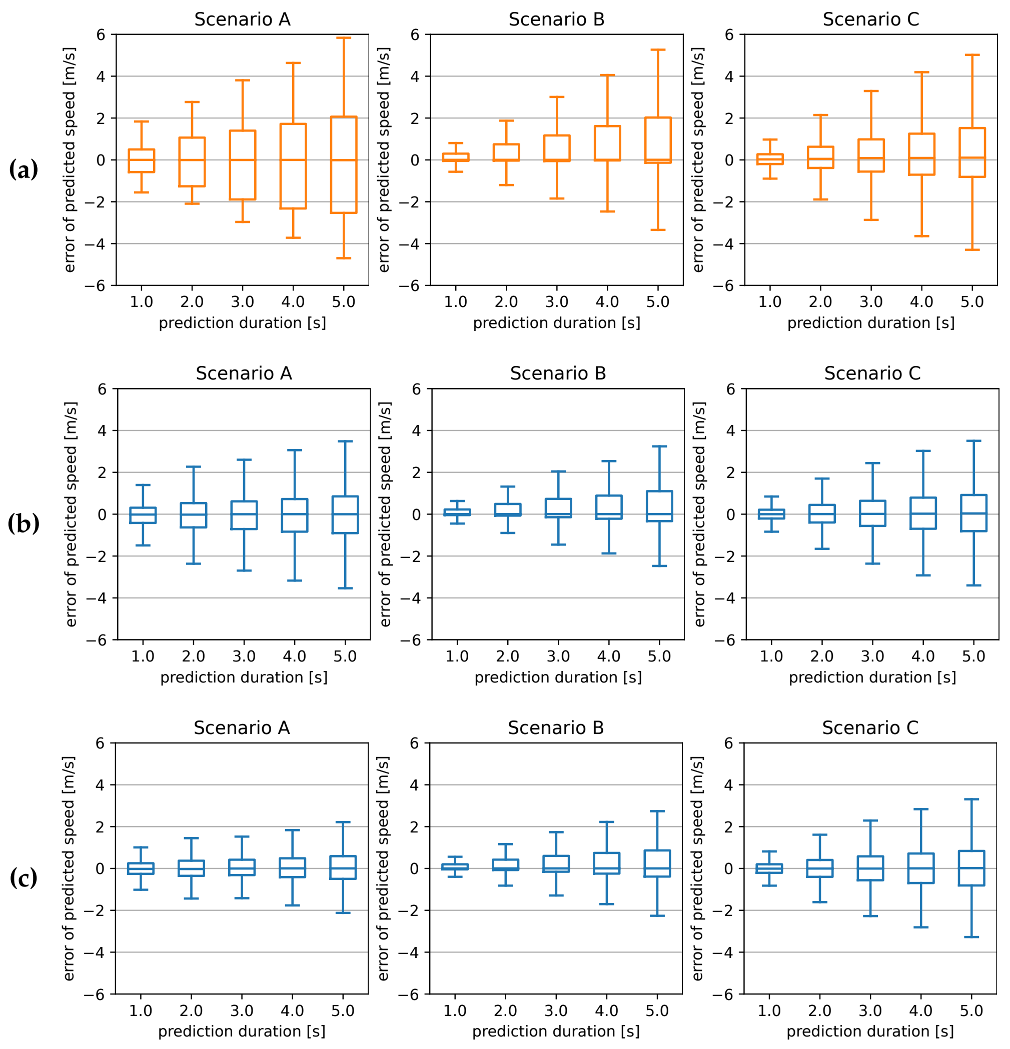


Figure 5.8: Box plots of the error of the predicted speed of objects over prediction duration. The error is the difference of the predicted speed to the actual recorded speed of the object at the respective point in time. Each column of the plots shows the results for one of the data sets. Row **(a)** presents the error in the prediction for the simple velocity-based approach, **(b)** for the proposed swarm-based approach. In **(c)** the error of the swarm-based approach is evaluated excluding predictions where the algorithm has to fall back to the velocity-based prediction due to the lack of vehicles driving ahead.

approach for the position and speed. Over all data sets, the predicted position error median decreases from 0.83 m to 0.21 m with a prediction duration of 1 s, and from 8.13 m to 3.32 m after 5 s. The maximum of the measured error is reduced by $\sim 1/3$ on average for prediction of the position as well as the speed. This improvement is even more highlighted when omitting the objects, for which the algorithm has to fall back to the velocity-based prediction due to the lack of vehicles driving ahead.

At which rate the swarm-based prediction is possible in the different data sets correlates directly with the average number of observed vehicles when taking into account how often oncoming traffic is observed at all. As the algorithm divides the traffic into two clusters (oncoming and same-directed), in most cases, two vehicles cannot be predicted using the other's trajectories: the vehicle driving in the respective front position. Furthermore, in some cases, there are two leading vehicles in different lanes in the same direction. Thus, in data set A, the rate of objects where the swarm-based prediction was possible is 78.52 % (with ten vehicles observed at all times). In data set B, the rate is at 69.41 % with six vehicles observed on average. The rate being above the ratio of 4/6 is because not at all times oncoming traffic was observed. This is also the case for data set C, where oncoming traffic is only observed in 15.83 %. The respective rate is 76.23 % at five vehicles observed on average.

It can be concluded that the proposed swarm-based approach substantially improves the performance of the prediction. The results on the data sets recorded in real traffic are quite comparable to the results in a pure simulation environment. As stated above, this is an essential factor for the performance of the STEBLE trajectory planning, as it heavily relies on predicted trajectories of other vehicles.

5.3 Evaluation of STEBLE Trajectory Planning

In this section, the major contribution of this work, the STEBLE trajectory planning, is evaluated in detail. In Section 5.3.1 the general parameters used throughout the experiments, as well as thresholds and weights of the objective functions, are discussed. A statistical analysis of the convergence of the objective function and the influence of the proposed initialization algorithm is given in Section 5.3.2. Section 5.3.3 discusses the general validity of the generated trajectories, based on statistics on the performance in the three evaluation data sets. Furthermore, exemplary trajectories are analyzed in detail for chosen scenarios. The accuracy of following other vehicles' paths is evaluated in Section 5.3.4. Finally, in Section 5.3.5 the trajectories generated by STEBLE trajectory planning are compared to those of a human driver.

5.3.1 Parameters, Weights and Thresholds

The performance of the STEBLE approach depends on several parameters. Some of those parameters' values can be derived from external requirements; some depend on the user's preferences. Also, many of the parameters - especially the weights and thresholds of the objective functions - are interdependent with the configuration of multiple other parameters. The remainder of this section discusses the parameter values used throughout the experiments presented in this chapter. Section 5.3.1 covers the parameters which affect the general properties of the approach and the whole of the trajectory. In Section 5.3.1 the weights and specific thresholds for each objective function are described.

General Parameters

In this subsection, parameters are discussed which affect the whole of the trajectory planning algorithm. Most of those parameters are directly related to the run time of the implementation. The most defining parameter in this regard is the frequency of replanning the trajectory, i.e., how often is the whole sequence from initialization to optimization executed. The pose of the ego vehicle and the observed environment are usually constantly changing, so we want to replan as often as possible in theory. On the other hand, the sensors perceiving the environment of the autonomous vehicle have a fixed frequency; any updates with the same sensory data are essentially useless. Moreover, the replanning frequency limits the overall runtime of the trajectory planning, as the processing of one trajectory should be completed before the next planning iteration starts to guarantee the timely availability of a valid trajectory. Taking all this into account, the replanning frequency for the experiments throughout this thesis was set to $f = 10\,\text{Hz}$, which results in a maximum time frame of 100 ms for each planning iteration.

The other global parameters were tuned in accordance with this time frame. In this context, the most critical parameter is the global time interval Δt, which describes the temporal distance between two consecutive poses on the trajectory. The value of Δt is one of two factors determining the number n of poses on the trajectory, one of the significant factors regarding the computational cost. Following the policy to reduce this cost as much as possible without affecting the performance of the algorithm in terms of safety and comfort, the value of Δt was set to 0.2 s in the context of this thesis. The vehicle MadeInGermany used in the experiments is allowed to drive autonomously at velocities up to 100 km/h. The resulting spatial distance of two consecutive poses at this velocity is $\sim$5.5 m, which is small enough to guarantee that no object can slip between two poses at velocities below this threshold (taking into account the minimum obstacle separation and vehicle dimensions).

The other determining factor for the number n of poses on the elastic band is the overall duration ΔT of the whole planned trajectory.

$$n = \frac{\Delta T}{\Delta t}$$

The duration ΔT was set to 5 s for the experiments. This property limits the number of poses to $n = 25$ but still enables some foresight regarding evasive maneuvers. It has to be noted that the actual number of poses on the elastic band (i.e., the duration of the trajectory) can be reduced to a number less than n due to the pruning of invalid states (compare Section 4.5). Furthermore, the (maximum) number n of poses on the elastic band also affects the cost of predicting objects, as it determines the number of each object's predicted poses (compare Chapter 3).

Another major factor influencing the run time of the algorithm is the number of iterations when optimizing the trajectory (compare Section 4.1.3). For the detection and pruning of invalid states (compare Section 4.5), those iterations are partitioned into an inner and outer loop: While the number of iterations of the inner loop represents the actual steps of the graph optimization, the range of the outer loop defines how many of those optimization batches are executed. The number of iterations was set to 10 for the inner loop and 4 for the outer loop, respectively. This results in an overall maximum of 40 iterations per elastic band. As can be seen in the statistical analysis of the objective function in Section 5.3.2, this number of iterations is more than sufficient. In the experiments throughout this thesis, no substantial optimization is performed after much fewer iterations. Nonetheless, there are still marginal improvements in some cases, so the maximum number of optimization iterations was chosen to come close to the limits of the given time frame for each planning iteration (i.e., 100 ms).

The run time is also largely affected by the number of simultaneous planning iterations. A total of three optimization runs are performed with different initial trajectories to increase the chance of finding a globally optimal trajectory (compare Section 4.6). The respective initial trajectories are based on: (1) the heuristically best target vehicle's trajectory, (2) the best target's path, but braking with maximum longitudinal acceleration is assumed, and (3) the heuristically second-best target vehicle's trajectory.

Other general parameters, which can be configured in the presented implementation, concern the vehicle's dimensions. The autonomous vehicle MadeInGermany, which was used throughout the experiments, is represented by a stadium shape anchored at the center of the front axle (compare Figure 4.8c from Section 4.2.4). The respective lengths were set to $a_r = 3.8\,\mathrm{m}$ and $a_f = 1\,\mathrm{m}$, resulting in an overall length of 4.8 m of the straight sides. The radius of the stadium shape was set to $r_{ego} = 1\,\mathrm{m}$. The wheelbase of the vehicle is $d_w = 2.709\,\mathrm{m}$, which is of relevance for the objective functions using the center of rotation, as they need to reproject the poses coordinates from the center of the front axle.

Weights and Thresholds of the Objective Functions

In this section, the weights and thresholds related to the individual components of the objective functions are discussed. It has to be stated that the influence of one distinct objective on the actual trajectory depends not only on its own weight but is also tightly connected to the parameters of the other objective functions. The relative differences in the weight define the priority of the objectives. Nonetheless, the values of the thresholds cannot be ignored in this regard. A significant part of the experiments in [39] focuses on this aspect. In summary, it can be said that it is very difficult to determine optimal weights and thresholds, given the infinitely large space of possible configurations and scenarios in real traffic. Many sets of values work equally well, measured by the subjective impression of safety and comfort of the generated trajectories. Thus, the optimal values also depend on the preferences of the user.

There are some fundamental considerations valid for all configurations of values. Kinematic constraints must have the highest priority; generating trajectories the vehicle cannot follow makes no sense. This high

Obj. Func.	Weight	Threshold	Threshold Description
f_{nhk}	1000000	n/a	
f_{turn}	1000000	5 m	The minimum turning radius of the vehicle
$f_{forward}$	1000000	n/a	
f_{a_cen}	4000	$2\,\mathrm{m/s^2}$	Maximum centripetal acceleration
f_{a_ang}	4000	$0.5\,\mathrm{rad/s^2}$	Maximum angular acceleration
f_{a_lon}	3500	$1\,\mathrm{m/s^2}$	Maximum long. positive acceleration
		$4\,\mathrm{m/s^2}$	Maximum long. negative acceleration
f_{ob}	1000	2 m	Minimum spatial distance to obstacles
		1 s	Dynamic long. safety distance (temporal)
f_{v_max}	500	on init	Maximum velocity on the trajectory (m/s)
f_{path}	400	n/a	
f_{v_opt}	30	on init	Optimal velocity on the trajectory (m/s)
f_{a_cen}	20	$0\,\mathrm{m/s^2}$	Desired centripetal acceleration
f_{a_ang}	20	$0\,\mathrm{rad/s^2}$	Desired angular acceleration
f_{a_lon}	10	$0\,\mathrm{m/s^2}$	Desired long. acceleration (pos. and neg.)

Table 5.1: Overview of weights and thresholds of the individual objective functions f_k. The thresholds related to the maximum and optimal velocity are determined during the initialization of each elastic band (compare Section 4.4). The objective functions related to acceleration are added twice with different weights and thresholds to reflect different limits for safety and comfort, respectively.

priority policy also includes the functions related to maximum acceleration and velocity, although the given thresholds do not represent the vehicle's limits and thus can be slightly violated. The same can be said for avoiding obstacles. Although it is undoubtedly essential to avoid collisions, the respective threshold was chosen to have a large safety margin that can be undershot. Thus, the relative weight for avoiding objects can be much lower than the kinematic constraints. Following the trajectories has a lower priority than the safety-related objectives, although it is an essential factor in the concept of STEBLE. Similarly, objectives representing comfort aspects of the trajectory have much lower weights. In this context, it has to be mentioned that the same objective function regarding the acceleration was used twice with different thresholds and weights to represent safety and comfort restrictions, respectively.

In the experiments presented in this thesis, the thresholds and weights given in table 5.1 were used. After deriving initial values from common sense and previous experiments (compare [39]), they were adjusted based on the preferences of the safety drivers of the autonomous vehicle MadeInGermany in many iterations of frequently occurring scenarios - simulated, live in real traffic, and based on recorded data.

One example of a parameter that is particularly prone to the driver's preference is the maximum longitudinal acceleration. While in the experiments relatively low values of $1\,\mathrm{m/s^2}$ for positive and $4\,\mathrm{m/s^2}$ for negative acceleration were used, most human drivers accelerate more aggressively. The lower values were chosen to give the safety drivers more time to intervene in case of misbehavior of the autonomous vehicle. The same reasoning applies to another parameter: the relatively large minimum spacial distance of 2 m to other objects. It also has to be noted that the chosen longitudinal safety distance of 1 s (defined in the time domain to take into account the respective velocity) is only slightly above the minimum of 0.9 s required by German law. According to [52], 41 % of the drivers are on average falling below the legal distance when following other vehicles on country roads or motorways. In an urban environment, the human drivers tend to leave much less space; the chosen safety distance of 1 s was often perceived as extraordinarily large during the experiments.

Another result from the tuning of the parameters under the preferences of the test drivers is the relatively higher weights of centripetal and angular acceleration in relation to the weight of the longitudinal acceleration. This order of the weights implies that braking is generally preferred over swerving for obstacle avoidance; braking maneuvers are generally easier to handle for the driver in most scenarios. The thresholds for the desired acceleration were set to zero for all kinds of acceleration, and the associated functions have much lower weights than their safety-related counterparts.

A particular case regarding the thresholds are the velocity-related objective functions. Their values are determined during the initialization of each elastic band (compare Section 4.4). The maximum velocity is based on the observations of other vehicles at the respective point in time. The value for the optimal velocity is used to control the follow distance to the chosen target vehicle and thus also depends on the current observations.

Hard Threshold	Description
4 m	The minimum turning radius of the vehicle
$4\,\mathrm{m/s^2}$	Maximum centripetal acceleration
$1\,\mathrm{rad/s^2}$	Maximum angular acceleration
$4\,\mathrm{m/s^2}$	Maximum long. positive acceleration
$8\,\mathrm{m/s^2}$	Maximum long. negative acceleration
0.5 m	Minimum spatial distance to obstacles
27.7 m/s	Maximum velocity on the trajectory

Table 5.2: Hard constraints for checking the validity of the optimized trajectory. The given threshold represents the actual limit of the vehicle or acceptable dynamics for the passengers. If one of the thresholds is violated for any pose (or pair/triple of poses, respectively), the trajectory is declared invalid.

As explained in detail in Section 4.5, the trajectories are validated and potentially pruned during the optimization process. In this validation process, the minimum distance to objects, the curvature, the velocity, as well as the longitudinal, angular, and centripetal acceleration at each pose (or pair/triple of poses, respectively) are compared to thresholds representing hard constraints, which can be found in Table 5.2.

5.3.2 Statistical Analysis of the Objective Function

The critical component of the optimization step of the STEBLE trajectory planning is the weighted multi-objective function $f(B)$. It calculates the weighted sum of several objective functions $f_k(B)$ with corresponding weights γ_k representing constraints and objectives for the trajectory (compare Section 4.2). It is hard to visualize the objective function $f(B)$ itself due to the high dimensionality. Instead, in this section, statistics on the resulting values and the values of the individual objective functions $f_k(B)$ are presented. The basis for this evaluation are ~ 30000 planning iterations sampled from all three data sets (compare Section 5.1). For each sample, the value of $f(B)$ as well as each $f_k(B)$ is stored for each iteration of the optimization process. As described in Section 5.3.1, the number of iterations is set to 10 for the inner loop and 4 for the outer loop, resulting in a total number of 40 iterations of the optimization step. Trajectories pruned due to the detection of invalid poses are omitted from this data set, i.e., all trajectories have a total length of 5 s, corresponding to precisely 25 poses at a temporal distance of 0.2 s.

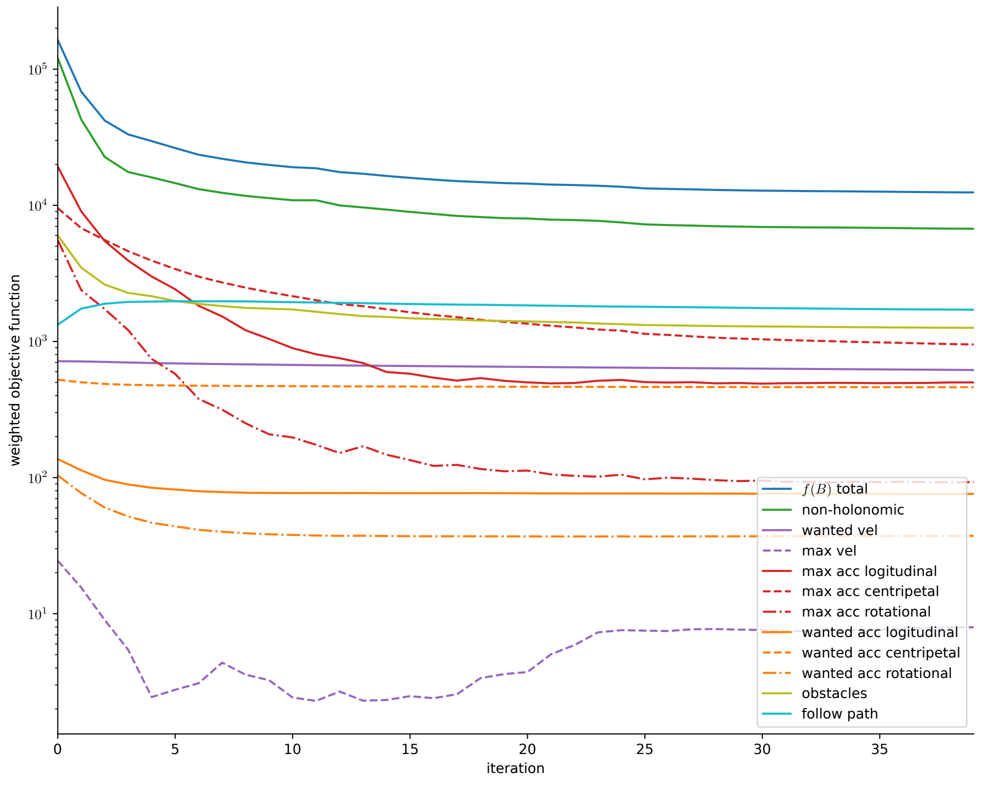


Figure 5.9: The mean value of the weighted multi-objective function $f(B)$ and individual weighted objective functions $\gamma_k f_k(B)$ over iterations of the optimization process. A base-10 logarithmic scale is used for the Y-axis of the graph for better presentability. Objective functions for forward driving and minimal turn radius were omitted, as they had zero values for all iterations for the evaluated data sets.

Figure 5.9 shows the mean value of the weighted multi-objective function $f(B)$ and individual weighted objective functions $\gamma_k f_k(B)$ over iterations of the optimization process. First of all, it can be seen that $f(B)$ converges to a local minimum. Furthermore, the contribution of the individual objective functions mostly correlates with the value of respective weights. Notable exceptions are the functions for following the path of other vehicles, forward driving, and the minimum turn radius. The relatively high impact of the path following function is due to it being the only objective bending the trajectory into curves, which often contradicts multiple other objectives (mostly related to acceleration). Consequently, the value of f_{path} is the only component of $f(B)$, which is increased on average during optimization. The same reasoning applies to the objective function for avoiding obstacles, although to a lesser extent. Regarding the (non-existent) penalty for not driving forward, it has to be noted that the ego vehicle is already in a forward motion for almost all of the taken samples, which adds the acceleration cost of braking down to zero speed to a potential maneuver of reversing the driving direction. The speed of the ego vehicle being distinctly above zero also explains the missing violations of the minimum turn radius, as those can only occur at very low velocities without drastically increasing the centripetal acceleration.

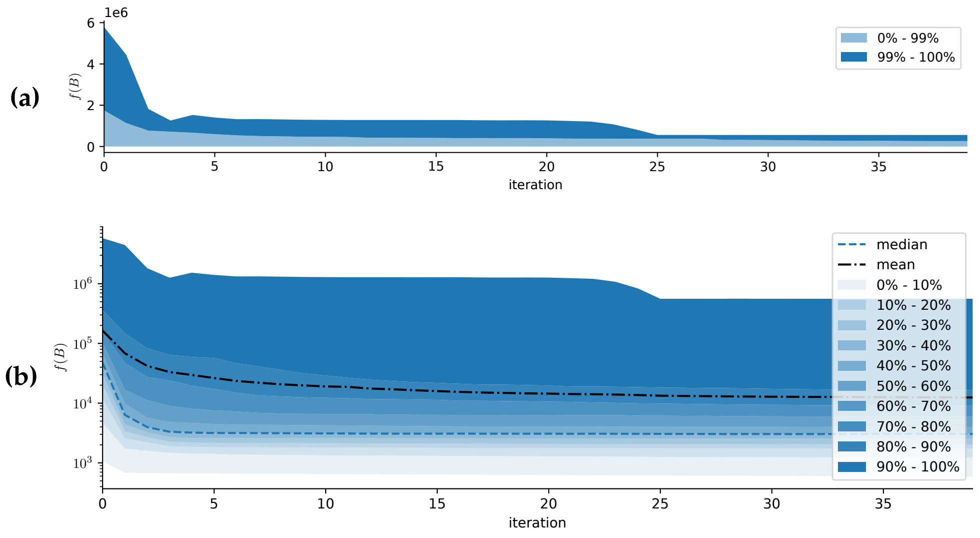


Figure 5.10: Percentile representations of the value of the weighted multi-objective function $f(B)$ over iterations of the optimization process. **(a)** The distribution of samples with values larger and lower then 99 % of all samples. **(b)** The the membership in the corresponding centiles, visualized using a base-10 logarithmic scale for the Y axis.

Another important observation is that the major part of the convergence is performed in the first steps of the optimization process. Figure 5.10 uses a percentile representation of the total value. The fast convergence in the first iterations can be observed throughout all centiles. In 80 % of the samples, the minimum is reached after 10 iterations, in 50 % the value of $f(B)$ decreases only marginally after 3 iterations.

One of the main factors regarding the number of optimization steps necessary is the initialization of the elastic band. In the following, two different initialization algorithms for STEBLE are compared: (1) the complex initialization proposed in Section 4.4 (CSTT, using cubic splines to transition smoothly to the target trajectory) and (2) a naive implementation using configurations sampled on a straight line to the selected target vehicle, with distances between samples calculated from linear interpolation between the ego and the target vehicles' velocities (in the following called STRAIGHT). For the data collection, two trajectory planning modules with the respective versions of the initialization were run simultaneously so that the samples were based on the same observations.

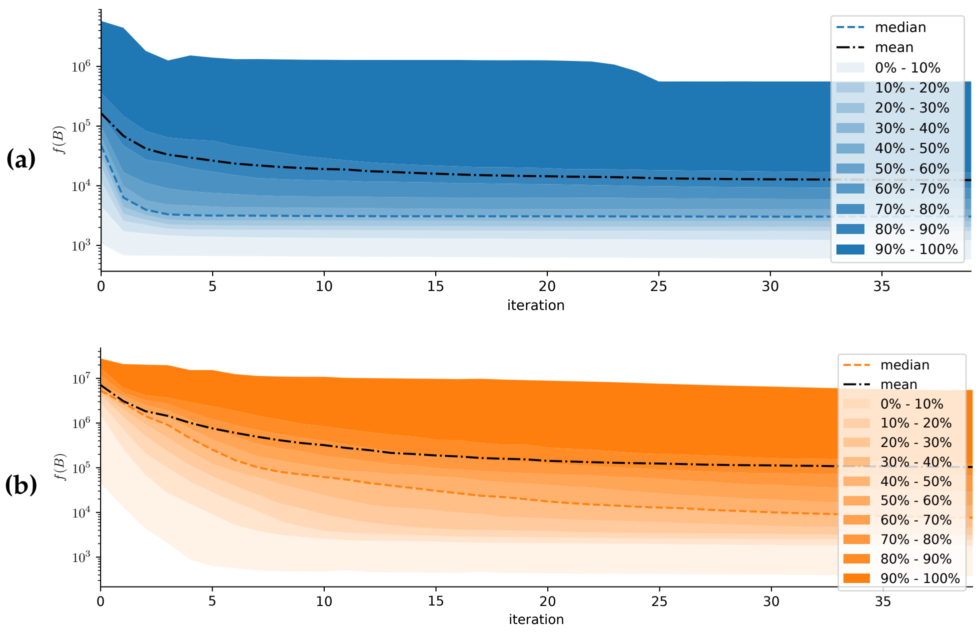


Figure 5.11: Percentile representations of the value of the weighted multi-objective function $f(B)$ over iterations of the optimization process, visualized using a base-10 logarithmic scale for the Y axis. **(a)** The proposed initialization algorithm (CSTT). **(b)** The naive initialization sampled on a straight line to the target (STRAIGHT).

Figure 5.11 compares the percentile representations of the value of $f(B)$ using CSTT and STRAIGHT initialization. It can be seen that the mean value of the proposed approach is lower by a magnitude greater than 50 before any optimization is applied. Even after 40 iterations, it is still lower by a magnitude of $\sim$10. It can be seen that the introduction of a more sophisticated initialization achieves a considerable performance gain, i.e., it reduces the number of necessary optimization iterations significantly. In the naive case, $f(B)$ is still decreasing in the mean even after 40 iterations. With the proposed initialization, there are only marginal changes after 20 iterations. When taking into account the samples with lower initial values, this trend is even more noticeable. Furthermore, the 10 % samples with the highest values are responsible for a substantial amount of the mean value. In the case of CSTT, the maximum initial value of $f(B)$ is decreased by a factor of more than 10 when taking only into account the 90 % lowest samples. In comparison, the maximum value is only reduced by a factor of ∼3.1 when using STRAIGHT.

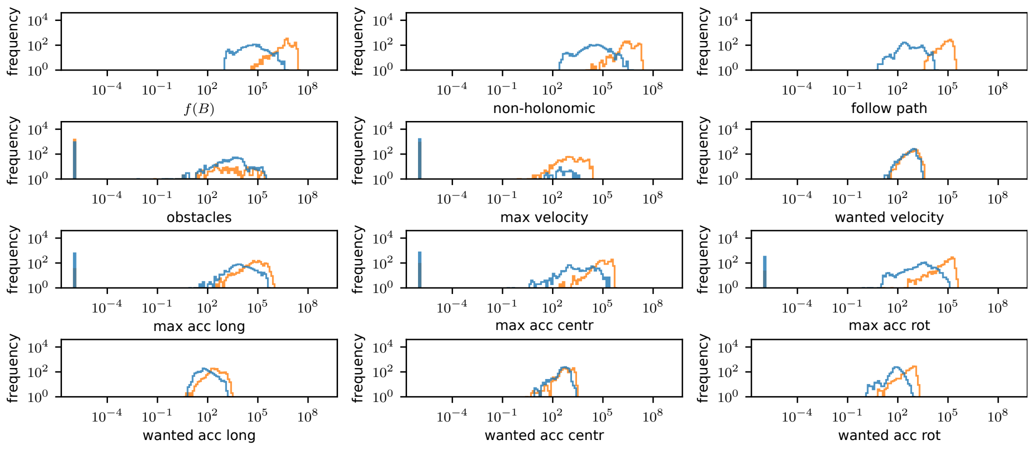


Figure 5.12: Distribution of the value of the weighted multi-objective function $f(B)$ and all relevant individual weighted objective functions $\gamma_k f_k(B)$ before optimization, using CSTT (blue) and STRAIGHT (orange). A base-10 logarithmic scale is used for the bin size of the histograms as well as both axes.

Figure 5.12 show the respective histograms for the individual objective functions before optimization for CSTT and STRAIGHT. CSTT's initial values are lower or equal for all objectives, except for obstacle avoidance. This is due to CSTT following the path of vehicles, where usually more vehicles can be found. However, it cannot be concluded that CSTT tends to find better local optima. A naive approach may converge to a similar (or even better) solution after more iterations. Nonetheless, the proposed initialization algorithm CSTT, explained in detail in Section 4.4, is clearly superior to the naive approach regarding the rate of convergence. Consequently, the number of necessary iterations is reduced significantly and, thus, the computational cost of the optimization.

5.3.3 General Validity of the Generated Trajectories

In this section, the general validity of the trajectories generated by STEBLE trajectory planning is evaluated. Since the STEBLE planning is very dynamic and depends much on the observation of objects, it is impossible to show the behavior of the trajectory planning in every potential scenario. Instead, in this thesis, the general validity of STEBLE is evaluated in two different ways: First, statistics on the performance in the three data sets described in Section 5.1 are presented. Second, the validity of trajectories is analyzed in detail for chosen exemplary scenarios.

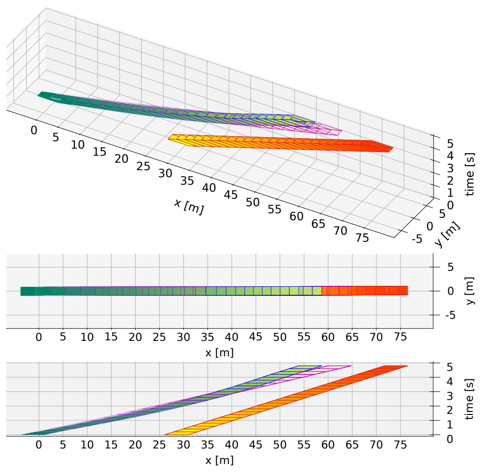
Another indication for the general validity of the proposed approach can be found in [39]: an early version of STEBLE (with fixed time intervals and a flexible goal position) was compared to the original TEB approach (with flexible time intervals and a fixed goal position), based on a simulated merge into traffic maneuver. While both algorithms generated smooth trajectories, the results with STEBLE trajectory planning were much safer concerning the distance to obstacles. This can primarily be attributed to the flexible goal not forcing the trajectory towards a specific location and abandoning the objective function that penalizes a longer total duration of the trajectory.

In the following, the performance of STEBLE is evaluated for all three data sets described in Section 5.1: (A) a small round course with heavily varying curvature, (B) an urban roundabout with several lanes, and (C) a highway environment. The three data sets provide different challenges for trajectory planning. Data set A has many curves with high and frequently changing curvature. Furthermore, the simulated vehicles' random lane changes and planning parameters generate a wide variety of different situations. The traffic environment of data set B also has many vehicles changing lanes and a very high curvature in relation to the speed of the other vehicles. Furthermore, there are sudden changes in the traffic flow due to traffic lights and vehicles turning in different directions in the roundabout. In contrast, the highway environment of data set C, with no traffic lights and no abrupt changes in the curvature, suits the STEBLE trajectory planning. On the other hand, the heavy traffic in the scenario still provides a major challenge, with many vehicles cutting in at significantly varying speeds.

STEBLE generates valid trajectories (as defined in Section 4.5) in all three different environments if a target to follow is available. Overall, at least one target is available in 92.9 % of the planning iterations. When breaking this number down to the individual data sets, data sets A and C always have at least one target available. In data set B, STEBLE can find a vehicle to follow in 80.1%. Although this value for data set B appears to be relatively low at first glance, at a closer look, the absence of valid targets occurs mainly when the ego vehicle is doing a U-turn to re-enter the roundabout (i.e., an unusual maneuver) or when standing in the front row at a traffic light. Also, in the vast majority of cases, more than one target is available in all data sets, which is vital for planning multiple alternative trajectories simultaneously (compare Section 4.6). For the two data sets B and C, based on recorded traffic data, the average of available targets per planning iteration is 2.3 and 3.1, respectively.

All trajectories generated by STEBLE are checked for validity, as described in Section 4.5. There are thresholds for the minimum distance to objects, the curvature, the velocity as well as the longitudinal, angular, and centripetal acceleration, which can be found in Table 5.2 in Section 5.3.1. Trajectories violating these thresholds are pruned at the respective poses. How often this is necessary varies for the data sets. While pruning does not occur at all in the highway scenario (data set C), in the tight round course populated with simulated vehicles for data set A 0.7 % of the generated trajectories have less than the maximum 5 s length. For the roundabout data set B 1.8 % of the generated trajectories are pruned. It has to be noted that the statistics on pruned trajectories include the simultaneously planned alternative trajectories, as described in Section 4.6. The density and distribution of objects can explain the differences in the pruning rate. Although the average distance to objects in the highway scenario C (10.14 m) is not much higher than in the roundabout of B (9.29 m), the vehicles have significantly more lateral distance due to the broader lanes in the former case. Also, in data set B, the distance to objects is biased by the unusual turn maneuvers (generating unusually large distances) and stopping at traffic lights (with low distances to surrounding vehicles). When taking into account only the transits of the roundabout without stopping, the average distance to objects is 6.13 m. In data set A, which consists of purely simulated traffic, the average distance to objects is significantly higher at 14.19 m. This is primarily due to the simulated vehicles using the default FUB_ROSCAR trajectory planning, which keeps much more safety distance than human drivers. The higher pruning rate of scenario A (compared to the highway scenario) can be explained by the lower lane width (i.e., lower lateral distance) and the simulated vehicles' randomly induced lane-change maneuvers. In regard of the overall minimum distance to objects, the values for all data sets are quite similar, with 1.35 m (A), 1.04 m (B) and 1.20 m (C).

While the validity of the trajectories can be checked against hard criteria defined by the physical limits, the quality of the generated trajectories is bound to be subjective since different drivers (and passengers) have different preferences for optimal velocity and tolerated accelerations. Regarding the quality, it can be concluded that the properties of the STEBLE trajectories are comparable to those of human drivers, as described in detail in Section 5.3.5.



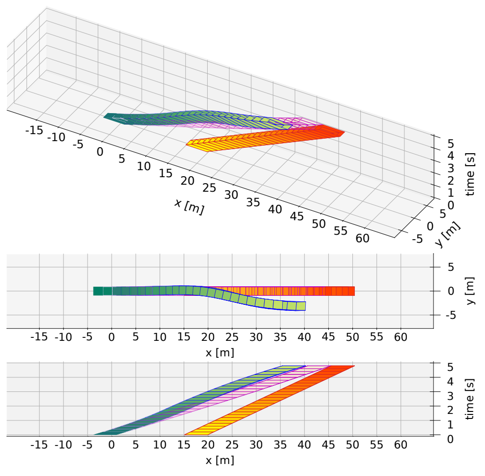


Figure 5.13: Two scenes of the ego vehicle following a vehicle, which performs a braking maneuver. The initial trajectory (pink), the optimized trajectory (green), and the predicted trajectory of the leading vehicle (orange) are shown from different perspectives. **(a)** The leading vehicle is braking with a negative acceleration of $2\,\mathrm{m/s^2}$. The ego vehicle has a higher speed, and the initial trajectory violates the minimal longitudinal distance to the leading vehicle. The optimized trajectory reduces the speed and thus maintains a safe separation while still following the path of the leading vehicle. **(b)** The leading vehicle is performing a very hard braking maneuver with a negative acceleration of $8\,\mathrm{m/s^2}$. The initial trajectory would collide with the leading vehicle. Since the respective objective function limits the maximum negative acceleration to $4\,\mathrm{m/s^2}$, the optimized trajectory cannot maintain a safe separation by braking alone. Instead, a swerve maneuver is planned. The ego vehicle switches to a shifted path parallel to the leading vehicle's.

In the remainder of this section, six exemplary trajectories for three different scenarios are shown in detail. In the first scenario, the leading vehicle is breaking, once with typical negative acceleration and once with very hard negative acceleration (Figure 5.13). The other two scenarios are variations of a merge into traffic maneuver. The merge into traffic maneuver is one of the most challenging scenarios for trajectory planning since it is very dynamic and needs advance planning. In most cases, multiple acceleration and deceleration phases are necessary to align the vehicle to a gap between the other vehicles and match the velocity simultaneously. Figure 5.14 presents two examples of other vehicles performing a merge into traffic, cutting in between the ego vehicle and the target vehicle to follow. In Figure 5.15 the leading vehicle performs a merge into traffic, and the ego vehicle has to do the same to follow.

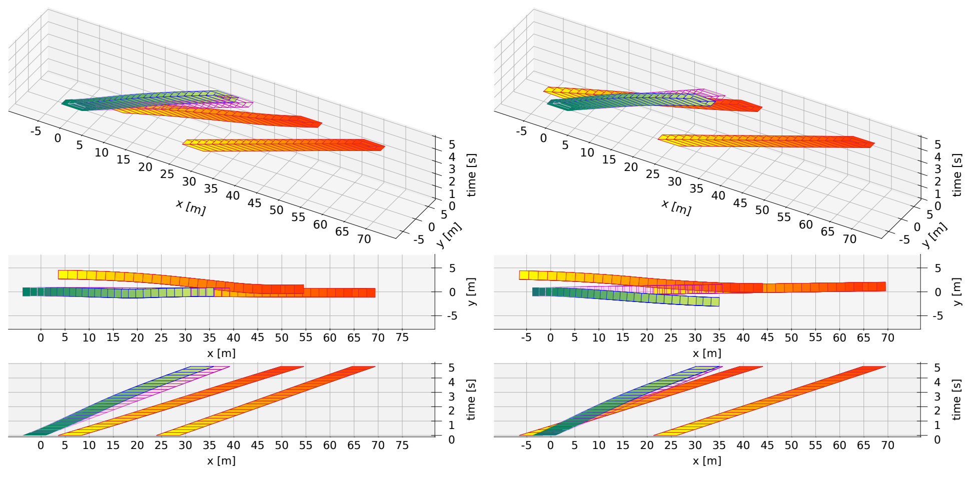
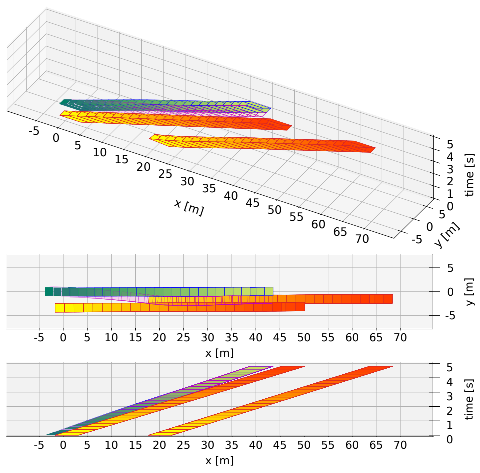
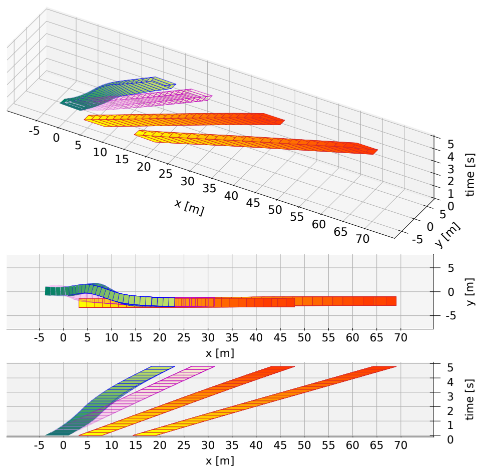


Figure 5.14: Two scenes of another vehicle cutting into the lane between the ego vehicle and a leading vehicle. The initial trajectory (pink), the optimized trajectory (green), and the predicted trajectories of the cutting and leading vehicles (orange) are shown from different perspectives. **(a)** The vehicle cutting into the ego vehicle's lane is positioned approximately 8 m ahead of the ego vehicle. All three vehicles have similar speed. The optimized trajectory reduces the speed and thus maintains a safe separation to both other vehicles, without deviating from the path of the leading vehicle. **(b)** The vehicle cutting into the ego vehicle's lane is positioned approximately 2.5 m behind the ego vehicle. Due to it's slightly higher (predicted) speed, the ego vehicle's initial trajectory collides. The optimized trajectory cannot maintain a safe separation by braking alone, while respecting the limits for longitudinal acceleration. The collision is avoided by shifting the trajectory in parallel to the predicted path of the cutting vehicle.



(a)



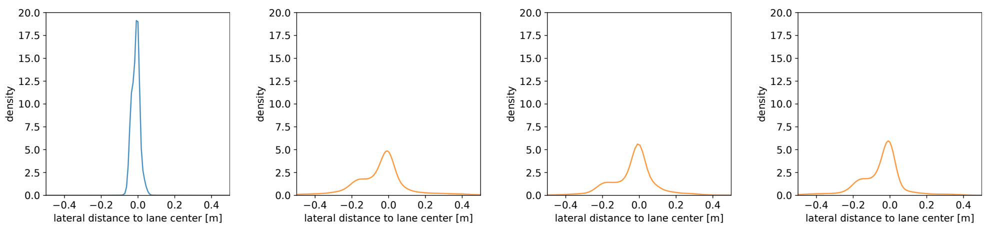
(b)

Figure 5.15: Two scenes of the ego vehicle following a vehicle, which performs a merge into traffic maneuver. The initial trajectory (pink), the optimized trajectory (green), and the predicted trajectories of the other vehicles (orange) are shown from different perspectives. **(a)** The leading vehicle has not yet finished the merge. The ego vehicle's initial trajectory collides with the predicted trajectory of another vehicle, positioned slightly ahead in the lane the leading vehicle merges into. The optimized trajectory avoids the collision by shifting on a parallel path and is falling slowly behind both other vehicles. Scene **(b)** is actually the continuation of scene (a): The leading vehicle has completed the merge maneuver. The ego vehicle has fallen further behind the vehicle blocking the lane. The optimized trajectory reduces the speed and defers the swerve to the other lane and thus maintains a safe separation to both other vehicles during the planned merge into traffic maneuver of the ego vehicle.

In summary, the presented scenes show that the STEBLE approach can not only follow other vehicles at the same speed on the same path (which it does in ∼99.9 % of the evaluated data sets). STEBLE can also deal with more complex scenarios. As can be seen in the figures, it does so by planning trajectories in a more defensive driving style. STEBLE always chooses the slower trajectory due to the lower centripetal force when given the option to either accelerate or decelerate to avoid obstacles. If a target to follow is available, the STEBLE trajectory planning generates safe and driveable trajectories in all evaluated environments. The most important properties of the planned trajectories, i.e., the minimum distance to objects, the curvature, the velocity, as well as the longitudinal, angular, and centripetal acceleration, are guaranteed to be within specified limits. Also, STEBLE trajectory planning can perform complex maneuvers, such as merging into traffic.

5.3.4 Evaluation of the Path Following Accuracy

Another interesting property of trajectories following other vehicles is how close exactly the paths are matching, i.e., what is the lateral offset between the paths. The planned trajectory of the ego vehicle and the paths of vehicles to follow consist of sequences of discrete poses. As they may have different temporal offsets, it is not feasible to directly compare the poses for evaluating the lateral offset. Instead, in Figure 5.16 the lateral distance of the vehicles to the center of the lane (which is represented by a cubic spline) is plotted for data set A.



(a) **(b)** **(c)** **(d)**

Figure 5.16: Kernel density estimate of the lateral distance to the center of the nearest lane for data set A: **(a)** a map-based trajectory planner (following the center of the lane), **(b)** STEBLE trajectory planning, **(c)** STEBLE trajectory planning with increased weight (1000) for path following objective, **(d)** STEBLE trajectory planning without obstacle avoidance.

Figure 5.16a shows the estimated probability density function for the simulated non-ego vehicles of data set A. They are using a map-based trajectory planner, following the center of the lane with slightly varying limits for acceleration and maximum speed (compare Section 5.1.1). Although the poses of the planned trajectory generated by the map-based planner are sampled exactly on the lane center spline, the simulated vehicles slightly deviate from this spline in curves due to the properties of the controller modules. Another source for the simulated vehicles departing from the center of the lane are the randomly enforced lane changes (the simulated vehicles are in a lane change maneuver in $\sim$5 % of the total time). Nonetheless, the non-ego vehicles' lateral distances to the lane center are distributed narrowly around zero.

With STEBLE trajectory planning, i.e., the ego vehicle following the other vehicles without prior knowledge of the road boundaries and center, different characteristics can be observed for left and right lateral shifts, respectively: On the right-hand side (positive lateral distance), the plots are quite similar. This side of the plot represents the outside of the curves. The vehicles are driving on the round course in a counter-clockwise direction; thus, left turns prevail. On the left-hand side, i.e., primarily the inner side of the curves, the deviation from the lane center is significantly larger, although still in a tolerable range.

While Figure 5.16b shows the plot for planning with default parameters (compare Section 5.3.1), Figure 5.16c highlights the influence of the weight of the path following objective. It can be seen that increasing this weight to more than two times the original value (i.e., from 400 to 1000) affects the distribution only slightly. It has to be noted that such a high weight for the path following is not feasible since it interferes with the objective of avoiding objects (which has a weight of 1000 and a similar range of values). Figure 5.16d highlights the influence of the obstacle avoidance objective. In contrast to the map-based planner, STEBLE is affected by vehicles driving in neighboring lanes. When hiding all objects except the target vehicle from the STEBLE planner, the distribution narrows, i.e., the accuracy in following the path of the target vehicle is improved.

It can be concluded that the STEBLE trajectory planner's lateral distance to the center of the lane is acceptable, although it deviates more than the reference map-based planner, particularly on the inside of curves. The variance of the lateral distance measured for data set A (where the position of all objects is known precisely) is much lower than the uncertainty in the measurements of data set B and C. As the error in the measured objects' position would dominate the actual offset between the driven trajectories, those data sets were omitted from the evaluation in this section.

5.3.5 Comparison with Human Drivers' Trajectories

To evaluate the subjective quality of the optimized trajectories, the acceleration and distance to obstacles were compared to the trajectories of human drivers in [40]. Although the results are still meaningful, those experiments were repeated for this work since the STEBLE trajectory planning was refined in several aspects.

The comparison is based on six selected transits of the roundabout from data set B (compare Section 5.1.2 for a detailed description of the data set). At the start of each of the six segments, the position of the simulated ego vehicle (using the STEBLE trajectory planning) is reset to the recorded pose of the ego vehicle (driven by a human driver). During the segments, the pose of the simulated ego vehicle never deviates more than 20 m from the recorded original pose. Thus, there are no jumps in the position for the STEBLE planning, which may have distorted the results. Also, for all segments, there was always at least one valid target to follow available.

Table 5.3 shows the results of measuring the three most meaningful values for judging the quality of a trajectory: the longitudinal acceleration, the centripetal acceleration, and the distance to the closest object. The properties of the recorded vehicle, driven by a human driver, are compared to the properties of a simulated vehicle, using the same recorded observations and the STEBLE trajectory planning. For comparison, the results of the evaluation in [40], using an earlier version of STEBLE, are also presented here. The respective rows are annotated with STEBLE'18.

The central statement of the previous evaluation is still valid (and even more so): The values for the simulated vehicle using STEBLE and the human-driven recorded vehicle are quite comparable. The minimum and maximum values are in a similar range. The difference in the average speed is only 0.7 m/s, the difference in the average longitudinal and centripetal acceleration is also insignificant at $0.2\,\mathrm{m/s^2}$ and $0.1\,\mathrm{m/s^2}$, respectively. Regarding the distance to objects, the STEBLE vehicle never has less separation than the human driver. The very low minimum distances in the second and sixth segments result from the vehicle being surrounded by other vehicles while stopping at the traffic light. Also, the average distance to objects is comparable (STEBLE: 6.13 m, human: 5.16 m), although the human driver tends to maintain more narrow safety margins (especially notable in segment one and four).

segment	cars observed (max/min/avg)	trajectory planning	valid targets (max/min/avg)	target changes	speed [m/s] (max/min/avg)	long acc [m/s^2] (max/avg)	cen acc [m/s^2] (max/avg)	dist to obst. [m] (max/min/avg)
1	18 / 6 / 11.2	STEBLE'18	10 / 1 / 5.5	2	13.9 / 0.0 / 8.8	4.3 / 1.2	2.1 / 0.8	10.4 / 0.2 / 7.0
		STEBLE	10 / 2 / 5.9	1	13.6 / 0.0 / 8.6	3.2 / 1.0	2.2 / 0.7	10.6 / 1.8 / 7.7
		human			10.8 / 0.0 / 7.5	3.2 / 0.7	2.8 / 0.7	11.4 / 1.8 / 4.0
2	10 / 4 / 7.1	STEBLE'18	7 / 1 / 4.1	0	13.5 / 0.0 / 9.0	3.4 / 0.6	2.5 / 1.0	10.5 / 0.6 / 4.8
		STEBLE	7 / 1 / 4.5	0	13.6 / 0.0 / 8.7	3.2 / 0.5	2.5 / 0.9	6.5 / 0.6 / 4.0
		human			14.0 / 0.0 / 8.4	3.0 / 0.6	5.4 / 1.0	5.5 / 0.4 / 3.1
3	9 / 5 / 6.8	STEBLE'18	5 / 2 / 2.4	0	12.5 / 7.5 / 11.3	2.2 / 0.5	1.7 / 0.7	14.9 / 1.7 / 8.1
		STEBLE	5 / 2 / 2.9	0	12.7 / 4.5 / 11.2	2.2 / 0.5	2.0 / 0.8	12.9 / 2.7 / 8.5
		human			13.4 / 4.2 / 10.7	1.9 / 0.5	2.8 / 1.0	17.0 / 3.2 / 9.4
4	14 / 4 / 5.6	STEBLE'18	6 / 1 / 2.7	0	13.8 / 6.9 / 10.9	2.0 / 0.7	1.8 / 0.7	6.8 / 1.1 / 5.1
		STEBLE	6 / 1 / 3.6	0	12.7 / 5.0 / 10.1	2.1 / 0.7	1.7 / 0.7	6.3 / 1.7 / 5.7
		human			13.8 / 4.7 / 10.6	3.6 / 0.8	3.7 / 0.6	7.6 / 0.5 / 3.9
5	18 / 2 / 9.4	STEBLE'18	12 / 1 / 4.4	4	12.3 / 0.0 / 7.9	3.5 / 0.8	2.9 / 0.8	11.5 / 0.9 / 6.5
		STEBLE	12 / 2 / 5.2	1	12.7 / 0.0 / 8.2	2.9 / 0.6	3.0 / 0.9	11.7 / 1.8 / 6.3
		human			12.3 / 0.0 / 8.6	3.9 / 0.7	3.8 / 1.1	11.2 / 1.2 / 5.8
6	15 / 5 / 11.1	STEBLE'18	9 / 1 / 6.1	3	13.5 / 0.0 / 8.7	4.8 / 0.9	1.9 / 0.7	10.1 / 0.1 / 5.3
		STEBLE	9 / 1 / 6.7	1	13.2 / 0.0 / 8.9	3.8 / 0.7	2.0 / 0.7	10.4 / 0.5 / 5.1
		human			11.7 / 0.0 / 7.5	3.8 / 0.9	2.6 / 0.9	8.6 / 0.5 / 4.8

Table 5.3: Comparison STEBLE trajectory planning vs. human driver

Furthermore, when comparing the results of the 2018 evaluation and the performance of the current version of the STEBLE trajectory planning, a significant improvement can be observed. The very low minimum distance to objects of STEBLE'18 in segments one and six is increased with the current version. This is due to two reasons: the introduction of the pruning step (compare Section 4.5), as well as optimizing multiple alternative trajectories simultaneously (compare Section 4.6). While the pruning prevents the elastic band from being stuck in invalid states, the concurrent trajectory planning enables finding potentially better local optima. Accordingly, the maximum longitudinal acceleration in those segments is reduced. Over all segments, the average longitudinal and centripetal acceleration is slightly lower, primarily due to the more accurate object prediction (compare Chapter 3) and improved concept for keeping the longitudinal distance (compare Section 4.2.4). Two other minor enhancements can be attributed to the swarm-based object prediction: First, the number of available valid targets to follow is slightly increased. This is due to vehicles driving in parallel to or slightly ahead of the ego vehicle can now be taken into account more often since their trajectories are predicted along the paths of vehicles driving ahead. Second, due to the more stable prediction, the target vehicle changes significantly less often.

The quality of the generated trajectories is bound to be subjective since different drivers (and passengers) have different preferences for optimal velocity, tolerated accelerations, and safety margins. The evaluation in this section shows that the generated STEBLE trajectories are comparable to those of human drivers, and the respective properties are guaranteed to be within the specified limits. Also, the comparison to the 2018 version of STEBLE trajectory planning reveals significant improvements due to several enhancements of the approach introduced in this work.

Chapter 6

Conclusions

This work presents an algorithm to enable trajectory planning for autonomous vehicles when no map or precise localization is available. The proposed STEBLE approach utilizes the swarm of surrounding vehicles as a source for environment perception and generates a collision-free and locally optimal trajectory within the physical limitations of the vehicle. The well-known concept of elastic bands is adapted to realize this approach, resulting in a comprehensive set of weighted objective functions. This set includes functions that can be adapted to different drivers' preferences and vehicles' dynamic restrictions. Other objective functions handle the avoidance of static and dynamic objects. Furthermore, one more objective function is proposed to realize a unique property of the STEBLE approach: Integrating into the swarm of local drivers by following the paths of other vehicles.

In combination, the presented objective functions enable two defining features of the STEBLE approach: (1) The time interval between discrete states representing the elastic band is constant, and (2) it is not necessary to spatially freeze the goal position of the generated trajectory. The first property stabilizes the trajectories and significantly improves performance during the optimization process due to the lower dimension of the search space. The second property enables complex dynamic maneuvers. Complementing the optimization step at the core of STEBLE, this work presents several more algorithms for selecting a target vehicle to follow, initializing the elastic band with poses transitioning smoothly to the target's trajectory, the validation and pruning of the elastic band, as well as the concurrent planning of alternative trajectories. Additionally, an algorithm for the swarm-based prediction of objects is proposed.

All of the algorithms mentioned above were evaluated in simulation and on recorded data. Also, live field tests in real traffic were conducted. The experimental results show the validity of the proposed approach to generate safe and driveable trajectories. The proposed initialization process is superior to a naive initialization, as significantly fewer iterations of the optimization step are required to achieve a locally optimal solution. The concept of validating and pruning the elastic band as well as the concurrent planning of alternative trajectories mitigate the inherent drawback of the optimization converging into local minima. While the former can guarantee a collision-free trajectory within the maximum bounds for vehicle dynamics, the latter enhances the probability of finding a valid trajectory. While all presented aspects have a significant impact on the trajectory planning performance, one of the most relevant improvements stems from a task not directly related: the swarm-based prediction of objects. The improvement in the accuracy of the prediction greatly enhances the result of the STEBLE trajectory planning, where the predicted trajectories of other vehicles are not only used for obstacle avoidance but also provide information on the road geometry and characteristics. Experiments were conducted in three fundamentally different environments and prove the applicability of the presented approaches. In the highway scenario, at least one valid target vehicle is available at all times. In the urban scenario, this prerequisite is also met in the vast majority of scenes (when excluding the unusual maneuvers to reenter the test area). The proposed swarm-based object prediction can be applied even more often, as the only prerequisite is that other vehicles are driving on the same road (including vehicles following the ego vehicle).

The presented state of STEBLE trajectory planning has potential for further development, and it can be enhanced in many different aspects. Some improvement could be achieved by adding more objective functions. A natural candidate is a function penalizing jerk, i.e., the rate at which the acceleration changes. The implementation of such a function is straightforward; many of the required computations are already performed for the functions related to acceleration. Nonetheless, the calculation of the jerk is still computationally expensive. Other objective functions could incorporate advanced behavior, such as aligning the vehicle in specific ways to other vehicles to increase the field of view or have more options for maneuvers. Also, in many European countries, it is obligatory to drive on the right-hand side of the road if possible and overtaking is only allowed on the left-hand side. Those rules could be integrated into existing or additional objectives.

Aside from the objective functions, more elaborated strategies for selecting the target vehicle to follow can be developed. On the one hand, this could include information on the accuracy of the information on the potential target. Given the significantly better performance, the availability of the swarm-based prediction could be a further criterion for selecting the target. On the other hand, explicit information directly acquired from other traffic participants could be taken into account (e.g., with vehicle-to-vehicle communication). A different approach for selecting the best target to follow could include a hierarchical solution for the optimization step. Rough trajectories (with a very low number of optimization steps and fewer poses) could be planned for all potential targets concurrently. Based on the evaluation of those rough trajectories, a subset of candidates can then be selected for further optimization with an increased level of detail. Such an approach can improve the selection of targets and reduce the overall computational complexity. The algorithms presented in this work provide all the necessary tools for dynamic parameterization and evaluation.

It can be concluded that the main objective of this thesis was achieved: efficient trajectory planning without relying on a map. The experimental results show that the approach of following other vehicles using elastic bands is valid to compensate (at least temporarily) the advantages of map-based planning. The presented STEBLE trajectory planning adapts the concept of flocking and trail pheromones to autonomous cars and expands the capabilities of the individual by using the group of other drivers as additional sensors, implementing a kind of swarm behavior for autonomous cars.

Appendix A: Implementation of Distance Calculations

The function $\text{dist}(\mathbf{l}_p, \mathbf{l}_q)$ calculates the minimal distance between two two-dimensional line segments $\mathbf{l}_p$ and $\mathbf{l}_q$. The implementation is a specialization of Lumelsky's algorithm [50] and performs ~74 % faster in the presented experiments then a naive implementation.

```
function dist($\mathbf{l}_p, \mathbf{l}_q$)
    $\mathbf{p_a} \leftarrow$ first end point of $\mathbf{l}_p$
    $\mathbf{p_b} \leftarrow$ second end point of $\mathbf{l}_p$
    $\mathbf{p_c} \leftarrow$ first end point of $\mathbf{l}_q$
    $\mathbf{p_d} \leftarrow$ second end point of $\mathbf{l}_q$
    $d_1 \leftarrow |\mathbf{p}_b - \mathbf{p}_a|^2$
    $d_2 \leftarrow |\mathbf{p}_d - \mathbf{p}_c|^2$
    $r \leftarrow (\mathbf{p}_b - \mathbf{p}_a) \cdot (\mathbf{p}_d - \mathbf{p}_c)$  // dot product
    $d \leftarrow d_1 d_2 - r^2$
    $s_1 \leftarrow (\mathbf{p}_b - \mathbf{p}_a) \cdot (\mathbf{p}_c - \mathbf{p}_a)$
    $s_2 \leftarrow (\mathbf{p}_d - \mathbf{p}_c) \cdot (\mathbf{p}_c - \mathbf{p}_a)$
    if $d \neq 0$ then
        $t \leftarrow (s_1 d_2 - s_2 r)/d$
        clamp($t$)
    else    // segments are parallel
        $t \leftarrow 0$
    $u \leftarrow (tr - s_2)/d_2$
    if clamp($u$) then
        $t \leftarrow ur + s_1/d_1$    // recalculate t
        clamp($t$)
    return $|t(\mathbf{p}_b - \mathbf{p}_a) - u(\mathbf{p}_d - \mathbf{p}_c) - (\mathbf{p}_c - \mathbf{p}_a)|$
```

```
function clamp($x$)
    if $x < 0$ then
        $x \leftarrow 0$
        return True
    if $x > 1$ then
        $x \leftarrow 1$
        return True
    return False
```

The function $\text{dist}(\mathbf{p}, \mathbf{l})$ calculates the minimal distance between two-dimensional point $\mathbf{p}$ and line segment $\mathbf{l}$.

```
function dist($\mathbf{p}, \mathbf{l}$)
    $\mathbf{p}_a \leftarrow$ first end point of $\mathbf{l}$
    $\mathbf{p}_b \leftarrow$ second end point of $\mathbf{l}$
    if $\mathbf{p}_a = \mathbf{p}_b$ then
        return $|\mathbf{p} - \mathbf{p}_a|$
    $t \leftarrow ((\mathbf{p} - \mathbf{p}_a) \cdot (\mathbf{p}_b - \mathbf{p}_a))/|\mathbf{p}_b - \mathbf{p}_a|^2$    // $\cdot$ is the dot product operator
    if $t \leq 0$ then
        return $|\mathbf{p} - \mathbf{p}_a|$
    if $t \geq 1$ then
        return $|\mathbf{p} - \mathbf{p}_b|$
    return $|\mathbf{p} - \mathbf{p}_a + td|$
```

Bibliography

[1] Gerardo Beni. “From Swarm Intelligence to Swarm Robotics”. In: *Swarm Robotics*. Springer, 2004, pp. 1–9. DOI: 10.1007/978-3-540-30552-1_1.

[2] Veysel Gazi and Kevin M. Passino. “Stability Analysis of Social Foraging Swarms”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*. Vol. 34. 1. IEEE, 2004, pp. 539–557. DOI: 10.1109/TSMCB.2003.817077.

[3] Craig W. Reynolds. “Flocks, Herds and Schools: A Distributed Behavioral Model”. In: *Proceedings of SIGGRAPH ’87 - 14th Annual Conference on Computer Graphics and Interactive Techniques*. Association for Computing Machinery, 1987, pp. 25–34. DOI: 10.1145/37401.37406.

[4] Craig W. Reynolds. “Steering Behaviors For Autonomous Characters”. In: *Proceedings of Game Developers Conference ’99*. Miller Freeman Game Group, 1999, pp. 763–782. URL: http://red3d.com/cwr/papers/1999/gdc99steer.html (visited on 07/01/2020).

[5] Richard Vaughan et al. “Experiments in Automatic Flock Control”. In: *Robotics and autonomous systems*. Vol. 31. 1. Elsevier, 2000, pp. 109–117. DOI: 10.1016/S0921-8890(99)00084-6.

[6] Reza Olfati-Saber. “Flocking for Multi-Agent Dynamic Systems: Algorithms and Theory”. In: *IEEE Transactions on Automatic Control*. Vol. 51. 3. IEEE, 2006, pp. 401–420. DOI: 10.1109/TAC.2005.864190.

[7] S. Hosseini Semnani, A. H. J. de Ruiter, and H. H. T. Liu. “Force-Based Algorithm for Motion Planning of Large Agent”. In: *IEEE Transactions on Cybernetics*. IEEE, 2020, pp. 1–12. DOI: 10.1109/TCYB.2020.2994122.

[8] James Kennedy and Russel Eberhart. “Particle Swarm Optimization”. In: *Proceedings of ICNN ’95 - International Conference on Neural Networks*. Vol. 4. IEEE, 1995, pp. 1942–1948. DOI: 10.1109/ICNN.1995.488968.

[9] Riccardo Poli. “Analysis of the Publications on the Applications of Particle Swarm Optimisation”. In: *Journal of Artificial Evolution and Applications*. Vol. 2008. Hindawi, 2008. DOI: 10.1155/2008/685175.

[10] Anugrah K. Pamosoaji, Mingxu Piao, and Keum-Shik Hong. “PSO-based Minimum-time Motion Planning for Multiple Vehicles Under Acceleration and Velocity Limitations”. In: *International Journal of Control, Automation and Systems*. Vol. 17. 10. Springer, 2019, pp. 2610–2623. DOI: 10.1007/s12555-018-0176-9.

[11] Michalis Mavrovouniotis, Changhe Li, and Shengxiang Yang. “A survey of swarm intelligence for dynamic optimization: Algorithms and applications”. In: *Swarm and Evolutionary Computation*. Vol. 33. Elsevier, 2017, pp. 1–17. DOI: 10.1016/j.swevo.2016.12.005.

[12] Marco Dorigo. “Optimization, Learning and Natural Algorithms”. PhD thesis. Dipartimento di Elettronica, Politecnico di Milano, 1992.

[13] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. “Ant Colony Optimization”. In: *IEEE Computational Intelligence Magazine*. Vol. 1. 4. IEEE, 2006, pp. 28–39. DOI: 10.1109/MCI.2006.329691.

[14] Johann Dréo - Wikimedia Commons. *The ant colony optimization of the travelling salesman problem*. File: Aco_TSP.svg. 2006. URL: http://commons.wikimedia.org/wiki/File:Aco_TSP.svg (visited on 07/01/2020).

[15] Rutger Claes and Tom Holvoet. “Ant Colony Optimization Applied to Route Planning Using Link Travel Time Predictions”. In: *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*. IEEE, 2011, pp. 358–365. DOI: 10.1109/IPDPS.2011.173.

[16] Michael Brand et al. “Ant Colony Optimization Algorithm for Robot Path Planning”. In: *Proceedings of ICCDA 2010 - International Conference On Computer Design and Applications*. Vol. 3. IEEE, 2010, pp. 436–440. DOI: 10.1109/ICCDA.2010.5541300.

[17] Yijing Wang, Xin Lu, and Zhiqiang Zuo. “Autonomous Vehicles Path Planning With Enhanced Ant Colony Optimization”. In: *Proceedings of 2019 Chinese Control Conference (CCC)*. IEEE, 2019, pp. 6633–6638. DOI: 10.23919/ChiCC.2019.8866128.

[18] Fritz Ulbrich et al. “Extracting Path Graphs from Vehicle Trajectories”. In: *Proceedings of IV 2016 - IEEE Intelligent Vehicles Symposium*. IEEE, 2016, pp. 1260–1264. DOI: 10.1109/IVS.2016.7535552.

[19] Simon Rotter. “Swarm Behaviour for Path Planning”. MA thesis. Institut für Informatik, Freie Universität Berlin, 2014. URL: http://www.mi.fu-berlin.de/inf/groups/ag-ki/Theses/Completed-theses/Master_Diploma-theses/2014/Rotter/Master-Rotter.pdf (visited on 07/01/2020).

[20] Fritz Ulbrich et al. “Adapting to the Traffic Swarm: Swarm Behaviour for Autonomous Cars”. In: *Handbook of Research on Design, Control, and Modeling of Swarm Robotics*. Ed. by Ying Tan. IGI Global, 2016, pp. 263–285. DOI: 10.4018/978-1-4666-9572-6.ch010.

[21] Oussama Khatib. “Real-time Obstacle Avoidance for Manipulators and Mobile Robots”. In: *Proceedings of ICRA ’85 - IEEE International Conference on Robotics and Automation*. Vol. 2. 1985, pp. 500–505. DOI: 10.1109/ROBOT.1985.1087247.

[22] John H. Reif and Hongyan Wang. “Social Potential Fields: A Distributed Behavioral Control for Autonomous Robots”. In: *Robotics and Autonomous Systems*. Vol. 27. 3. Elsevier, 1999, pp. 171–194. DOI: 10.1016/S0921-8890(99)00004-4.

[23] J. Christian Gerdes and Eric J. Rossetter. “A Unified Approach to Driver Assistance Systems Based on Artificial Potential Fields”. In: *Journal of Dynamic Systems, Measurement, and Control*. Vol. 123. 3. ASME, 1999, pp. 431–438. DOI: 10.1115/1.1386788.

[24] Eric J. Rossetter, J. P. Switkes, and J. Christian Gerdes. “A Gentle Nudge Towards Safety: Experimental Validation of the Potential Field Driver Assistance System”. In: *Proceedings of the 2003 American Control Conference*. Vol. 5. IEEE, 2003, pp. 3744–3749. DOI: 10.1109/ACC.2003.1240417.

[25] Danilo A. De Lima and Guilherme A. S. Pereira. “Navigation of an Autonomous Car Using Vector Fields and the Dynamic Window Approach”. In: *Journal of Control, Automation and Electrical Systems*. Vol. 24. 1-2. Springer, 2013, pp. 106–116. DOI: 10.1007/s40313-013-0006-5.

[26] Yadollah Rasekhipour et al. “A Potential Field-based Model Predictive Path-planning Controller for Autonomous Road Vehicles”. In: *IEEE Transactions on Intelligent Transportation Systems*. Vol. 18. 5. IEEE, 2016, pp. 1255–1267. DOI: 10.1109/TITS.2016.2604240.

[27] Zahra Boroujeni et al. “Autonomous Car Navigation Using Vector Fields”. In: *Proceedings of IV 2018 - IEEE Intelligent Vehicles Symposium*. IEEE, 2018, pp. 794–799. DOI: 10.1109/IVS.2018.8500446.

[28] Sean Quinlan. “Real-time Modification of Collision-free Paths”. PhD thesis. Computer Science Department, Stanford University, 1994. URL: http://books.google.de/books?id=etggAQAAIAAJ (visited on 07/01/2020).

[29] Sean Quinlan and Oussama Khatib. “Elastic Bands: Connecting Path Planning and Control”. In: *Proceedings of ICRA ’93 - IEEE International Conference on Robotics and Automation*. Vol. 2. IEEE, 1993, pp. 802–807. DOI: 10.1109/ROBOT.1993.291936.

[30] Stefan K. Gehrig and Fridtjof J. Stein. “Collision Avoidance for Vehicle-Following Systems”. In: *IEEE Transactions on Intelligent Transportation Systems*. Vol. 8. 2. IEEE, 2007, pp. 233–244. DOI: 10.1109/TITS.2006.888594.

[31] Jens Hilgert et al. “Emergency Path Planning for Autonomous Vehicles Using Elastic Band Theory”. In: *Proceedings of AIM 2003 - IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. IEEE, 2003, pp. 1390–1395. DOI: 10.1109/AIM.2003.1225546.

[32] Karina Hirsch et al. “Optimization of Emergency Trajectories for Autonomous Vehicles with respect to Linear Vehicle Dynamics”. In: *Proceedings of the 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. IEEE, 2005, pp. 528–533. DOI: 10.1109/AIM.2005.1511036.

[33] Majdoleen Khatib et al. “Dynamic path modification for car-like nonholonomic mobile robots”. In: *Proceedings of International Conference on Robotics and Automation*. Vol. 4. IEEE, 1997, pp. 2920–2925. DOI: 10.1109/ROBOT.1997.606730.

[34] Thomas Sattel and Thorsten Brandt. “Ground vehicle guidance along collision-free trajectories using elastic bands”. In: *Proceedings of the 2005 American Control Conference*. Vol. 7. IEEE, 2005, pp. 4991–4996. DOI: 10.1109/ACC.2005.1470798.

[35] Xiaolin Song, Haotian Cao, and Jiang Huang. “Vehicle path planning in various driving situations based on the elastic band theory for highway collision avoidance”. In: *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*. Vol. 227. 12. SAGE Publications, 2013, pp. 1706–1722. DOI: 10.1177/0954407013481299.

[36] Christoph Rösmann et al. “Trajectory modification considering dynamic constraints of autonomous robots”. In: *Proceedings of ROBOTIK 2012 - 7th German Conference on Robotics*. VDE-Verlag, 2012, pp. 74–79. URL: http://www.vde-verlag.de/proceedings-en/453418014.html (visited on 07/01/2020).

[37] Christoph Rösmann, Frank Hoffmann, and Torsten Bertram. “Planning of Multiple Robot Trajectories in Distinctive Topologies”. In: *Proceedings of ECMR 2015 - European Conference on Mobile Robots*. IEEE, 2015. DOI: 10.1109/ECMR.2015.7324179.

[38] Martin Keller et al. “Planning of Optimal Collision Avoidance Trajectories with Timed Elastic Bands”. In: *Proceedings of 19th IFAC World Congress*. Vol. 47. 3. Elsevier, 2014, pp. 9822–9827. DOI: 10.3182/20140824-6-ZA-1003.01143.

[39] Fritz Ulbrich et al. “Stable Timed Elastic Bands with Loose Ends”. In: *Proceedings of IV 2017 - IEEE Intelligent Vehicles Symposium*. IEEE, 2017, pp. 186–192. DOI: 10.1109/IVS.2017.7995718.

[40] Fritz Ulbrich et al. “Following Cars With Elastic Bands”. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1529–1536. DOI: 10.1109/IVS.2018.8500481.

[41] Freie Universität Berlin. *MadeInGermany – AutoNOMOS Labs*. 2020. URL: http://autonomos.inf.fu-berlin.de/vehicles/made-in-germany (visited on 07/01/2020).

[42] Ibeo Automotive Systems GmbH. *Ibeo LUX - Sensoren - Products - Homepage*. 2020. URL: http://www.ibeo-as.com/en/produkte/sensoren/IbeoLUX (visited on 07/01/2020).

[43] Applanix - A Trimble Company. *Applanix : POSLV*. 2020. URL: http://www.applanix.com/products/poslv.htm (visited on 07/01/2020).

[44] Stanford Artificial Intelligence Laboratory. *Robotic Operating System*. Version ROS Melodic Morenia. May 23, 2018. URL: http://www.ros.org (visited on 07/01/2020).

[45] Daniel Stonier. *ECL Geometry Package for ROS, Spline Tutorial*. 2012. URL: http://wiki.ros.org/ecl_geometry/Tutorials/Splines (visited on 07/01/2020).

[46] William H. Press et al. *Numerical Recipes in C (2nd Ed.): The Art of Scientific Computing*. Cambridge University Press, 1992, p. 115. URL: http://dl.acm.org/doi/book/10.5555/148286 (visited on 07/01/2020).

[47] Christoph Rösmann. *TEB Local Planner Package for ROS*. 2020. URL: http://wiki.ros.org/teb_local_planner (visited on 07/01/2020).

[48] Rainer Kümmerle et al. “g2o: A General Framework for Graph Optimization”. In: *Proceedings of ICRA 2011 - IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3607–3613. DOI: 10.1109/ICRA.2011.5979949.

[49] Antonin Guttman. “R-trees: A Dynamic Index Structure for Spatial Searching”. In: *Proceedings of SIGMOID ’84 - ACM International Conference on Management of Data*. Boston, Massachusetts: ACM, 1984, pp. 47–57. DOI: 10.1145/602259.602266.

[50] Vladimir J. Lumelsky. “On Fast Computation of Distance between Line Segments”. In: *Information Processing Letters*. Vol. 21. 2. Elsevier, 1985, pp. 55–61. DOI: 10.1016/0020-0190(85)90032-8.

[51] Lester E. Dubins. “On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents”. In: *American Journal of mathematics*. Vol. 79. 3. Johns Hopkins University Press, 1957, pp. 497–516. DOI: 10.2307/2372560.

[52] Björn Filzek and Bert Breuer. *Distance behavior on motorways with regard to active safety - A comparison between adaptive-cruise-control (ACC) and driver*. Tech. rep. SAE, 2001. URL: http://www.sae.org/publications/technical-papers/content/2001-06-0066 (visited on 07/01/2020).