Technische Universität München
Fakultät für Informatik

Technical University of Munich

Using Reachability Analysis for Motion Planning of Autonomous Vehicles in Complex Traffic Situations

Stefanie Manzinger

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Die Dissertation wurde am 19.04.2021 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 17.06.2021 angenommen.

Abstract

Despite the recent successes in the development of autonomous vehicles, there are still major challenges regarding the motion planning of autonomous vehicles in complex traffic situations. In complex traffic situations, autonomous vehicles need to select from various maneuver options, such as overtaking another vehicle to the left or right, but not all maneuvers are safe or reach the desired set of goal states. Since the exhaustive exploration and validation of all maneuver options is typically impossible due to hard real-time constraints, the question arises how to quickly identify feasible maneuver candidates. This challenge becomes even more evident when considering the maneuver-based cooperation within a group of communicating vehicles, where the number of joint maneuvers grows exponentially with each participating vehicle. Moreover there are infinitely many traffic situations and their future evolution is unknown, therefore, motion planning methods need to generalize to arbitrary traffic situations so that safety can be ensured at all times.

To address the aforementioned challenges, this thesis proposes a holistic motion planning approach that can be used for individual, cooperative, and safe motion planning in arbitrary, complex traffic situations. By combining set-based reachability analysis and continuous optimization, we demonstrate how to (a) efficiently explore the solution space for trajectory planning of autonomous vehicles and (b) to identify suitable representations of collision avoidance constraints for arbitrary trajectory planners based on nonlinear programs or (successive) convexification. Since the computation times of our approach typically decrease with increasing criticality of the traffic situation, our approach is particularly suited for planning fail-safe trajectories. In this regard, we integrate our approach into a safety layer that guarantees that the autonomous vehicle complies with legal safety, i.e., the autonomous vehicle never causes accidents even though other road users are allowed to perform any legal behavior. The safety layer verifies whether intended trajectories of the autonomous vehicle comply with legal safety and provides fail-safe trajectories that can be executed in critical situations. To enable maneuver-based cooperation, we propose a cooperation mechanism in which communicating vehicles negotiate space-time reservations based on their reachable sets. This cooperation mechanism is suited for arbitrary, mixed traffic situations and facilitates the transition from individual to cooperative driving.

To demonstrate the benefits of our approach, we extensively validated it in simulation and the drivability of the planned trajectories using a BMW 7-series vehicle. Our test cases include real traffic scenarios publicly available in the CommonRoad benchmark suite that we have initiated with the aim to increase the reproducibility of scientific results. The CommonRoad benchmarks for motion planning are uniquely identified by a short ID, making it easy to describe all necessary details of experiments in publications. CommonRoad is open source and can be easily extended by other researchers helping to accelerate motion planning research.

Zusammenfassung

Trotz jüngster Erfolge in der Entwicklung des autonomen Fahrens bestehen weiterhin große Herausforderungen in der Bewegungsplanung autonomer Fahrzeuge in komplexen Verkehrssituationen. In komplexen Verkehrssituationen müssen autonome Fahrzeuge ein Fahrmanöver aus einer Vielzahl an möglichen Manövern auswählen, jedoch sind nicht alle Manöver sicher oder erreichen die gewünschte Menge an Zielzuständen. Da die vollständige Exploration und Validierung aller Fahrmanöver auf Grund von harten Echtzeitbedingungen für gewöhnlich unmöglich ist, stellt sich die Frage wie möglichst schnell geeignete Manöverkandidaten gefunden werden können. Diese Herausforderung wird sehr deutlich bei der Betrachtung manöverbasierter Kooperationen von Gruppen kommunizierender Fahrzeuge, bei der die Anzahl der kombinierten Fahrmanöver mit jedem teilnehmenden Fahrzeug exponentiell anwächst. Zusätzlich gibt es eine unendliche Anzahl von Verkehrssituationen, deren zukünftige Weiterentwicklung unbekannt ist, weshalb Bewegungsplanungsmethoden auf beliebige Verkehrssituationen generalisierbar sein müssen um den sicheren Betrieb autonomer Fahrzeuge zu gewährleisten.

Um diese Herausforderungen zu adressieren, führt die vorliegende Arbeit einen ganzheitlichen Ansatz für die individuelle, kooperative und sichere Bewegungsplanung autonomer Fahrzeuge in beliebigen, komplexen Verkehrssituationen ein. Mittels der Kombination aus mengenbasierter Erreichbarkeitsanalyse und kontinuierlicher Optimierung zeigt diese Arbeit wie (a) der Lösungsraum für die Trajektorienplanung effizient exploriert und (b) geeignete Nebenbedingungen für die Kollisionsvermeidung für beliebige Trajektorienplaner basierend auf nichtlinearen Programmen oder (sukzessiver) Konvexifizierung identifiziert werden können. Da die Rechenzeiten des vorgestellten Ansatzes in kritischen Verkehrssituationen für gewöhnlich abnehmen, eignet sich dieser insbesondere für die Planung ausfallsicherer Trajektorien. Der vorgestellte Ansatz wird deshalb in eine Sicherheitsebene für Trajektorienplaner integriert, die gewährleistet, dass autonome Fahrzeuge keine Unfälle verursachen, obwohl andere Verkehrsteilnehmer jede mögliche legale Bewegung ausführen dürfen. Die Sicherheitsebene verifiziert dabei die geplanten Trajektorien des autonomen Fahrzeuges und berechnet ausfallsichere Trajektorien, die in sicherheitskritischen Situationen ausgeführt werden können. Für die manöverbasierte Kooperation wird ein Mechanismus vorgeschlagen, bei dem kommunizierende Fahrzeuge Raum-Zeit-Reservierungen basierend auf erreichbare Mengen aushandeln. Der Kooperationsmechanismus ist für die Anwendung in Verkehrssituationen mit gemischtem Verkehr aus menschlichen Fahrern und autonomen Fahrzeugen geeignet und erleichtert den Übergang von individuellem zu kooperativem Fahren.

Die Vorteile des vorgestellten Verfahrens werden ausgiebig in Simulation und die Fahrbarkeit von geplanten Trajektorien mit Hilfe eines BMW 7er Versuchsfahrzeuges demonstriert. Die verwendeten Testfälle beinhalten reale Verkehrsdaten, die öffentlich in der CommonRoad Benchmark Suite zugänglich sind. Die CommonRoad Benchmark Suite wurde im Zuge der vorliegenden Arbeit initiiert, um die Reproduzierbarkeit wissenschaftlicher Ergebnisse zu verbessern. Die CommonRoad Benchmarks für die Bewegungsplanung sind über eine kurze

ID eindeutig identifizierbar, so dass alle notwendigen Details von Experimenten in Publikationen einfach beschrieben werden können. CommonRoad ist frei verfügbar und erweiterbar durch andere Wissenschaftler, um den wissenschaftlichen Fortschritt in der Bewegungsplanung autonomer Fahrzeuge zu beschleunigen.

Acknowledgments

I would like to thank my supervisor Prof. Matthias Althoff for giving me the opportunity to research and pursue a PhD in the field of motion planning for autonomous vehicles. I am grateful for the continuous academic mentoring over the past years, which has greatly influenced my work.

I am happy to have worked in such a great research group and thank all my colleagues at TUM for the countless fun and memorable moments we had together and for the fruitful scientific discussions. I would especially like to thank Dr. Christian Pek and Dr. Markus Koschi for their steady support and valuable feedback on my work. I also thank the students whose theses, seminars, or practical courses I have supervised, and the former HiWis who have supported my work. Special thanks go to Lukas Schäfer for the constructive cooperation over the past years and Vitaliy Rusinov for his contribution to fast collision checking in the reachability analysis. For all the kind support in organizational matters, I would like to thank Amy Bücherl, Dr. Alexander Lenz, Ute Lomp, and Dr. Daniel Renjewski.

My sincere gratitude goes to my family. I am deeply indebted to my parents for their unconditional support at all times and to my sister Katharina for her encouragement in difficult times. I also like to thank my grandmothers Cäcilia and Emma. I am very grateful to my partner Lukas for the unlimited support whenever I needed it and for being a constant source of inspiration. A big thank you goes to my friends, especially Christa, Julia, and Franzi.

Last but not least, I gratefully acknowledge the financial support by the Deutsche Forschungsgemeinschaft (German Research Foundation) within the Priority Programme SPP 1835 Cooperative Interacting Automobiles. I would also like to thank BMW for providing test vehicles for traffic data collection and for supporting our driving tests. In particular, I would like to thank Dr. Sebastian Gnatzig and Dr. Tobias Rehder for their help during our driving tests.

Contents

Chapter 1

Introduction

Autonomous vehicles are envisioned to revolutionize the transportation sector [11]. The expected benefits are numerous including the reduction of traffic injuries and fatalities; time savings since persons are no longer involved in the driving task and can engage in other activities; transportation for persons with restricted mobility; and decrease in vehicle ownership due to a higher utilization of on-demand vehicle sharing [11]. To further enhance the benefits of autonomous driving, wireless communication technologies can be used to connect autonomous vehicles, enabling the explicit exchange and coordination of maneuvers [12]. By orchestrating the motions of a group of connected autonomous vehicles, the capacity of the road infrastructure can be increased, e.g., by traffic shaping [13] or by reducing the inter-vehicle distance [14], resulting in improved traffic flow and shorter commuting times. In addition, connected autonomous vehicles have the potential to further increase traffic safety by enabling collision avoidance maneuvers that would not be possible through individual planning, e.g., when an evasive maneuver in dense traffic requires coordination with surrounding road users, as otherwise a collision would occur.

Given its tremendous potential, the development of autonomous driving has received much attention in the last years. An important aspect in the technical realization of autonomous vehicles is the development of motion planning algorithms. Initially, research has mainly focused on the motion planning for individual vehicles, while cooperative motion planning has recently gained increasing interest. Although remarkable progress has been made, the following research questions remain open:

- How can we efficiently plan drivable, i.e., collision-free and dynamically feasible, motions for autonomous vehicles that reach certain goal states in arbitrary, cluttered environments with multiple static and dynamic obstacles?
- How can we efficiently plan cooperative maneuvers for autonomous vehicles in mixed traffic situations, i.e., situations where human-driven and autonomous vehicles share the road with vulnerable road users such as pedestrians?
- How can we ensure that the planned motions of autonomous vehicles are safe at all times in any traffic situation?
- How can we unify the motion planning for individual vehicles and a group of cooperative vehicles?

The goal of this thesis is the development of methods and algorithms for the holistic solution of the above mentioned research questions. In the following sections, we discuss the challenges of individual, cooperative, and safe motion planning for autonomous vehicles and give an overview of current approaches in the literature that aim to solve these challenges.

1.1 Individual Motion Planning

Automated vehicles need to be able to cope with complex traffic situations, i.e., traffic situations with several static and dynamic obstacles and non-trivial road layouts such as intersections or multi-lane highways. However, both the obstacles and the structure of the road layout may result in the autonomous vehicle only being able to operate in a small subset of its state space, making it difficult to find feasible motions. To operate autonomous vehicles in complex traffic situations, automated driving architectures have been developed that break down the overall problem into more manageable and easier to solve sub-problems. The architectures typically consist of a perception module to generate a model of the current environment; a decision module to plan trajectories given the current environment model and various task-specific constraints; and a control module responsible for tracking the computed trajectory [15]. The decision module is often separated into the route planning, behavioral, and motion planning layer [16]. This work focuses on the motion planning layer which is concerned with the planning of local trajectories that reach a set of goal states starting from the initial state of the autonomous vehicle. The goal states are typically provided by the behavioral layer that is responsible for tactical decision making. Below, we discuss the challenges arising in the field of motion planning in complex traffic situations and review solution approaches found in the literature.

1.1.1 Challenges

Fig. 1.1 illustrates the main challenges of motion planning in complex traffic situations. The autonomous vehicle is approaching a construction site and a pedestrian is predicted to cross the road (see blue areas in Fig. 1.1). The trajectories of the autonomous vehicle are subject to constraints that ensure drivability, i.e., collision avoidance with surrounding obstacles and feasibility in terms of vehicle dynamics, taking into account dynamic and kinematic limits (e.g., minimum and maximum steering angles, acceleration, and curvature changes). In general, the vehicle dynamics are non-linear and collision avoidance with surrounding obstacles is an inherently combinatorial problem. For each obstacle in the environment, different maneuver choices can be taken, such as yielding, overtaking on the left or right side, or following (see Fig. 1.1); and with each additional obstacle in the environment, the number of maneuver options grows exponentially.

Especially in complex traffic situations, it can become increasingly difficult to find drivable motions in time. The main reasons are: (a) it may not be possible to exhaustively evaluate all possible maneuvers under strict real-time constraints due to the high number of maneuver options. (b) Many maneuvers may not be drivable which is often the case in safety-critical situations, i.e., complex traffic situations with small and convoluted solution spaces for motion planning. (c) It might be necessary to detect narrow passageways in the solution space for trajectory planning. As shown in Fig. 1.1, it may be challenging to find a drivable trajectory that avoids the pedestrian, since the space on the road for such a maneuver is rather limited. (d) Some maneuvers may not reach the desired set of goal states (see Fig. 1.1) provided by

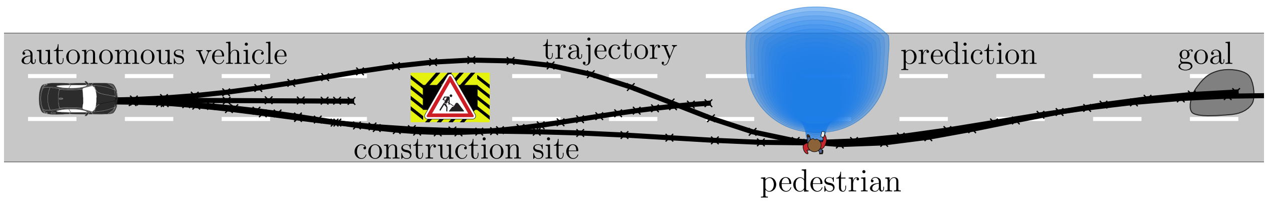


Figure 1.1: The autonomous vehicle has to reach a set of goal states while approaching a construction site and a pedestrian who is likely to cross the road. There are several maneuver options that the autonomous vehicle can select, but not all maneuvers lead to the desired set of goal states.

the behavioral layer. There may also be the possibility that the set of goal states cannot be reached at all, which would require some kind of error handling or default behavior.

In general, the planned trajectories should not only be drivable, but also optimal in terms of a cost function that, for instance, minimizes acceleration or jerk. Since the motion planning problem is generally PSPACE-hard (at least as difficult to solve as an NP-complete problem) [16], it is challenging to find solutions in time. To reliably operate autonomous vehicles in any traffic situation, we require motion planners that generalize well to different and previously unseen traffic situations.

1.1.2 Literature Review

Many techniques for the motion planning of autonomous vehicles have been developed over the past years that aim to solve the aforementioned challenges. Recently, approaches based on machine learning have gained increasing interest, such as reinforcement learning or end-to-end learning [17–19]. In contrast to approaches based on hierarchal architectures for motion planning, end-to-end learning approaches do not distinguish between the perception, decision, and control module, instead they generate the control commands for the autonomous vehicle directly from the sensor data [20]. Current solutions based on machine learning are promising, but their scalability to arbitrary traffic scenarios, safety, and interpretability are still open research questions. Our review on motion planning techniques therefore focuses on graph search, incremental search, and variational methods according to [16]. Further comprehensive overviews of different motion planning techniques for autonomous vehicles can be found in [15, 16, 21, 22].

Graph and Incremental Search Techniques

To plan trajectories for the autonomous vehicle with graph search algorithms like A*or ARA* [23], the state space of the autonomous vehicle is typically discretized by sampling the control or state space. For instance, by forward simulation of a selected vehicle model using sampled control inputs [16], motion primitives can be generated. The obtained motion primitives can be concatenated to create a search graph [24, 25] (see Fig. 1.2). To reduce the computational burden at runtime, motion primitives can also be precomputed offline [26], so that complex vehicle models can be applied. However, the concatenated motion primitives can form an irregular pattern. In the worst case, no final state of one concatenated motion primitive coincides with another, resulting in a huge search tree. To overcome this problem, motion primitives can be constructed so that the resulting graph creates a regular pattern referred

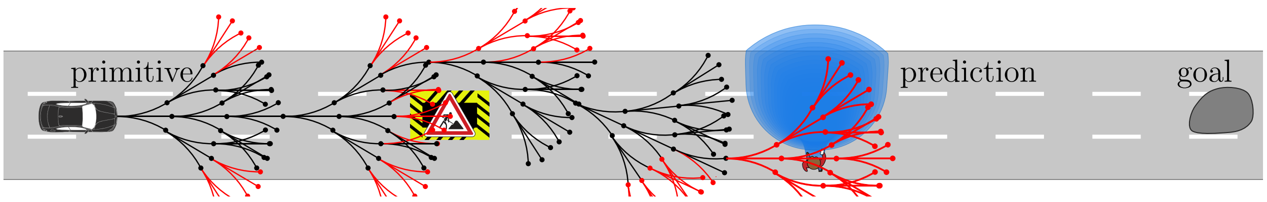


Figure 1.2: Motion primitives are concatenated to find a drivable trajectory for the autonomous vehicle. Motion primitives that lead to a collision with surrounding obstacles are colored in red. Due to the fixed discretization of the graph, there may exist no collision-free combination of motion primitives that evades the pedestrian.

to as state lattice [27–29]. To further reduce the combinatorial complexity, the initial state of the autonomous vehicle can be directly connected to several goal states via geometric curves such as quintic polynomials [30] or circular arcs [31]. The resulting search tree has a single root node to which each goal node is connected by a single edge. A key advantage of graph search techniques is that they naturally explore combinatorial maneuver choices such as overtaking an obstacle to the left or right side. However, graph search techniques are only resolution complete, i.e., a coarse resolution may result in solutions not being found, which is critical when maneuvering in tight spaces. Therefore, the selection of the number of candidate trajectories is challenging: a high number of possible trajectories might be necessary to find a solution, but can lead to a high computational burden.

In contrast to graph search methods, which have a fixed discretization of the search graph, incremental search methods gradually refine the discretization of the state space [16]. Popular representatives of incremental search techniques are rapidly exploring random trees (RRTs) and variants thereof that apply random sampling to efficiently explore complex search spaces [32–34]. Under certain assumptions, RRTs are probabilistically complete [16], which means that the probability of finding an existing solution converges to one if the number of samples approaches infinity. However, computation times are usually unpredictable and can become increasingly higher in cluttered environments, since narrow passageways in the solution space may not be detected in time. In addition, the obtained trajectories are often jagged and therefore uncomfortable for passengers. There exist asymptotically optimal variants of RRTs, e.g., RRT* [34]; however, the convergence rate to optimal solutions is rather slow [35, 36]. Anytime RRT* [37] improves on the computational efficiency of RRT*, but the resulting trajectories tend to be jagged and suboptimal for limited planning times [35]. For a comprehensive overview of further variants of RRTs, we refer to the surveys in [35, 36].

Optimal Control Techniques

Motion planning can also be formulated as a constrained optimal control problem, where a cost function is minimized subject to a set of constraints, e.g., physical and safety constraints. To obtain solutions for the constrained optimal control problem, variational methods, such as direct and indirect methods, can be applied [16]. In contrast to graph search and incremental search approaches, the trajectories are optimized in the continuous state and control space, thus, eliminating undesired discretization effects. However, one major difficulty lies in the handling of collision avoidance constraints that are generally non-convex and non-differentiable, making it difficult to find a collision-free solution efficiently. Therefore, a growing body of literature has investigated various formulations of collision avoidance constraints, which we review below.

Mixed-integer Optimization Current solutions often use mixed-integer programming to model disjunctive decisions, i.e., inequality constraints related by "OR" operators, that arise during maneuver planning [38–40]. Using the so-called Big-M method, logical expressions such as disjunctive constraints can be reformulated as inequalities with binary variables [41]. Thus, binary variables enable the modeling of discrete maneuver decisions in the trajectory optimization such as overtaking an obstacle on the left or right side. Although mixed-integer programs can find globally optimal solutions provided that the underlying continuous subproblems are convex, their application for systems with strict real-time requirements is limited due to their high computational complexity.

Nonlinear Optimization If globally optimal solutions are not required, nonlinear programming can be applied to identify locally optimal solutions. Collision avoidance can then be expressed via the distance or the signed distance function between convex sets such as polyhedra or ellipsoids [42,43] that model the autonomous vehicle and obstacles. While the distance function is zero for intersecting objects, the signed distance function changes sign as soon as objects intersect. Thus, in contrast to the distance function, the signed distance function provides an estimate of the penetration depth of two objects and, therefore, can be used to plan least intrusive trajectories if collisions cannot be avoided. Generally, the (signed) distance function is non-differentiable, but there exist exact and smooth reformulations based on strong duality of convex optimization [42]. However, the resulting collision avoidance constraints are non-convex, and thus, the approach strongly relies on a suitable initial guess. In general, non-convex optimization problems may get stuck in infeasible local minima and no bounds on the number of iterations exist.

Convex Optimization Convex optimization problems can be solved efficiently to global optimality [44]. Typically, a single or a sequence of convex optimization problems are used to approximate the original non-convex optimization problem for trajectory planning see, e.g., [45,46]. The convexification of collision avoidance constraints is often achieved by direct-linearization [47], projection and subsequent linearization [46], convex under-approximation of the feasible set of positions, e.g., using polytopes or ellipsoids [48], or decoupling the longitudinal and lateral vehicle dynamics [45]. However, these methods either require a collision-free initial guess; are conservative with respect to the set of feasible positions, which is problematic in situations that demand precise maneuvering; or simplify the vehicle dynamics.

Driving Corridors Non-linear as well as convex optimization problems can benefit from guidance through the solution space, e.g., in form of driving corridors that represent spatio-temporal constraints (see Fig. 1.3). By determining driving corridors, combinatorial aspects of motion planning are (partially) solved prior to optimization, e.g., the driving corridor in Fig. 1.3 determines that the construction site must be overtaken to the left and the pedestrian on the right. In this way, driving corridors can eliminate infeasible local minima due to obstacles as passing sides are determined prior to optimization and convex under-approximations of the feasible set of positions can be obtained more easily. Often inspired by the concept

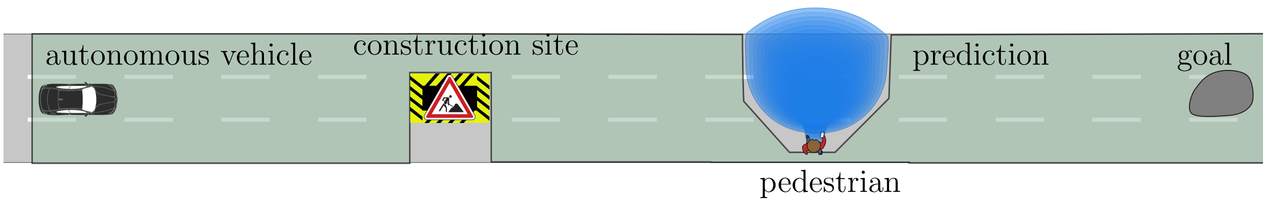


Figure 1.3: A illustration of a driving corridor shown in green that represents spatio-temporal constraints for the motion planning of the autonomous vehicle. For sake of clarity, the time-dependence of the driving corridor is neglected, i.e., the spatio-temporal constraints may change over time, e.g., due to moving road users.

of homotopy or homology [49], there exist topological approaches that identify driving corridors through the decomposition of the collision-free space into (convex) cells; a sequence of cells represents a driving corridor [50–53]. Another line of research determines driving corridors by computing a tunnel around a coarse reference path or trajectory that is obtained through sampling or graph search [54–57]. Within the tunnel, the final trajectory can be optimized. By assigning a passing side to each obstacle, support vector machine optimization can be leveraged to obtain driving corridors [58, 59]. However, current solutions either neglect the vehicle dynamics, suffer from a high combinatorial complexity, inherit drawbacks of discretization-based motion planning methods, cannot decide on the passing side of obstacles, or are only applicable to static environments.

1.2 Cooperative Motion Planning

The ability of autonomous vehicles to behave cooperatively is gaining increasing interest; however, despite this interest, there seems to be no commonly accepted taxonomy of cooperation in the field of autonomous vehicles [12, 60–63]. To categorize this work, we adopt the definitions and taxonomy presented in [12, 62, 64] and discuss their main results below.

1.2.1 Types of Cooperation

In [12, 62, 64], vehicles quantify the usefulness of their cooperative behavior through utility functions and the total utility is defined as a combination of the vehicle's individual utilities. Cooperative behavior is then defined as "[...] a behavior that willingly and knowingly increases the total utility of participating road users in a coupled situation" [64, p. 9]. There are various ways in which road users can show cooperative behavior, with communication playing a central role.

Road users can implicitly communicate with each other meaning that behaviors of other road users are interpreted and utilities are estimated. Cooperation can then be achieved without explicit communication, which is typically referred to as interactive behavior-aware motion planning [20]. These approaches typically do not separate the behavioral and motion planning layers (see Sec. 1.1) to consider the interdependence of the actions of the autonomous vehicle and the reactions of surrounding road users. Often, game-theoretic [65, 66] or probabilistic approaches [67, 68] are leveraged [20].

Road users can also explicitly exchange information, e.g., via vehicle-to-vehicle, vehicle-to-infrastructure, or vehicle-to-everything communication, which is referred to as information-based cooperation [12]. Applications of information-based cooperation are cooperative perception and prediction. Moreover, road users may coordinate their behavior with other road users such that the total utility increases, which is referred to as maneuver-based cooperation [12]. Maneuver-based cooperation with explicit communication allows the vehicles to negotiate coordinated maneuvers. Typically, sensor data, state information, intentions, and possible maneuver options are therefore explicitly communicated [12]. Since explicit communication eliminates the uncertainties about the intentions of communicating road users, thus, contributing to increased road safety, we focus mainly on cooperation using explicit communication.

1.2.2 Challenges

Although maneuver-based cooperation with explicit communication has several advantages, e.g., reduced uncertainties in the prediction of cooperative vehicles or in the perception of the environment, many challenges of individual motion planning also apply to cooperative motion planning (see Sec. 1.1.1). Even if we ignore obstacles in the environment, cooperative motion planning is still an inherently combinatorial problem whose complexity grows exponentially with the number of cooperative vehicles. For instance, if N cooperative vehicles can each perform M admissible actions, we obtain M^N possible maneuvers. Despite this sheer amount of possible maneuvers, cooperative vehicles must be able to find an agreement in time on how to coordinate their driving behavior. This agreement must be adapted to the current traffic situation and take into account the objectives of the involved vehicles. As it will take a considerable amount of time before autonomous vehicles are fully adopted, non-communicating and vulnerable road users must be considered as obstacles in the motion planning. In addition, collision avoidance between cooperative vehicles must be considered.

1.2.3 Literature Review

Several methods for cooperative motion planning have been developed over the years see, e.g., the comprehensive surveys on cooperative intersection management [69–71] and lane changing or merging [71]. Below, we review approaches based on formation control, multi-vehicle trajectory planning, and reservations.

Formation Control

Vehicle platooning has been studied for many years for its potential to improve traffic throughput and reduce fuel consumption [14,72,73]. In general, a platoon is a group of vehicles moving in a linear formation with reduced relative distance [72,73]. Besides the linear formation there are also other ways to arrange a group of vehicles, e.g., wedge shape or diamond shape [74]. The generation of a desired formation and its maintenance while the participating vehicles move along a trajectory can be realized by means of formation control [74, 75]. Existing approaches to formation control can be categorized into behavior-based, leader-follower, and virtual structure approaches [74, 75].

Behavior-based approaches implement different control schemes for each vehicle, such as avoiding obstacles and maintaining the formation to yield the final behavior of a vehicle [76–78]. The implemented control schemes may have conflicting goals, but are executed simultaneously by computing their weighted combination [75]. This approach is suitable for use in cluttered environments; however, a major limitation is the difficulty in proving system stability, since the design of the controller is usually independent of the kinematics or dynamics of the vehicles [74, 75].

In the leader-follower approach, following vehicles aim to maintain a desired distance/ orientation relative to the leading vehicle, which was applied to orchestrate the longitudinal motion of a platoon at a signalized intersection [79], for instance. While the leader-follower approach is widely adopted [75], the main disadvantage of the approach is that a malfunction of the leading vehicle can negatively affect the entire formation [74, 75].

The virtual structure approach considers the spatial relationship between vehicles as rigid structure, e.g., vehicles in wedge shape formation, which was applied for cooperation in multi-lane traffic situations [80], for instance. Due to the inflexibility of the virtual structure regarding deformations, collision avoidance capabilities may be limited [75].

Multi-vehicle Trajectory Planning

If cooperative vehicles are not required to maintain a formation, conflict-free maneuvers can be planned using trajectory planning approaches, e.g., mixed-integer optimization [40,81,82], elastic bands [83], or graph-based methods [64, 84, 85], which we briefly discuss below.

Mixed-integer Optimization Generally, the coordination problem of multiple communicating vehicles can be formulated as a constrained optimal control problem [86]. Similarly to individual motion planning (see Sec. 1.1), we require constraints that ensure the drivability of the planned motions for each vehicle. The overall cost function can be formulated as the weighted sum of the individual cost functions of the cooperative vehicles. Depending on the weighting chosen, costs may be biased in favor of one or a few cooperating vehicles, or the costs of all vehicles may be considered equally weighted. Using mixed-integer programming, collision avoidance among cooperative vehicles and obstacles can be formulated with binary variables [40, 82]. However, the flexibility to plan arbitrary cooperative maneuvers is gained at the cost of high computational effort.

It is also possible to restrict the set of admissible cooperative maneuvers in advance by using a hierarchical maneuver automaton [81]. The top layer is a finite state machine that qualitatively encodes possible maneuvers, such as lane change to the left or right side, overtaking, or lane keeping. The lower layer is a hybrid automaton that describes different phases of such a maneuver, e.g., in [81], three phases are distinguished for the overtaking maneuver based on the relative distances of the vehicles involved. The transitions of these phases are modeled with binary variables. In [81], it is shown that the formulation reduces the worst-time complexity of the optimal control problem, but the approach is currently not applicable to mixed traffic situations.

Elastic Bands Elastic band methods have also been applied to cooperative motion planning [83]. An elastic band consist of several nodes that represent the positions of a cooperative vehicle at discrete time steps. Constraints on the nodes can be formulated using virtual forces, such as a repulsive force to maintain a desired distance from other cooperative vehicles and obstacles. The elastic bands are deformed until an equilibrium is achieved, i.e., the forces vanish at each node. According to [83], the method scales well with the number of cooperative vehicles, but the solutions depend strongly on the initialization of the elastic bands.

Graph-based Methods Cooperative motions can also be planned using graph-based approaches. For example, candidate trajectories for the cooperative vehicles can be obtained by connecting their initial state with sampled target states using polynomials; all possible combinations of their trajectories can be computed and evaluated for drivability and cost [64]. Similar to graph search in individual motion planning, it is also possible to build a tree of motion primitives by simulating the vehicle models of the cooperative vehicles forward with a discretized set of control inputs [84]. At each level of the tree, the composite state and control inputs of the cooperative vehicles are considered. By pre-computations and pruning, computation times can be improved [84]. Since the tree grows rapidly with the number of cooperative vehicles, only coarse discretizations of the time horizon and control inputs of motion primitives are computationally tractable. To reduce the combinatorial complexity of motion planning for multiple vehicles, conflict resolution schemes based on the reservation of road areas can be considered that are discussed next.

Reservation-based Conflict Resolution

Since the space on the road is a shared but limited resource, it seems intuitive to grant individual vehicles exclusive access to certain areas on the road for a limited period of time. A popular mechanism for conflict resolution between several communicating vehicles is therefore space-time reservations, which ensure that a certain spatial region on the road is not occupied by more than one vehicle at any time. Reservation requests can be described in many different ways, e.g., by parts of lanes [87], by discretization of the space on the road into tiles [88], or by critical points in intersection areas [89].

The allocation of reservations can be performed distributed see, e.g., [87], or centralized see, e.g., [88]. In fully distributed approaches, the vehicles rely exclusively on their local knowledge, whereas in fully centralized approaches there is a central unit, e.g., a roadside unit or vehicle, which orchestrates the motions for all vehicles based on the information collected. While fully distributed approaches are usually less computationally intensive and robust against failures [90], the solutions achieved can be suboptimal. In contrast, fully centralized approaches are often computationally intensive with high communication effort and suffer from a single point of failure [90], but globally optimal solutions can be achieved.

There also exist different policies for reservation assignments, e.g., simple approaches such as first-come, first-served [88], but also market-inspired approaches [91–93]. A common mechanism for market-based approaches are auctions [90], where participants can bid for a number of offered items. In the winner determination phase, the allocation of items to bidders is determined. Auctions are especially interesting for conflict resolution in traffic, as individual goals of vehicles, but also common goals can be considered. Furthermore, auction-based approaches are often hybrid approaches [90], i.e., parts of the calculation can be distributed over several vehicles so that the advantages of distributed approaches are preserved, but decisions are made by a central unit so that the solutions found may be of higher quality than with distributed approaches.

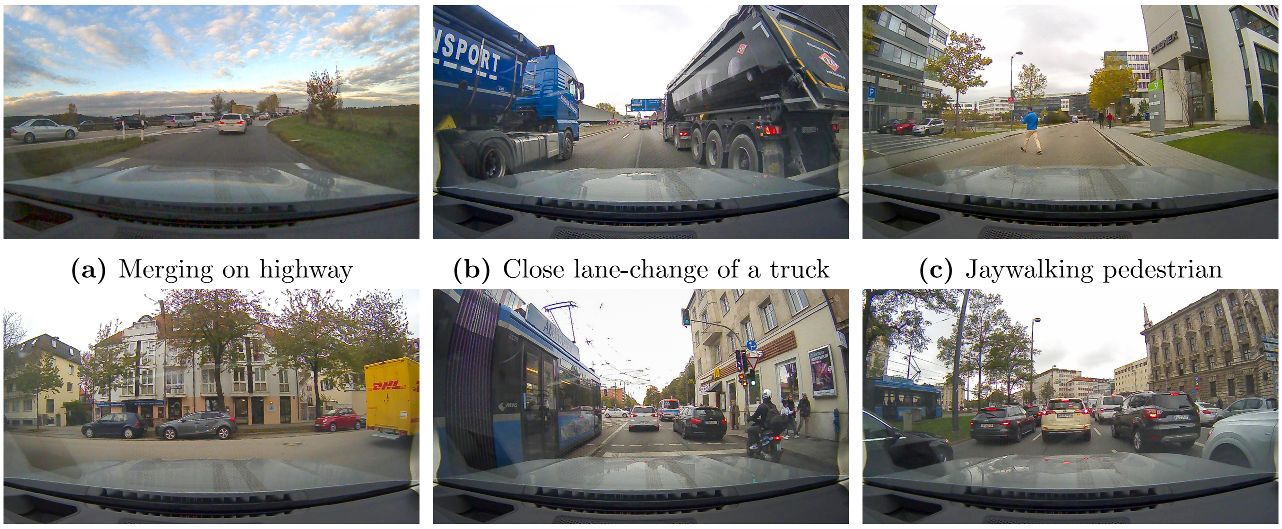
1.3 Safe Motion Planning

The full benefits of autonomous driving will only be realized when autonomous vehicles, whether they are driving individually or within a group of vehicles, are able to make safe decisions in any traffic situation at any time. What challenges must be overcome to ensure the safety of autonomous vehicles?

1.3.1 Challenges

In [94], three criteria for motion safety were introduced. These state that the autonomous vehicle must reason (a) over an infinite time horizon to decide its own future motion, taking into account (b) its own dynamics and (c) the future behavior of obstacles in its environment [94]. If any of these criteria is violated, safety cannot be ensured. For instance, if its is ignored that the autonomous vehicle is not able to stop instantaneously due to its dynamics, a collision may occur. Similarly, a collision may occur if it is incorrectly assumed that all obstacles in the environment are static. Furthermore, if only a limited time horizon is considered for motion planning, a collision may happen just outside this time horizon. Since the motions of the autonomous vehicle are typically planned over finite time horizons due to computational reasons, the motion planner has to consider at least the time horizon until the autonomous vehicle can reach a set of safe goal states [94]. This safe set of goal states needs to ensure persistent feasibility, i.e., motion planning is recursively feasible.

As the future evolution of a traffic scene is unknown and there are an infinite number of unique real-world traffic scenarios that an autonomous vehicle could potentially encounter (see, e.g., scenarios in Fig. 1.4), it is challenging to meet all safety criteria, especially in the light that the autonomous vehicle has only limited time to make decisions.



(a) Merging on highway **(b)** Close lane-change of a truck **(c)** Jaywalking pedestrian

(d) Occluded intersection **(e)** Green light at busy intersection **(f)** Dense traffic

Figure 1.4: Traffic scenarios recorded in the cities of Munich and Garching-Hochbrück, Germany, on October 25, 2018, between 1 p.m. to 6 p.m.

1.3.2 Literature Review

Until now, vehicle safety has been based primarily on simulation and testing prior to deployment. Simulation techniques can provide counterexamples, i.e., situations in which unsafe decisions were taken, but can only cover a finite number of test cases. Moreover, a recent study [95] has revealed that a fleet of 100 autonomous vehicles would have to be driven tens of years to demonstrate their reliability. Thus, relying solely on testing is impractical. As only a finite number of test cases can be covered, both techniques cannot provide sufficient safety guarantees [20, 95].

In contrast to simulation and testing, formal verification can be applied to mathematically prove the safety of autonomous vehicles. To this end, a formal specification is needed that defines requirements to be met by the autonomous vehicle. Typically, the formal specification relies on the concepts of safe and unsafe sets of states for the motion planning of autonomous vehicles that depend on the future motion of other road users. Before we elaborate on different techniques of formal verification, we begin with a brief overview of motion predictions of other road users, followed by the introduction of different concepts of safe and unsafe sets of states.

Motion Prediction

Since the actual future behavior of obstacles is unknown, we can only estimate their future behavior in terms of motion predictions. A comprehensive survey on motion prediction in the context of autonomous driving is given in [96]. Many approaches compute most likely behaviors of other road users, e.g., using probabilistic or machine learning methods [97–100]. By planning motions for the autonomous vehicle based on these predictions, the risk of collisions can be reduced. However, if road users deviate from their predicted behavior—which often occurs in real traffic—collisions can no longer be ruled out.

To guarantee strict collision avoidance without residual risks, set-based reachability analysis can be leveraged to compute all possible future behaviors of other road users [101–103]. Reachability analysis computes the set of states that a system can reach at a given time considering its initial state and dynamics. However, the reachable sets grow rapidly over time as more positions become reachable. Therefore, assumptions or specifications must be made about the possible future behaviors of other road users, otherwise the maneuverability of the autonomous vehicle will be severely limited.

Safe and Unsafe Sets

Based on the motion prediction of obstacles, unsafe sets of states can be computed using the concept of inevitable collision states. Inevitable collision states [104, 105] are states where the autonomous vehicle cannot avoid a collision with obstacles regardless of the future trajectory. Thus, by keeping the autonomous vehicle outside of inevitable collision states at all times, collision avoidance can be guaranteed because there exist at least one trajectory that is persistently feasible for an infinite time horizon.

In contrast, it is also possible to compute safe sets of states based on motion predictions that are invariant sets [106, 107]. Invariant sets represent states in which the system is safe for an infinite time horizon. Since the computation of inevitable collision states and invariant sets is often costly, the work of [108] proposes to compute invariably safe states. Invariably safe states are subsets of the collision-free states that guarantee that a collision-free trajectory to another set of invariably safe sets exists for an infinite time horizon. It is shown that an under-approximation of invariably safe states based on formal safe distances for vehicle following and evasive distances can be efficiently computed. To ensure that the autonomous vehicle always stays within the computed safe set of states or outside an unsafe set of states, we can use formal methods which we discuss next.

Formal Methods

Popular techniques of formal verification are model checking, theorem proving, correct-by-construction controller, and set-based reachability analysis that we review below.

Model checking [109–111] verifies whether a formal specification holds for a finite-state model of a system by exhaustively exploring the state space of that model. Thus, it is possible to provide counterexamples, i.e., states in which the formal specification is violated. However, model checking techniques inherently suffer from the curse of dimensionality and, therefore, the computational effort increases considerably for high-dimensional state spaces. Another technique is theorem proving [112, 113] that relies on logic and mathematical reasoning to prove that a desired system property holds. Theorem proving is applicable to high-dimensional systems, but due to its complexity, theorem proving typically requires user interactions.

It is also possible to automatically synthesize controllers that are provably correct [114,115], i.e., it is guaranteed that the system fulfills the formal specification under the computed control law. Thus, it is possible to keep the autonomous vehicle in a safe set of states. It is common to use a hierarchical approach to controller synthesis [114], which typically works with a discrete abstraction of the environment. Another line of research uses set-based reachability analysis to guarantee collision avoidance [101, 116, 117]. By computing the reachable set of the autonomous vehicle, it can be verified whether unsafe states in the environment can be reached. The main challenges are the generalization to arbitrary traffic scenarios and the computational efficiency.

Using formal methods, strict safety guarantees can be given. However, it is still impractical to aim for absolute safety, i.e., the autonomous vehicle is never involved in any accident regardless of what happens. The reason for this is that the autonomous vehicle may not be able to avoid collisions due to intentional malicious behavior of other vehicles, such as when another vehicle deliberately provokes a rear-end collision. Therefore, many works rely upon a relaxation of absolute safety and are dedicated to eliminating self-inflicted accidents of autonomous vehicles. We therefore proceed with the discussion of various safety specifications for the prevention of self-inflicted accidents.

Safety Specifications

Self-inflicted accidents by autonomous vehicles must be ruled out, as they can lead to serious personal injury and property damage, causing enormous economic losses. With this in mind, much work has been devoted to preventing self-inflicted accidents of autonomous vehicles [101, 118–120]. There are three key concepts that are responsibility-sensitive safety [119], not-at-fault driving [118], and legal safety [101, 120].

Responsibility-sensitive safety [119] formalizes common sense rules for vehicles, such as that it is illegal to hit someone from behind or take the right of way. Based on safe distances in the longitudinal and lateral directions, proper responses by the autonomous vehicle are defined. If all vehicles behave according to the proper responses, no collisions will occur.

In [118], not-at-fault driving is introduced, in which the autonomous vehicle is required to avoid collisions with surrounding obstacles while moving. If the autonomous vehicle is stationary, it is always considered to be not-at-fault. The approach in [118] assumes that a conservative prediction is given, which always includes the real behavior of obstacles. Based on the conservative prediction and offline computed reachable sets of the autonomous vehicle, trajectories are planned that are provably not-at-fault. Therefore, each trajectory is divided into three phases, namely the moving phase, the braking phase, and the stopping phase. The braking phase represents a fail-safe maneuver that brings the autonomous vehicle to a standstill in hazardous situations.

In legal safety [101, 120], the motions of the autonomous vehicle must be collision-free against all legal behaviors of other road users. The legal behaviors of other road users are based on traffic rules, e.g., the Vienna Convention on Road Traffic [121] may serve as a reference [120]. Legal safety also stipulates that the autonomous vehicle must avoid or mitigate collisions with road users that violate traffic rules. In safety-critical situations, the autonomous vehicle should come to a stop in dedicated areas on the road, e.g., emergency lanes on highways [120].

1.4 Contributions

This thesis proposes a holistic motion planning method for autonomous vehicles that (a) handles complex traffic scenarios with multiple static and dynamic obstacles, (b) can be used for individual and cooperative motion planning with only slight modifications, and (c) is particularly suitable for planning safe motions in critical traffic situations. The key idea is to efficiently represent the collision-free space of feasible motions to eliminate the disadvantages of existing optimization-based trajectory planning methods (see Sec. 1.1.2). By using set-based reachability analysis, our proposed approach computes the collision-free reachable states of the autonomous vehicle over time. To obtain a computationally efficient representation of collision avoidance constraints for the trajectory optimization, we identify dynamics-aware driving corridors within the reachable sets. From these driving corridors, collision avoidance constraints are extracted.

By combining existing optimization-based trajectory planning algorithms with our approach, several advantages are offered. The reachable sets of the autonomous vehicle are computed online, allowing us to explicitly consider each traffic scenario on-the-fly, i.e., during the operation of the autonomous vehicle. Since the computation of the collision-free reachable sets is based on simple collision detection methods, different obstacle representations such as circles or polygons can be chosen, making our approach applicable in arbitrary traffic scenarios, i.e., traffic scenarios with different road geometries and number and type of obstacles. As we use set-based reachability analysis, the computational effort of our approach typically decreases with increasing criticality of the traffic scenario, i.e., with shrinking solution space for trajectory planning. Moreover, even narrow passageways in the solution space can be detected which can be of utmost importance in safety-critical situations.

We conceptualized driving corridors in such a way that the collision avoidance constraints extracted from them can be integrated into arbitrary gradient- and Hessian-based solvers.

The obtained collision avoidance constraints can be either represented by convex keep-out zones or polyhedral sets of feasible positions. Depending on the choice of representation, various existing trajectory planners based on nonlinear programming as well as (successive) convexification can be leveraged. This also entails that various vehicle models can be used for motion planning, even models that consider the combined longitudinal and lateral dynamics of the vehicle. The proposed driving corridors facilitate the search for a suitable initial guess for nonlinear optimization problems and eliminate local minima induced by obstacles. The driving corridors can be restricted to end in a predefined set of goal states, e.g., standstill on the shoulder lane, allowing one to reason over infinite time horizons (see Sec. 1.1.1).

We have implemented the proposed motion planning method as a prototype in C++ and Python for use in a BMW 7-series test vehicle and validated successfully the drivability of planned motions. In simulation, we have demonstrated that by augmenting existing optimization-based motion planners with our computation of driving corridors, infeasible local minima due to obstacles are eliminated. We have further compared our motion planning method to state-of-the-art planning algorithms based on sampling and mixed-integer programming, showing that our method has reliable runtimes and identifies narrow passageways without increased computational effort.

The unique characteristics of our approach make it particularly useful in safety-critical situations. We therefore extended previous work on safe motion planning [45,122] and safety verification [101] by integrating the computation of driving corridors into the fail-safe trajectory planner proposed in [45], which provides safe fallback trajectories for safety-critical situations. The integration of driving corridors facilitates the efficient computation of fail-safe trajectories in arbitrary traffic situations and the consideration of multiple safe sets of states. The modified fail-safe trajectory planner is embedded into a safety layer for existing motion planning frameworks to guarantee legal safety (see Sec. 1.3.2), i.e., autonomous vehicles never cause accidents even though other road users may legitimately perform any kind of behavior in compliance with traffic rules. The safety layer verifies online intended motions generated by an existing motion planner of the autonomous vehicle and provides fail-safe trajectories in dangerous situations using the presented motion planning approach. As a result, the safety layer prevents the autonomous vehicle from causing accidents at all times independent of the nominal motion planner, e.g., even motion planning methods based on machine learning can be applied.

We have successfully validated the proposed safety layer in critical urban traffic situations that we recorded in real traffic in the area of Munich. The recorded traffic scenarios include measurement uncertainties and are publicly available for usage by other researchers and interested parties. We have also successfully demonstrated the general applicability of the proposed safety layer by testing it with three different intended motion planners. One motion planner even ignored other road users, but the experiments confirm that our approach ensures legal safety at all times. Overall, the results indicate that a non-conservative and safe driving behavior can be achieved with the proposed safety layer.

Our proposed motion planning method can be extended to enable maneuver-based cooperation among communicating vehicles. We therefore consider loosely coupled groups of communicating vehicles with planning horizons of a few seconds in mixed traffic situations. Non-communicating road users are treated as obstacles, while the motions of communicating vehicles are planned jointly. Our cooperation mechanism is based on space-time reservations so that vehicles can have exclusive access to areas on the road for a limited time. The space-time reservations are negotiated among the cooperative vehicles based on their reachable sets. To perform the negotiations, we have developed two different schemes: the first scheme aims at distributing the space on the road fairly, i.e., all vehicles have equally large areas on the road available for motion planning. The second scheme is based on combinatorial optimization inspired by auctions, which enables to take into account individual objectives of cooperating vehicles. The resulting space-time reservations are represented by subsets of the reachable sets of cooperative vehicles. These subsets can then be used for motion planning as in the single-vehicle case; thus, making it easy to transition from cooperative to non-cooperative driving.

The methods developed in this work were tested with scenarios from the CommonRoad benchmark suite available at **commonroad.in.tum.de**, which was co-initiated by the author of this work. CommonRoad provides researchers with a platform to test and compare their motion planning algorithms, improving the reproducibility of experimental results. Since the project start in 2017, many people have contributed to CommonRoad. Several open source software tools for motion planning of autonomous vehicles have been developed, e.g., the CommonRoad-IO package, which provides methods for reading, writing and visualizing CommonRoad scenarios and planning problems, and the CommonRoad Drivability Checker, which unifies tests for collision avoidance, kinematic feasibility, and road-compliance to ensure the drivability of planned motions for autonomous vehicles. As CommonRoad is an open source project, it can be easily used and extended by other researchers.

1.5 Publications and Outline

This thesis is based on six peer-reviewed, first-author publications that have been published in international journals or conferences [1–6]:

[1] C. Pek*, **S. Manzinger***, M. Koschi*, and M. Althoff, “Using online verification to prevent autonomous vehicles from causing accidents,” *Nature Machine Intelligence*, vol. 2, no. 9, pp. 518–528, 2020.

[2] **S. Manzinger**, C. Pek, and M. Althoff, “Using reachable sets for trajectory planning of automated vehicles,” *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 232–248, 2021.

[3] L. Schäfer*, **S. Manzinger***, and M. Althoff, “Computation of solution spaces for optimization-based trajectory planning,” *IEEE Transactions on Intelligent Vehicles*, 2021, [early access].

[4] **S. Manzinger** and M. Althoff, “Tactical decision making for cooperative vehicles using reachable sets,” in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2018, pp. 444–451.

[5] **S. Manzinger** and M. Althoff, “Negotiation of drivable areas of cooperative vehicles for conflict resolution,” in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2017, pp. 1–8.

[6] M. Althoff*, M. Koschi*, and **S. Manzinger***, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 719–726.

* *These authors have contributed equally and share the first authorship.*

The publications [1–6] are included in this thesis together with a short summary of the content and a description of the contributions of all authors. The following peer-reviewed publications of the author from related research are excluded:

[7] C. Pek, V. Rusinov, **S. Manzinger**, M. C. Üste, and M. Althoff, "CommonRoad Drivability Checker: Simplifying the development and validation of motion planning algorithms," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2020, pp. 1013–1020.

[8] A. Zhu, **S. Manzinger**, and M. Althoff, "Evaluating location compliance approaches for automated road vehicles," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 642–649.

[9] **S. Manzinger**, M. Leibold, and M. Althoff, "Kooperative Bewegungsplanung autonomer Fahrzeuge unter Verwendung von Manöver-Templates," in *AAET - Automatisiertes und vernetztes Fahren*, ITS automotive nord e.V., Ed., 2017, pp. 348–367.

[10] ——, "Driving strategy selection for cooperative vehicles using maneuver templates," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 647–654.

The remainder of this thesis is structured as follows: Chapter 2 introduces the necessary mathematical preliminaries and gives a brief overview of the novel methods proposed in this thesis. Chapter 3 finishes with conclusions and outlines possible directions for future work to further improve the performance of the proposed methods. Appendix A contains the reprint of each included publication and Appendix B lists the Bachelor and Master theses supervised by the author of this work.

Chapter 2

Methods

2.1 Preliminaries

Let us introduce the compact set $\mathcal{X} \subset \mathbb{R}^{n_x}$ of admissible states $x \in \mathcal{X}$ and the set $\mathcal{U} \subset \mathbb{R}^{n_u}$ of admissible control inputs $u \in \mathcal{U}$ of an autonomous vehicle, whose motion is governed by

$$x_{k+1} = f(x_k, u_k), \tag{2.1}$$

where the system dynamics $f(x_k, u_k)$ is a continuously differentiable nonlinear function on $\mathcal{X}$. The discrete time step $k \in \mathbb{N}_0$ corresponds to the time $t_k = k\Delta t$, with the time increment $\Delta t \in \mathbb{R}^+$. Although the vehicle dynamics is an inherently continuous-time system, it is sufficient for us to model it as a discrete-time system. This is because we formulate the trajectory planning problems as discrete-time optimal control problems, as will be shown later in Sec. 2.3 and Sec. 2.4. A possible state and input trajectory of the system is denoted by $x(\cdot)$ and $u(\cdot)$, respectively. Without loss of generality, we assume that $t_0 = 0$ and the planning horizon $t_h \in \mathbb{R}^+$ is finite.

In the literature, the reachable set of a system is usually referred to as the set of states that can be reached from an initial set of states $\mathcal{X}_0$ considering all admissible input trajectories $u(\cdot)$. In this work, we use an extended definition of the reachable set and consider only those states that do not enter a set of forbidden states. After introducing the occupancy operator, we define the set of forbidden states of the autonomous vehicle.

Definition 1 (Occupancy) The operator $\mathrm{occ}(x)$ relates the state $x \in \mathcal{X}$ to the set of points in the position domain occupied by the autonomous vehicle as $\mathrm{occ} : \mathcal{X} \to \mathbb{P}(\mathbb{R}^2)$, where $\mathbb{P}(\mathbb{R}^2)$ denotes the power set of $\mathbb{R}^2$.

Definition 2 (Set of Forbidden States) Given the set $\mathcal{O}_k$ of occupied positions of all obstacles (e.g., other cars and pedestrians) including the space outside of the road, the set of forbidden states at time k is

$$\mathcal{F}_k = \{x_k \in \mathcal{X} \mid \mathrm{occ}(x_k) \cap \mathcal{O}_k \neq \emptyset\}.$$

The reachable set of the autonomous vehicle is then defined as:

Definition 3 (Reachable Set) Let us introduce $\mathcal{R}^{\mathrm{e}}_0 = \mathcal{X}_0$ as the set of collision-free initial states of the autonomous vehicle including measurement uncertainties. The reachable set $\mathcal{R}^{\mathrm{e}}_{k+1}$ at time step $k+1$ is the set of states that can be reached within one time step from $\mathcal{R}^{\mathrm{e}}_k \subseteq \mathcal{X}$ without intersecting the set of forbidden states $\mathcal{F}_{k+1}$:

$$\mathcal{R}^{\mathrm{e}}_{k+1} = \left\{ x_{k+1} \in \mathcal{X} \,\middle|\, \exists x_k \in \mathcal{R}^{\mathrm{e}}_k,\, \exists u_k \in \mathcal{U} \colon x_{k+1} = f(x_k, u_k) \,\wedge\, x_{k+1} \notin \mathcal{F}_{k+1} \right\}.$$

We further introduce the projection operator that projects a state x onto the position domain to obtain the drivable area of the autonomous vehicle.

Definition 4 (Projection) The operator $\mathrm{proj} : \mathcal{X} \to \mathbb{R}^2$ maps the state $x \in \mathcal{X}$ to the position domain. Using the same notation, we project a set of states $\mathcal{X}$: $\mathrm{proj}(\mathcal{X}) := \{\mathrm{proj}(x) \,|\, x \in \mathcal{X}\}$.

Definition 5 (Drivable Area) The drivable area $\mathcal{D}_k^{\mathrm{e}}$ at time step k is defined as the projection of the reachable set onto the position domain, i.e., $\mathcal{D}_k^{\mathrm{e}} := \mathrm{proj}(\mathcal{R}_k^{\mathrm{e}})$.

2.2 Reachability Analysis

In this work, reachable sets are used to explore the collision-free state space of the autonomous vehicle in cluttered environments. For this purpose, set-based reachability analysis is particularly suitable since even narrow passages can be detected by its calculation in the continuous state space. However, the computation of the exact reachable set $\mathcal{R}^{\mathrm{e}}$ is generally only possible for certain classes of systems [123, Ch. 3]. Moreover, the non-linear vehicle models used for motion planning and their rather high-dimensional state space make it difficult to calculate the reachable set under hard real-time constraints. We therefore aim to compute accurate approximations $\mathcal{R}$ and $\mathcal{D}$ of the exact reachable set $\mathcal{R}^{\mathrm{e}}$ and drivable area $\mathcal{D}^{\mathrm{e}}$, i.e., $\mathcal{R} \approx \mathcal{R}^{\mathrm{e}}$ and $\mathcal{D} \approx \mathcal{D}^{\mathrm{e}}$, to achieve computational feasibility, which is explained below.

For the reachable set computation, the vehicle dynamics is approximated by two second-order integrator models in the road-aligned coordinate system F^{L}. The state $x = (s_\zeta, v_\zeta, s_\eta, v_\eta)^T$ and input $u = (a_\zeta, a_\eta)^T$ of the system are composed of the position s, velocity v, and acceleration a in the longitudinal ζ- and lateral η-directions, where both the velocity and acceleration are bounded:

$$x_{k+1} = \begin{pmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{pmatrix} x_k + \begin{pmatrix} \frac{1}{2}\Delta t^2 & 0 \\ \Delta t & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ 0 & \Delta t \end{pmatrix} u_k, \tag{2.2a}$$

$$\underline{v}_\zeta \leq v_{\zeta,k} \leq \overline{v}_\zeta, \quad \underline{v}_\eta \leq v_{\eta,k} \leq \overline{v}_\eta, \tag{2.2b}$$

$$\underline{a}_\zeta \leq a_{\zeta,k} \leq \overline{a}_\zeta, \quad \underline{a}_\eta \leq a_{\eta,k} \leq \overline{a}_\eta, \tag{2.2c}$$

where $\underline{\square}$ and $\overline{\square}$ denote the minimum and maximum value of a variable $\square$, respectively.

In the absence of obstacles, a closed-form solution for the reachable set of a second-order integrator model exists; however, the presence of arbitrarily shaped obstacles requires numerical computations [124]. We therefore approximate the reachable set $\mathcal{R}_k^{\mathrm{e}}$ with the union of base sets $\mathcal{R}_k^{(i)}$, $i \in \mathbb{N}_0$. A base set $\mathcal{R}_k^{(i)}$ is the Cartesian product of two convex polytopes $\mathcal{P}_{\zeta,k}^{(i)}$ and $\mathcal{P}_{\eta,k}^{(i)}$ in the (s_ζ, v_ζ) and (s_η, v_η) plane [124], respectively, so that

$$\mathcal{R}_k^{\mathrm{e}} \approx \mathcal{R}_k := \bigcup_i \mathcal{R}_k^{(i)}, \text{ with } \mathcal{R}_k^{(i)} = \mathcal{P}_{\zeta,k}^{(i)} \times \mathcal{P}_{\eta,k}^{(i)}. \tag{2.3}$$

There are indeed other options to represent the reachable set, such as ellipsoids, zonotopes, or oriented rectangular hulls, to name a few possible alternatives. Yet, we select polytopes as they are a convex and explicit set representation that is closed under the required operations such as Minkowski sum, linear mapping, and intersection. The projection of a base set $\mathrm{proj}(\mathcal{R}_k^{(i)})$ onto the position domain yields an axis-aligned rectangle $\mathcal{D}_k^{(i)}$ representing reachable positions. The union of $\mathcal{D}_k^{(i)}$ approximates the drivable area $\mathcal{D}_k^{\mathrm{e}}$: $\mathcal{D}_k^{\mathrm{e}} \approx \mathcal{D}_k := \bigcup_i \mathcal{D}_k^{(i)}$. To simplify notation, we refer to both the union $\bigcup_i \mathcal{R}_k^{(i)}$ and the collection $\{\mathcal{R}_k^{(0)}, \mathcal{R}_k^{(1)}, \ldots\}$ of base sets $\mathcal{R}_k^{(i)}$ as $\mathcal{R}_k$; this is done analogously for the drivable area $\mathcal{D}_k$.

The initial reachable set is $\mathcal{R}_0 = \mathcal{R}_0^{(0)}$, where $\mathcal{R}_0^{(0)}$ encloses the set of initial states of the autonomous vehicle including measurement uncertainties. In order to determine the reachable set of consecutive time steps, the following procedure is performed iteratively [2, 124]: first, the base sets $\mathcal{R}_k^{(i)}$ of the previous time step k are propagated according to the system model (2.2). We denote the propagated reachable set with $\mathcal{R}_{k+1}^{\mathrm{prop}}$. Second, the set of forbidden states $\mathcal{F}_{k+1}$ is removed from the propagated reachable set $\mathcal{R}_{k+1}^{\mathrm{prop}}$. Since $\mathcal{R}_{k+1}^{\mathrm{prop}} \setminus \mathcal{F}_{k+1}$ cannot be represented by convex polytopes in general, we approximate the result. Due to the removal of forbidden states, a base set $\mathcal{R}_{k+1}^{(j)}$ may be reachable from multiple base sets $\mathcal{R}_k^{(i)}$ within one time step. For later use, we store this relationship in a directed graph $\mathcal{G}_{\mathcal{R}}$, in which a node corresponds to a base set $\mathcal{R}_k^{(i)}$ and an edge indicates that $\mathcal{R}_k^{(i)}$ reaches $\mathcal{R}_{k+1}^{(j)}$. For a more detailed description of the algorithm, the reader is referred to [2,124] (see also Appendix A.2).

2.3 Motion Planning Using Reachable Sets

The trajectory planning problem of the autonomous vehicle is defined as:

Problem 1 (Trajectory Planning) Find an optimal control $u^*(\cdot) \in \mathcal{U}$ and state trajectory $x^*(\cdot) \in \mathcal{X}$ that solve the following non-convex optimization problem:

$$\min_{u(\cdot)} \sum_{k=0}^{h} J(x_k, u_k) \tag{2.4a}$$

such that

$$x_0 = \tilde{x}_0,\ x_h \in \mathcal{X}_{\mathrm{goal}}, \tag{2.4b}$$

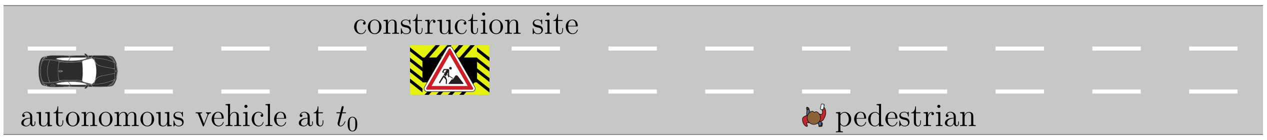
$$\forall k \in \{0, \ldots, h-1\}:\ x_{k+1} = f(x_k, u_k), \tag{2.4c}$$

$$\forall k \in \{0, \ldots, h\}:\ g(x_k, u_k, k) \leq 0, \tag{2.4d}$$

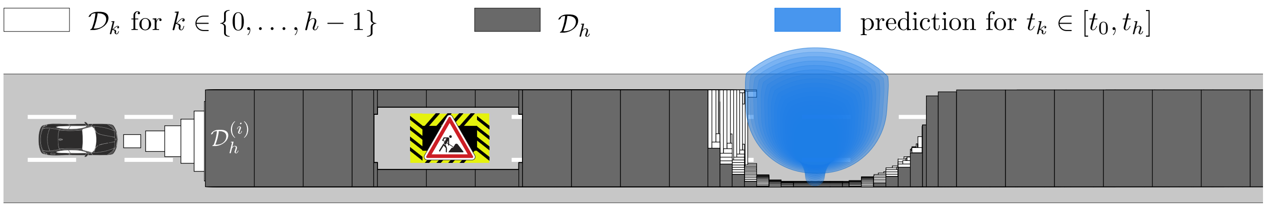
$$\mathrm{proj}(x_k) \in \mathcal{D}_k, \tag{2.4e}$$

where $\tilde{x}_0$ denotes the measured initial state and the cost function $J : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ is continuously differentiable. The dynamics (2.4c) is subject to the continuously differentiable, time-variant constraints $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{N}_0 \to \mathbb{R}^{n_g}$ (2.4d), where n_g denotes the number of constraints. Examples of such constraints are acceleration limitations and minimum and maximum bounds on the steering angle. Collision avoidance is represented by the non-convex, non-differentiable constraint (2.4e), i.e., the positions of the autonomous vehicle are limited by the collision-free drivable area at each time step k.

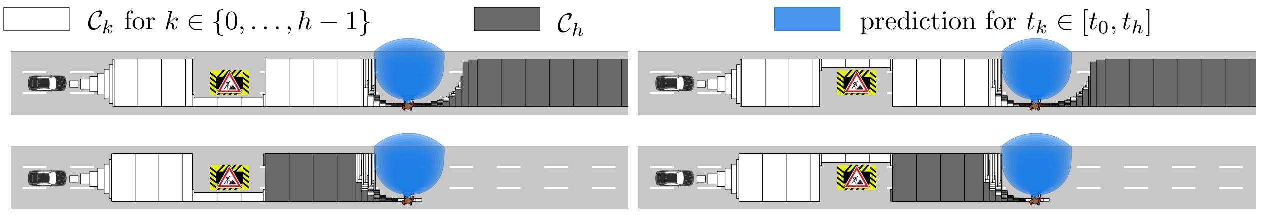
A feasible solution for Problem 1 can be found using non-convex or convex optimization techniques (see Chapter 1.1.2). As outlined in Chapter 1, it can be advantageous or even necessary for both convex and non-convex optimization to provide guidance through the



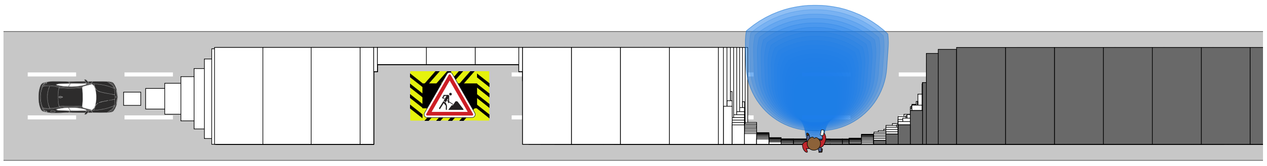
(a) The autonomous vehicle approaches a construction site and a pedestrian steps onto the road.


$\mathcal{D}_k$ for $k \in \{0, \dots, h-1\}$
$\mathcal{D}_h$
prediction for $t_k \in [t_0, t_h]$
$\mathcal{D}_h^{(i)}$

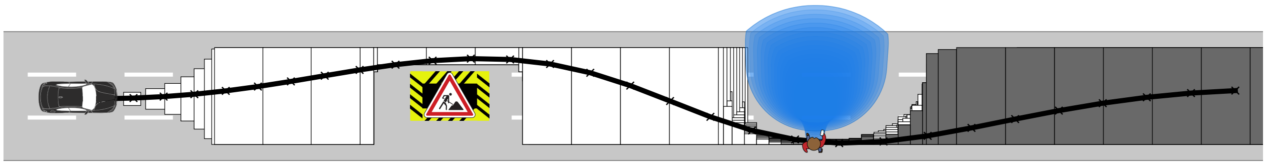
(b) We compute the drivable area $\mathcal{D}_k$ of the autonomous vehicle for consecutive points in time t_k, $k \in \{0, \dots, h\}$, to determine the collision-free solution space for trajectory planning. The drivable area at the final time t_h is colored in gray.


$\mathcal{C}_k$ for $k \in \{0, \dots, h-1\}$
$\mathcal{C}_h$
prediction for $t_k \in [t_0, t_h]$

(c) Identification of different driving corridors $\mathcal{C}$ within the drivable area



(d) Selection of a suitable driving corridor.



(e) Extraction of collision avoidance constraints from the driving corridor and trajectory optimization.

Figure 2.1: General procedure when using reachable sets for the motion planning of autonomous vehicles in cluttered environments. The autonomous vehicle is heading towards a construction site while a pedestrian is predicted to cross the road.

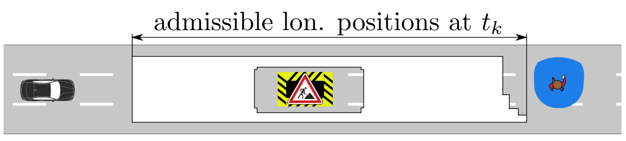
collision-free solution space. Since the obtained drivable area $\mathcal{D}_k$ is typically non-convex and often disconnected due to the presence of obstacles, we decompose it into driving corridors. The resulting driving corridors $\mathcal{C}$ are temporal sequences of connected sets $\mathcal{C}_k$ that are subsets of the drivable area, i.e., $\mathcal{C}_k \subseteq \mathcal{D}_k$. There may exist several driving corridors; however, we refrain from introducing an additional identifier to differentiate between various driving corridors in order to improve readability.

The general procedure for the motion planning using reachable sets is composed of the following steps (see Fig. 2.1):

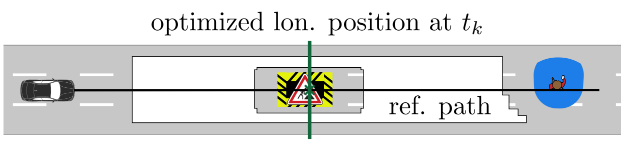
1. Exploration of the non-convex solution space for trajectory planning by computing reachable sets over consecutive time steps (see Fig. 2.1b);
2. Identification of driving corridors within the reachable sets (see Fig. 2.1c);
3. Selection of a suitable driving corridor for trajectory planning (see Fig. 2.1d);
4. Extraction of collision avoidance constraints from the selected driving corridor;
5. Optimization-based trajectory planning using the collision-avoidance constraints obtained from the driving corridor (see Fig. 2.1e) that substitute (2.4e).

In the course of this work, we have developed two alternative methods to obtain driving corridors that can be combined with various state-of-the-art motion planning methods (see Figs. 2.2 and 2.3). We first summarize the main concept of each method and then briefly compare both methods.

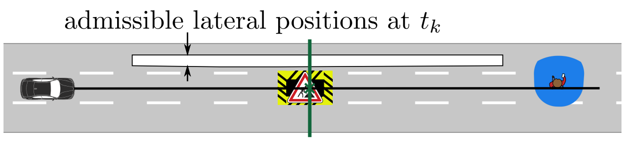
In [2] (see also Appendix A.2), Problem 1 is convexified by describing the motion of the autonomous vehicle with respect to a predefined reference path and linearization. This enables the decoupling of the longitudinal and lateral vehicle dynamics as shown in [45, 125]. The longitudinal dynamics are represented by a fourth-order integrator model with bounded velocity, acceleration, and jerk; the lateral dynamics is given by a linearized kinematic single-track model with limits on the drivable curvature and change of curvature. Due to the decoupling of the kinematics, we define a driving corridor for both the longitudinal and lateral trajectory planning and refer to the driving corridors as longitudinal and lateral driving corridors. The collision avoidance constraints extracted from the driving corridors are formulated as admissible position intervals for the longitudinal or lateral trajectory optimization (see Figs. 2.2a



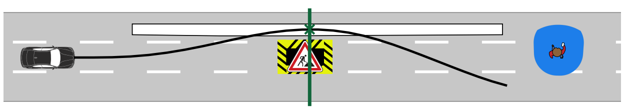
(a) The longitudinal driving corridor is a temporal sequences of connected sets from which we obtain the admissible longitudinal positions.



(b) The longitudinal trajectory is optimized subject to the box constraints of the longitudinal driving corridor.

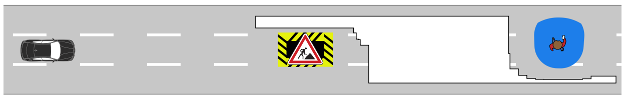


(c) A lateral driving corridor is a temporal sequence of connected sets with unique passing side.

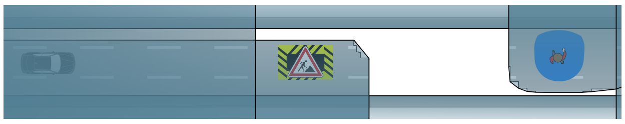


(d) The lateral trajectory is optimized subject to the box constraints of the lateral driving corridor.

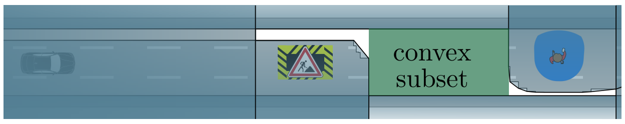
Figure 2.2: Main computation steps of [2] (see appendix A.2) for the example scenario shown in Fig. 2.1a: the longitudinal and lateral trajectory of the autonomous vehicle are optimized with respect to a longitudinal and lateral driving corridor which are identified using reachability analysis.



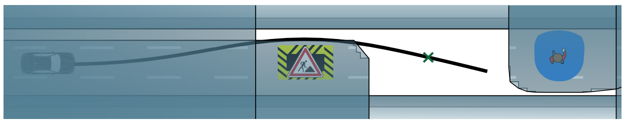
(a) Driving corridors are temporal sequences of connected, vertically convex sets. We depict the driving corridor at a specific time t_k.



(b) The complement of the driving corridor is over-approximated with convex keep-out zones at each time t_k.



(c) Collision avoidance constraints are determined based on the convex keep-out zones, e.g., a convex subset of admissible positions is extracted.



(d) The trajectory of the autonomous vehicle is optimized subject to the collision avoidance constraints of the driving corridor.

Figure 2.3: Main computation steps of [3] (see Appendix A.3) for the example scenario shown in Fig. 2.1a: using reachable sets, our approach computes driving corridors that can be combined with existing optimization-based trajectory planning algorithms that rely on gradient- or Hessian-based solvers to plan trajectories for the autonomous vehicle.

and 2.2c). The overall design of our approach enables the consideration of collision avoidance for the full-dimensional vehicle.

In [3] (see also Appendix A.3), a driving corridor can simultaneously represent different maneuvering possibilities with respect to each obstacle in the environment, such as yielding, evading, or following, but the passing sides for obstacles are uniquely described by a driving corridor (see Fig. 2.1c). This is achieved by defining driving corridors as temporal sequences of connected, vertically convex sets (see Fig. 2.3a). By over-approximating the complement of a driving corridor with a fixed number of polyhedra (see Fig. 2.3b), we are able to exploit smooth reformulations of the (signed) distance function for collision avoidance in nonlinear programs [42] or to extract convex subsets of the set of feasible positions for convex optimization problems (see Fig. 2.3c).

In contrast to [2], our approach introduced in [3] can be combined with many different trajectory planning methods based on nonlinear programming or (successive) convexification. While the driving corridors in [2] are adapted to the decoupled trajectory planner, the driving corridors in [3] are intended to provide general representations of collision avoidance constraints. The generic form of collision avoidance constraints allows the usage of various vehicle models of different fidelity and the combined longitudinal and lateral trajectory planning. While our approach in [3] considers point mass models, our approach in [2] ensures collision avoidance for the full-dimensional vehicle.

2.4 Cooperative Motion Planning

Let us consider a set of N cooperative vehicles V_n, $n \in \mathcal{N} = \{1, 2, \dots, N\}$, whose motions are described by (2.1). We introduce $\square$ as the placeholder for a variable and the subscript $\square_n$ to denote the corresponding variable of the n-th cooperative vehicle. Our goal is to jointly orchestrate the motion of the cooperative vehicles so that collisions with each other and with non-communicating road users $\mathcal{O}_k$ are avoided. Thus, the cooperative motion planning problem is defined as:

Problem 2 (Cooperative Motion Planning) Find an optimal control $u_n^*(\cdot) \in \mathcal{U}_n$ and state trajectory $x_n^*(\cdot) \in \mathcal{X}_n$ for each cooperative vehicle V_n that solve the following non-convex optimization problem:

$$\min_{\hat{u}(\cdot)} \sum_{n=1}^{N} \sum_{k=0}^{h} J_n(x_{k,n}, u_{k,n}) \tag{2.5a}$$

such that

$$\forall n \in \mathcal{N} : x_{0,n} = \tilde{x}_{0,n},\ x_{h,n} \in \mathcal{X}_{\text{goal},n}, \tag{2.5b}$$

$$\forall k \in \{0, \dots, h-1\},\ \forall n \in \mathcal{N} : x_{k+1,n} = f_n(x_{k,n}, u_{k,n}), \tag{2.5c}$$

$$\forall k \in \{0, \dots, h\},\ \forall n, j \in \mathcal{N} : g_n(x_{k,n}, u_{k,n}, k) \leq 0, \tag{2.5d}$$

$$\text{proj}(x_{k,n}) \in \mathcal{D}_{k,n}, \tag{2.5e}$$

$$\text{occ}(x_{k,n}) \cap \text{occ}(x_{k,j}) = \emptyset, n \neq j, \tag{2.5f}$$

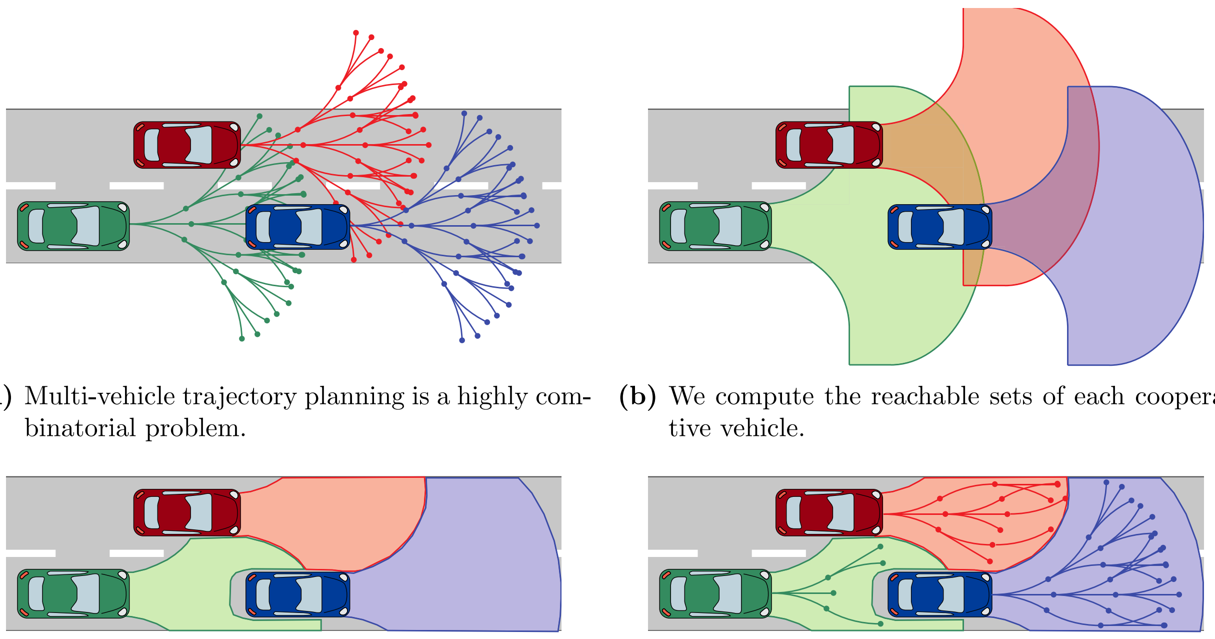
where $\hat{u}(\cdot) = [u_1^T(\cdot), u_2^T(\cdot), \dots, u_N^T(\cdot)]^T$ contains the control inputs for each vehicle over the entire time horizon. As in Problem 1, (2.5c) represents the vehicle dynamics that is subject to constraints (2.5d). Collision avoidance with other road users is represented by (2.5e) and between the cooperative vehicles by (2.5f).

We simplify Problem 2 to obtain a solution in a computationally efficient way. Our approach is inspired by reservation-based conflict resolution schemes, where vehicles reserve areas on the road for exclusive use. To determine the potential areas on the road that can be reserved by each vehicle, we leverage reachability analysis. Since we can incorporate non-communicating road users in the reachable set computation, our approach is applicable to mixed-traffic scenarios. While collision avoidance with non-communicating road users is considered by constraint (2.5e), the reachable sets of two cooperative vehicles V_n and V_j, $n \neq j$, may be in conflict, i.e., two vehicles can reach the same area on the road at the same time (see Fig. 2.4b). This means that a collision between two cooperative vehicles V_n and V_j may occur at a specific time step k:

$$\exists x_{k,n} \in \mathcal{R}_{k,n}, \exists x_{k,j} \in \mathcal{R}_{k,j} :\ \text{occ}(x_{k,n}) \cap \text{occ}(x_{k,j}) \neq \emptyset. \tag{2.6}$$

We demand that the cooperative vehicles negotiate their conflicting reachable set, such that each vehicle receives its own area for trajectory planning (see Fig. 2.4c). The trajectory planning for each cooperative vehicle can then be performed independently using our motion planning approaches introduced in Sec. 2.3 (see Fig. 2.4d). The overall procedure for cooperative motion planning using reachable sets is then:

1. The iterative computation of the reachable sets for each cooperative vehicle V_n and the negotiation of conflicting reachable sets for every time step until the planning horizon is reached;
2. Identification of driving corridors within the negotiated reachable sets for each V_n;
3. Selection of a suitable driving corridor for trajectory planning for each V_n;



(a) Multi-vehicle trajectory planning is a highly combinatorial problem.

(b) We compute the reachable sets of each cooperative vehicle.

(c) Conflicting reachable sets are negotiated such that each vehicle receives its own area on the road.

(d) Vehicles can plan their trajectories independently within their negotiated drivable areas.

Figure 2.4: Cooperative motion planning using reachable sets: instead of planning the trajectories for all vehicles directly, we resolve conflicts by negotiating conflicting reachable sets. As a result, each vehicle receives its own area on the road for motion planning.

4. Extraction of collision avoidance constraints from the selected driving corridors;
5. Optimization-based trajectory planning for each vehicle V_n using the collision avoidance constraints obtained from their driving corridors.

In our first work on cooperative motion planning using reachable sets [5] (see Appendix A.4), we designed the negotiation process with the aim that the drivable space on the road is fairly distributed between the cooperative vehicles. The approach focuses on traffic scenarios with road networks that have negligible curvatures, e.g., highway scenarios. For the negotiation, we group vehicles in coalitions that represent all unique subsets of vehicles with conflicting drivable areas. The method to redistribute conflicting drivable areas within a coalition is inspired by the centroid-based classification (see, e.g., [126]): each cooperative vehicle V_n constitutes its own class. For each V_n in a coalition, we determine the geometric centroid of the conflict-free drivable area. The conflicting reachable set is partitioned into smaller subsets which are assigned to the vehicle with the nearest centroid.

In our second work on cooperative motion planning [4] (see Appendix A.5), we introduce a grid on the road as an abstraction layer and cooperative vehicles determine grid cells that they can reach via reachability analysis. Conflicting grid cells are then negotiated. In contrast to our previous approach in [5], the reachable sets of cooperative vehicles can be computed in their own vehicle-specific curvilinear coordinate system, facilitating the handling of different environments. Furthermore, the abstraction layer allows to formulate the negotiation process as a combinatorial optimization problem—the so-called winner determination problem [127]—

which is also known from auctions. The basic principle is as follows: each conflicting cell on the road is considered as a single assets. These assets can be combined in different packages that are unions of assets. Each cooperative vehicle can now bid on different packages. We aim to find a distribution of packages such that the revenue is maximized and no single asset is assigned more than once. This winner determination problem is generally NP-hard to solve [128] and requires each vehicle to evaluate $2^B - 1$ packages, where B is the number of conflicting cells. Fortunately, computational tractability can be achieved through hierarchical structuring of packages as a tree [128], where packages are decomposed into disjoint subsets at each level of the tree. In this way, finding the optimal allocation of packages has polynomial time complexity in the number of conflicting cells. The complexity of the negotiation process is thus independent of the number of vehicles. Moreover, the maximum number of possible packages to be evaluated reduces to $2B - 1$.

2.5 Ensuring Legal Safety of Autonomous Vehicles

We have developed an online verification framework [1] (see Appendix A.1) to ensure legal safety in urban environments:

Problem 3 (Legal Safety) The autonomous vehicle must not collide with any legal behavior of other traffic participants:

$$\forall t \geq t_0 : \ \mathrm{occ}(x(t)) \cap \mathcal{O}^{\mathrm{legal}}(t) = \emptyset,$$

where $\mathcal{O}^{\mathrm{legal}}(t)$ denotes the occupied positions of other road users for all their legal behaviors including the area outside of the road.

Our safety layer is located between the motion planning layer and control module of the autonomous vehicle, and the safety verification is carried out in successive verification cycles. A new verification cycle is triggered when a new intended trajectory is forwarded from the nominal motion planner to the safety layer. We assume that the forwarded intended trajectories are kinematically feasible but may not be safe, i.e., collision-free and recursively feasible. Given the current environment model and the intended trajectory, our approach performs the following steps to ensure legal safety (see Fig. 2.5): (1) prediction of all legal behaviors of other traffic participants, (2) collision checking of the intended trajectory, and (3) fail-safe trajectory planning. Below, we briefly explain all steps.

Our prediction technique represents a core module for the verification. We use *SPOT* [103] to predict all legal future behaviors of other traffic participants (see Fig. 2.5). In contrast to computing only a countable number of most likely behaviors (see Fig. 2.5), *SPOT* predicts an infinite number of behaviors by using set-based computations. Since the exact computation of the set $\mathcal{O}^{\mathrm{legal}}(t)$ is infeasible, *SPOT* over-approximates $\mathcal{O}^{\mathrm{legal}}(t)$, i.e., the computed set is a superset of $\mathcal{O}^{\mathrm{legal}}(t)$. To obtain a tight superset of $\mathcal{O}^{\mathrm{legal}}(t)$, *SPOT* proceeds in two steps: first, the set of all dynamically feasible behaviors of other road users is over-approximated using reachability analysis. For the reachable set computation, we consider a second-order integrator model, but unlike Def. 3, we do not consider a set of forbidden states. Second, illegal behaviors are removed from the obtained reachable sets. The illegal behaviors are inferred from our legal specification that is based on the Vienna Convention on Road Traffic [121].

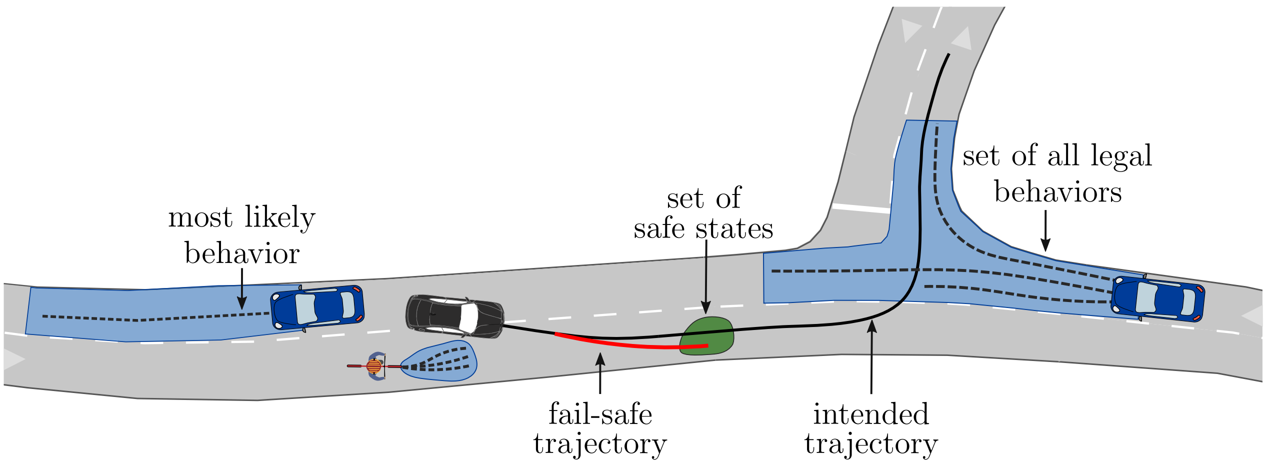


Figure 2.5: The proposed safety layer ensures legal safety at all times. Instead of predicting only a countable number of most likely behaviors of other road users, we predict their set of all legal behaviors. Based on the prediction, we verify whether intended trajectories of the autonomous vehicle are legally safe and provide fail-safe trajectories for safety-critical situations. The fail-safe trajectories lead the autonomous vehicle to a safe set of states in which it can remain legally safe for an infinite time horizon.

The legal specification contains rules, such as speed limits and safe distances. The fewer traffic rules we define in the legal specification, the more cautiously the autonomous vehicle will behave. This is because the prediction remains over-approximative even if no traffic rules are considered in the legal specification, since only illegal behaviors are removed from the set of all dynamically feasible behaviors.

Based on the predicted occupied positions of other road users, we can verify whether intended trajectories are safe and can plan fail-safe trajectories. To verify the safety of the intended trajectory, we check if it is collision-free against the predicted occupancy sets from *SPOT*. Usually, the intended trajectory is planned for longer time horizons, which is necessary for the planning of anticipatory maneuvers, e.g., overtaking a slower driving vehicle. Yet, the predicted occupancy sets by *SPOT* become increasingly larger for longer time horizons, as more space on the road is reachable for other road users. We therefore only consider a short part of the intended trajectory for the safety verification, otherwise the results would be overly conservative. If the considered part of the intended trajectory is verified as safe, i.e., it is collision-free against the predicted set of legal behaviors, we continue to plan a fail-safe trajectory.

The fail-safe trajectory must be collision-free against the predicted occupancy sets by *SPOT* and transition the autonomous vehicle to a set of safe states (see Fig. 2.5). Within the set of safe states, the autonomous vehicle is legally safe for an infinite time horizon, e.g., standstill in dedicated areas on the road. To plan the fail-safe trajectory, we use our motion planning approach that is based on the decoupling of the longitudinal and lateral dynamics of the autonomous vehicle (see Sec. 2.3 and Appendix A.2).

The verification is successful, if the considered part of the intended trajectory could be verified as safe and a feasible fail-safe trajectory could be planned. The considered part of the intended trajectory and the fail-safe trajectory are then concatenated and can be executed by the autonomous vehicle. If the verification was unsuccessful, the autonomous vehicle continues on its previously verified trajectory. By using the principal of induction, it can be proven that the presented approach guarantees legal safety at all times (see Appendix A.1).

Chapter 3

Conclusions

This thesis has investigated novel solutions for the motion planning of autonomous vehicles. Below, we first summarize the work of this thesis and subsequently discuss future research directions.

3.1 Summary

This work proposes a novel approach for individual, cooperative, and safe motion planning. The core methods applied are reachability analysis and continuous optimization, which we have combined to plan trajectories for autonomous vehicles in complex traffic situations. In our experiments, we have successfully demonstrated that our approach generalizes to arbitrary traffic situations, e.g., intersections [1], roundabouts [3], and highways [2]. Our experiments show that our approach is suitable for planning trajectories in cluttered environments as the runtime of our approach decreases with decreasing solution space for trajectory planning (see [2]) and scales favorably with the number of obstacles (see [3]). We also compared our approach with state-of-the-art motion planning methods. The experiments show that (a) in contrast to sampling-based motion planners, our approach efficiently detects narrow passageways in the solution space for trajectory planning without increased computational effort or the necessity to tune parameters (see [2]). (b) For a small number of obstacles, our approach has comparable median computation times with a planner based on mixed-integer quadratic programming and lower computation times for a higher number of obstacles (see [2]). (c) By augmenting a motion planner based on sequential quadratic programming with our approach, feasible solutions can be found in traffic situations that were previously not solvable by the original motion planner (see [3]). This is due to driving corridors eliminating local minima induced by obstacles. Moreover, we have successfully demonstrated in experiments that driving corridors ease the initialization of trajectory planners based on successive convexification so that the number of convex programming iterations can be reduced by an average of approximately 20% (see [3]). We have also successfully validated the drivability of planned trajectories using a BMW 7-series vehicle, where we applied our method online and detected obstacles in the environment using the on-board LiDAR sensors of the test vehicle (see [2]).

Our approach holistically treats individual and cooperative motion planning. The coordination of multiple vehicles is based on the negotiation of their conflicting areas on the road, which are determined using reachability analysis. Since the motions of cooperative vehicles can be planned within their negotiated areas as in the single vehicle case, the transition from cooperative to individual driving is seamless. Although different methods can be implemented for various traffic situations and driving modes, a safe transition between these methods must be ensured. However, there exist an infinite number of different traffic situations, therefore, recognizing the most suitable method and ensuring the existence of an applicable method for the current traffic situations is an error-prone and complex task. In contrast, our approach handles arbitrary traffic situations automatically. We demonstrate the efficacy of our method in multiple traffic scenarios featuring a roundabout, an urban road, a crossing, and a highway scenario (see [4, 5]). Although all these scenarios differ in their road geometry, the number of cooperative vehicles and non-communicating road users, cooperative maneuvers are planned successfully.

We have further embedded our novel motion planning method in a safety layer ensuring that autonomous vehicles adhere to legal safety at all times. Our approach is particularly useful for planning fail-safe trajectories since it (a) generalizes well to different traffic situations, (b) typically becomes faster in critical traffic situations, and (c) can restrict trajectories to end in a set of safe goal states so that recursive feasibility is ensured. The benefits of the proposed safety layer are evaluated on complex urban traffic situations that we have recorded in the cities of Munich and Garching-Hochbrück, Germany. The scenarios include a left-turn maneuver at an intersection, a critical situation with a jaywalking pedestrian, and a lane-change maneuver in dense traffic (see [1]). The experiments show that fail-safe trajectories ensure that the autonomous vehicle respects the right of way of other road users at intersections and keep the autonomous vehicle from stopping within the intersection area. Our results further demonstrate that the conservativeness of our safety layer can be adjusted by its users, e.g., mobility providers, since the set-based prediction $SPOT$ allows one to define legal behaviors differently for specific types of road users. In this way, one can easily consider that inattentive pedestrians cross the road even though it is unlawful, and, in case all traffic rules are disregarded, all dynamically feasible behaviors are predicted. Thus, legal safety is always ensured regardless of the number of considered traffic rules. We have also formally proved that this property holds, i.e., the safety layer is correct-by-construction according to our legal specification. To demonstrate that our safety layer safeguards the autonomous vehicle for arbitrary intended trajectories, we used three different methods to plan intended trajectories for the autonomous vehicle. Even for intended trajectories that ignored other road users, the proposed safety layer successfully guaranteed legal safety.

3.2 Future Work

The initial results of this thesis indicate that our approach efficiently plans drivable motions for individual and cooperative vehicles in arbitrary, complex traffic situations. Moreover, our experiments indicate that legal safety can be strictly ensured without suffering from substantial performance drops. At the same time, our findings suggest opportunities for future research, which we will explain in more detail below.

We use reachability analysis prior to trajectory planning to determine areas to which the search for feasible trajectories can be limited. However, the reachable sets are currently recomputed in each planning cycle, which requires a considerable amount of time. In order to achieve short computation times, we propose to re-use the computed reachable sets of the previous planning cycles in an anytime fashion, i.e., we start with a rough approximation of the reachable set using previous results and refine it as long as time permits. The expected benefits are the availability of solutions after a short start-up phase and improvements in the overall computational efficiency of our approach.

We further recommend that future research should be undertaken to estimate the drivability and suitability of driving corridors. It is desirable to know prior to trajectory planning if a feasible trajectory in a driving corridor exists. Moreover, if there exist multiple driving corridors, the question also arises as to which corridor is best suited to the respective driving task. For this, tactical reasoning and semantic maneuver interpretation are key aspects. Therefore, a possible future direction would be to enrich driving corridors with semantics [129]. Inspired by the concept of hybrid systems that exhibit both discrete and continuous time evolutions, we can extend states in the reachable set computation with discrete state information such as “in lane A” or “in front of vehicle B”. Given the semantic labels on the reachable sets, we are able to extract driving corridors that represent desired driving maneuvers such as first pass vehicle A, then follow vehicle B until reaching the stop sign. Since many conceivable high-level maneuvers are inappropriate in traffic, e.g., repeatedly overtaking the same vehicle by accelerating and braking, some tactical maneuvers could already be excluded during the reachable set computation. The semantic labeling of reachable sets also facilitates the consideration of traffic rules such as the safe distance [130].

Our results on cooperative motion planning using combinatorial optimization are encouraging and should be validated in further experiments. An important matter to resolve for future studies is the development of a bidding strategy for cooperative vehicles. At present, we use an utility function to determine the bids for the cooperative vehicles. All vehicles use the same universal utility function to prevent one vehicle from constantly outbidding others due to different scales and weights that may occur when using different utility functions. Future work should focus on improving the bidding strategy which is usually influenced by a pricing mechanism that determines how much a bidder must pay for a set of items. In future studies, pricing mechanisms could be investigated that compensate cooperative vehicles which hand-over drivable areas.

Our proposed safety layer (see Sec. 2.5) can be extended for fail-safe motion planning with communicating vehicles. Further studies are needed to analyze the communication requirements of the proposed approach in order to agree on a protocol for data exchange between the cooperative vehicles. So far, we have assumed that the communication between cooperative vehicles is stable, but it is of utmost importance to provide fail-safe solutions in case of communication failures.

The prospect of being able to eliminate self-inflicted accidents of autonomous vehicles serves as a continuous incentive to future research. Our results obtained on real data recorded in urban traffic are promising; therefore, the evaluation should be extended to closed-loop test drives in the future.

With regard to fail-safe motion planning, experiments can be conducted in situations of unavoidable accidents caused, for example, by illegal or even hostile behavior of other road users. Our novel motion planning method already allows one to find least-intrusive trajectories as described in Sec. 2.3 by combining driving corridors with the smooth reformulations of the collision avoidance constraints based on the signed distance function [42]. We propose that further research should be undertaken to mitigate the impact of collisions to reduce the risk of serious injury in situations of inevitable accidents.

Bibliography

[11] J. N. Bajpai, "Emerging vehicle technologies & the search for urban mobility solutions," *Urban, Planning and Transport Research*, vol. 4, no. 1, pp. 83–100, 2016.

[12] C. Burger, P. F. Orzechowski, Ö. Ş. Taş, and C. Stiller, "Rating cooperative driving: A scheme for behavior assessment," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2017, pp. 1–6.

[13] S. W. Loke, "Cooperative automated vehicles: A review of opportunities and challenges in socially intelligent vehicles beyond networking," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 4, pp. 509–518, 2019.

[14] J. Axelsson, "Safety in vehicle platooning: A systematic literature review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1033–1045, 2017.

[15] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

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

[17] B. Mirchevska, C. Pek, M. Werling, M. Althoff, and J. Boedecker, "High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2018, pp. 2156–2162.

[18] D. A. Pomerleau, "ALVINN: An autonomous land vehicle in a neural network," in *Advances in Neural Information Processing Systems 1.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 305–313.

[19] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, pp. 1–9, 2016.

[20] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 187–210, 2018.

[21] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 416–442, 2015.

[22] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser, "A review of motion planning for highway autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 5, pp. 1826–1848, 2020.

[23] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," in *Proc. of the International Conference on Neural Information Processing Systems.* Cambridge, MA, USA: MIT Press, 2003, pp. 767–774.

[24] E. Frazzoli, M. A. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1077–1091, 2005.

[25] D. Heß, M. Althoff, and T. Sattel, "Formal verification of maneuver automata for parameterized motion primitives," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 1474–1481.

[26] D. J. Grymin, C. B. Neas, and M. Farhood, "A hierarchical approach for primitive-based motion planning and control of autonomous vehicles," *Robotics and Autonomous Systems*, vol. 62, no. 2, pp. 214–228, 2014.

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

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

[29] M. Pivtoraiko and A. Kelly, "Kinodynamic motion planning with state lattice motion primitives," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 2172–2179.

[30] M. Werling, S. Kammel, J. Ziegler, and L. Gröll, "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *International Journal of Robotics Research*, vol. 31, no. 3, pp. 346–359, 2012.

[31] F. von Hundelshausen, M. Himmelsbach, F. Hecker, A. Mueller, and H.-J. Wuensche, "Driving with tentacles: Integral structures for sensing and motion," *Journal of Field Robotics*, vol. 25, no. 9, pp. 640–673, 2008.

[32] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[33] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," in *Proc. of the American Control Conference*, 2001, pp. 43–49.

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

[35] I. Noreen, A. Khan, and Z. Habib, "Optimal path planning using RRT* based approaches: A survey and future directions," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 11, 2016.

[36] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.

[37] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2011, pp. 1478–1483.

[38] X. Qian, F. Altché, P. Bender, C. Stiller, and A. de La Fortelle, "Optimal trajectory planning for autonomous driving integrating logical constraints: An MIQP perspective," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2016, pp. 205–210.

[39] C. Miller, C. Pek, and M. Althoff, "Efficient mixed-integer programming for longitudinal and lateral motion planning of autonomous vehicles," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 1954–1961.

[40] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *Proc. of the IEEE European Control Conference*, 2001, pp. 2603–2608.

[41] H. P. Williams, *Model building in mathematical programming.* John Wiley & Sons, 2013.

[42] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, 2021.

[43] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[44] S. P. Boyd and L. Vandenberghe, *Convex optimization.* Cambridge University Press, 2004.

[45] C. Pek and M. Althoff, "Computationally efficient fail-safe trajectory planning for self-driving vehicles using convex optimization," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2018, pp. 1447–1454.

[46] Y. Mao, D. Dueri, M. Szmuk, and B. Açıkmeşe, "Successive convexification of non-convex optimal control problems with state constraints," in *IFAC-PapersOnLine*, 2017, vol. 50, no. 1, pp. 4063–4069.

[47] C. Zagaris, H. Park, J. Virgili-Llop, R. Zappulla, M. Romano, and I. Kolmanovsky, "Model predictive control of spacecraft relative motion with convexified keep-out-zone constraints," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 9, pp. 2054–2062, 2018.

[48] R. Deits and R. Tedrake, “Computing large convex regions of obstacle-free space through semidefinite programming,” in *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*, H. L. Akin, N. M. Amato, V. Isler, and A. F. van der Stappen, Eds. Cham: Springer International Publishing, 2015, pp. 109–124.

[49] S. Bhattacharya and R. Ghrist, “Path homotopy invariants and their application to optimal trajectory planning,” *Annals of Mathematics and Artificial Intelligence*, vol. 84, pp. 139–160, 2018.

[50] F. Altché and A. de La Fortelle, “Partitioning of the free space-time for on-road navigation of autonomous ground vehicles,” in *Proc. of the IEEE Conference on Decision and Control*, 2017, pp. 2126–2133.

[51] M. Schmidt, C. Wissing, J. Braun, T. Nattermann, and T. Bertram, “Maneuver identification for interaction-aware highway lane change behavior planning based on polygon clipping and convex optimization,” in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2019, pp. 3948–3953.

[52] P. Bender, Ö. Ş. Taş, J. Ziegler, and C. Stiller, “The combinatorial aspect of motion planning: Maneuver variants in structured environments.” in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2015, pp. 1386–1392.

[53] J. Park, S. Karumanchi, and K. Iagnemma, “Homotopy-based divide-and-conquer strategy for optimal trajectory planning via mixed-integer programming,” *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1101–1115, 2015.

[54] L. Campos-Macías, D. Gómez-Gutiérrez, R. Aldana-López, R. de la Guardia, and J. I. Parra-Vilchis, “A hybrid method for online trajectory planning of mobile robots in cluttered environments,” *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 935–942, 2017.

[55] W. Lim, S. Lee, M. Sunwoo, and K. Jo, “Hierarchical trajectory planning of an autonomous car based on the integration of a sampling and an optimization method,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 2, pp. 613–626, 2018.

[56] Z. Zhu, E. Schmerling, and M. Pavone, “A convex optimization approach to smooth trajectories for motion planning with car-like robots,” in *Proc. of the IEEE Conference on Decision and Control*, 2015, pp. 835–842.

[57] B. Li and Y. Zhang, “Fast trajectory planning for off-road autonomous driving with a spatiotemporal tunnel and numerical optimal control approach,” in *Proc. of the IEEE International Conference on Advanced Robotics and Mechatronics*, 2019, pp. 924–929.

[58] Q. H. Do, S. Mita, and K. Yoneda, “Narrow passage path planning using fast marching method and support vector machine,” in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2014, pp. 630–635.

[59] M. Morsali, J. Åslund, and E. Frisk, “Trajectory planning in traffic scenarios using support vector machines,” *IFAC-PapersOnLine*, vol. 52, no. 5, pp. 91–96, 2019.

[60] A. Riener and A. Ferscha, “Enhancing future mass ICT with social capabilities,” in *Co-evolution of Intelligent Socio-technical Systems: Modelling and Applications in Large Scale Emergency and Transport Domains*, E. Mitleton-Kelly, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 141–184.

[61] S. Ulbrich, S. Grossjohann, C. Appelt, K. Homeier, J. Rieken, and M. Maurer, “Structuring cooperative behavior planning implementations for automated driving,” in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2015, pp. 2159–2165.

[62] M. Düring and P. Pascheka, “Cooperative decentralized decision making for conflict resolution among autonomous agents,” in *Proc. of the IEEE International Symposium on Innovations in Intelligent Systems and Applications*, 2014, pp. 154–161.

[63] M. Zimmermann and K. Bengler, “A multimodal interaction concept for cooperative driving,” in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2013, pp. 1285–1290.

[64] M. Düring and K. Lemmer, “Cooperative maneuver planning for cooperative driving,” *IEEE Intelligent Transportation Systems Magazine*, vol. 8, no. 3, pp. 8–22, 2016.

[65] D. Lenz, T. Kessler, and A. Knoll, “Tactical cooperative planning for autonomous highway driving using Monte-Carlo Tree Search,” in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2016, pp. 447–453.

[66] K. Kurzer, F. Engelhorn, and J. M. Zöllner, “Decentralized cooperative planning for automated vehicles with continuous Monte Carlo Tree Search,” in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2018, pp. 452–459.

[67] M. Naumann, M. Lauer, and C. Stiller, “Generating comfortable, safe and comprehensible trajectories for automated vehicles in mixed traffic,” in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2018, pp. 575–582.

[68] C. Hubmann, J. Schulz, G. Xu, D. Althoff, and C. Stiller, “A belief state planner for interactive merge maneuvers in congested traffic,” in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2018, pp. 1617–1624.

[69] M. Khayatian, M. Mehrabian, E. Andert, R. Dedinsky, S. Choudhary, Y. Lou, and A. Shirvastava, “A survey on intersection management of connected autonomous vehicles,” *ACM Transactions on Cyber-Physical Systems*, vol. 4, no. 4, 2020, article no.: 48.

[70] L. Chen and C. Englund, “Cooperative intersection management: A survey,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 570–586, 2016.

[71] J. Rios-Torres and A. A. Malikopoulos, “A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1066–1077, 2017.

[72] F. Fakhfakh, M. Tounsi, and M. Mosbah, “Vehicle platooning systems: Review, classification and validation strategies,” *International Journal of Networked and Distributed Computing*, vol. 8, no. 4, pp. 203–213, 2020.

[73] E. Kulla, N. Jiang, E. Spaho, and N. Nishihara, “A survey on platooning techniques in VANETs,” in *Complex, Intelligent, and Software Intensive Systems*, L. Barolli, N. Javaid, M. Ikeda, and M. Takizawa, Eds. Cham: Springer International Publishing, 2019, pp. 650–659.

[74] A. Soni and H. Hu, “Formation control for a fleet of autonomous ground vehicles: A survey,” *Robotics*, vol. 7, no. 4, 2018, article no.: 67.

[75] Y. Liu and R. Bucknall, “A survey of formation control and motion planning of multiple unmanned vehicles,” *Robotica*, vol. 36, no. 7, pp. 1019–1047, 2018.

[76] T. Balch and R. C. Arkin, “Behavior-based formation control for multirobot teams,” *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 926–939, 1998.

[77] L. Gao, D. Chu, Y. Cao, L. Lu, and C. Wu, “Multi-lane convoy control for autonomous vehicles based on distributed graph and potential field,” in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2019, pp. 2463–2469.

[78] Z. Huang, D. Chu, C. Wu, and Y. He, “Path planning and cooperative control for automated vehicle platoon using hybrid automata,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 3, pp. 959–974, 2019.

[79] S. W. Smith, Y. Kim, J. Guanetti, A. A. Kurzhanskiy, M. Arcak, and F. Borrelli, “Balancing safety and traffic throughput in cooperative vehicle platooning,” in *Proc. of the IEEE European Control Conference*, 2019, pp. 2197–2202.

[80] X. Qian, A. de La Fortelle, and F. Moutarde, “A hierarchical model predictive control framework for on-road formation control of autonomous vehicles,” in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2016, pp. 376–381.

[81] J. Eilbrecht and O. Stursberg, “Optimization-based maneuver automata for cooperative trajectory planning of autonomous vehicles,” in *Proc. of the IEEE European Control Conference*, 2018, pp. 82–88.

[82] C. Burger and M. Lauer, “Cooperative multiple vehicle trajectory planning using MIQP,” in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2018, pp. 602–607.

[83] C. Frese and J. Beyerer, “A comparison of motion planning algorithms for cooperative collision avoidance of multiple cognitive automobiles,” in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2011, pp. 1156–1162.

[84] C. Frese and J. Beyerer, “Planning cooperative motions of cognitive automobiles using tree search algorithms,” in *KI 2010: Advances in Artificial Intelligence*, R. Dillmann, J. Beyerer, U. D. Hanebeck, and T. Schultz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 91–98.

[85] M. Cirillo, T. Uras, and S. Koenig, "A lattice-based approach to multi-robot motion planning for non-holonomic vehicles," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 232–239.

[86] R. Hult, G. R. Campos, E. Steinmetz, L. Hammarstrand, P. Falcone, and H. Wymeersch, "Coordination of cooperative autonomous vehicles: Toward safer and more efficient road transportation," *IEEE Signal Processing Magazine*, vol. 33, no. 6, pp. 74–84, 2016.

[87] D. Heß, R. Lattarulo, J. Pérez, T. Hesse, and F. Köster, "Negotiation of cooperative maneuvers for automated vehicles: Experimental results," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2019, pp. 1545–1551.

[88] K. Dresner and P. Stone, "Multiagent traffic management: a reservation-based intersection control mechanism," in *Proc. of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, 2004, pp. 530–537.

[89] O. Mehani and A. de La Fortelle, "Trajectory planning in a crossroads for a fleet of driverless vehicles," in *Computer Aided Systems Theory – EUROCAST 2007*, R. Moreno Díaz, F. Pichler, and A. Quesada Arencibia, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1159–1166.

[90] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proc. of the IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006.

[91] M. Vasirani and S. Ossowski, "A market-inspired approach for intersection management in urban road traffic networks," *Journal of Artificial Intelligence Research*, vol. 43, no. 1, pp. 621–659, 2012.

[92] H. Schepperle and K. Böhm, "Auction-based traffic management: Towards effective concurrent utilization of road intersections," in *10th IEEE Conference on E-Commerce Technology and the 5th IEEE Conference on Enterprise Computing, E-Commerce and E-Services*, 2008, pp. 105–112.

[93] D. Carlino, S. D. Boyles, and P. Stone, "Auction-based autonomous intersection management," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2013, pp. 529–534.

[94] T. Fraichard, "A short paper about motion safety," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2007, pp. 1140–1145.

[95] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016.

[96] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH Journal*, vol. 1, no. 1, pp. 1–14, 2014.

[97] J. Schulz, C. Hubmann, J. Löchner, and D. Burschka, "Interaction-aware probabilistic behavior prediction in urban environments," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 3999–4006.

[98] T. Gindele, S. Brechtel, and R. Dillmann, “Learning driver behavior models from traffic observations for decision making and planning,” *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 69–79, 2015.

[99] C. Tang, J. Chen, and M. Tomizuka, “Adaptive probabilistic vehicle trajectory prediction through physically feasible bayesian recurrent neural network,” in *Proc. of the IEEE International Conference on Robotics and Automation*, 2019, pp. 3846–3852.

[100] E. A. I. Pool, J. F. P. Kooij, and D. M. Gavrila, “Context-based cyclist path prediction using recurrent neural networks,” in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2019, pp. 824–830.

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

[102] S. Bouraine, T. Fraichard, and H. Salhi, “Provably safe navigation for mobile robots with limited field-of-views in unknown dynamic environments,” in *Proc. of the IEEE International Conference on Robotics and Automation*, 2012, pp. 174–179.

[103] M. Koschi and M. Althoff, “Set-based prediction of traffic participants considering occlusions and traffic rules,” *IEEE Transactions on Intelligent Vehicles*, 2020, [early access].

[104] T. Fraichard and H. Asama, “Inevitable collision states. A step towards safer robots?” in *Proc. of the IEEE/RSJ Conference on Intelligent Robots and Systems*, 2003, pp. 388–393.

[105] D. Althoff, M. Werling, N. Kaempchen, D. Wollherr, and M. Buss, “Lane-based safety assessment of road scenes using Inevitable Collision States,” in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2012, pp. 31–36.

[106] M. Jalalmaab, B. Fidan, S. Jeon, and P. Falcone, “Guaranteeing persistent feasibility of model predictive motion planning for autonomous vehicles,” in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 843–848.

[107] K. Berntorp, A. Weiss, C. Danielson, I. V. Kolmanovsky, and S. Di Cairano, “Automated driving: Safe motion planning using positively invariant sets,” in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2017, pp. 1–6.

[108] C. Pek and M. Althoff, “Efficient computation of invariably safe states for motion planning of self-driving vehicles,” in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 3523–3530.

[109] C. Baier and J.-P. Katoen, *Principles of model checking.* MIT press, 2008.

[110] M. Kamali, L. A. Dennis, O. McAree, M. Fisher, and S. M. Veres, “Formal verification of autonomous vehicle platooning,” *Science of Computer Programming*, vol. 148, pp. 88–106, 2017.

[111] M. Völker, M. Kloock, L. Rabanus, B. Alrifaee, and S. Kowalewski, “Verification of cooperative vehicle behavior using temporal logic,” *IFAC-PapersOnLine*, vol. 52, no. 8, pp. 99–104, 2019.

[112] A. Rizaldi, F. Immler, B. Schürmann, and M. Althoff, “A formally verified motion planner for autonomous vehicles,” in *Automated Technology for Verification and Analysis*, S. K. Lahiri and C. Wang, Eds. Cham: Springer International Publishing, 2018, pp. 75–90.

[113] M. Hilscher, S. Linker, and E.-R. Olderog, “Proving safety of traffic manoeuvres on country roads,” in *Theories of Programming and Formal Methods*, Z. Liu, J. Woodcock, and H. Zhu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 196–212.

[114] J. Tumova, G. C. Hall, S. Karaman, E. Frazzoli, and D. Rus, “Least-violating control strategy synthesis with safety rules,” in *Proc. of the International Conference on Hybrid Systems: Computation and Control*, 2013, pp. 1–10.

[115] T. Wongpiromsarn, S. Karaman, and E. Frazzoli, “Synthesis of provably correct controllers for autonomous vehicles in urban environments,” in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2011, pp. 1168–1173.

[116] B. Schürmann, D. Heß, J. Eilbrecht, O. Stursberg, F. Köster, and M. Althoff, “Ensuring drivability of planned motions using formal methods,” in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2017, pp. 1–8.

[117] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, “FaSTrack: A modular framework for fast and guaranteed safe motion planning,” in *Proc. of the IEEE Conference on Decision and Control*, 2017, pp. 1517–1522.

[118] S. Vaskov, S. Kousik, H. Larson, F. Bu, J. Ward, S. Worrall, M. Johnson-Roberson, and R. Vasudevan, “Towards provably not-at-fault control of autonomous robots in arbitrary dynamic environments,” in *Proc. of Robotics: Science and Systems*, 2019, pp. 1–9.

[119] S. Shalev-Shwartz, S. Shammah, and A. Shashua, “On a formal model of safe and scalable self-driving cars,” *CoRR*, vol. abs/1708.06374v6, pp. 1–37, 2018.

[120] B. Vanholme, D. Gruyer, B. Lusetti, S. Glaser, and S. Mammar, “Highly automated driving on highways based on legal safety,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 333–347, 2013.

[121] Economic Comission for Europe: Inland Transport Committee. (1968) Convention on road traffic. [Online]. Available: http://unece.org/fileadmin/DAM/trans/conventn/crt1968e.pdf

[122] S. Magdici and M. Althoff, “Fail-safe motion planning of autonomous vehicles,” in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2016, pp. 452–458.

[123] M. Althoff, “Reachability analysis and its application to the safety assessment of autonomous cars,” Dissertation, Technische Universität München, München, 2010.

[124] S. Söntges and M. Althoff, “Computing the drivable area of autonomous road vehicles in dynamic road scenes,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 6, pp. 1855–1866, 2018.

[125] B. Gutjahr, L. Gröll, and M. Werling, “Lateral vehicle trajectory optimization using constrained linear time-varying MPC,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1586–1595, 2017.

[126] E.-H. S. Han and G. Karypis, “Centroid-based document classification: Analysis and experimental results,” in *Principles of Data Mining and Knowledge Discovery*, D. A. Zighed, J. Komorowski, and J. Żytkow, Eds. Springer Berlin Heidelberg, 2000, pp. 424–431.

[127] P. Cramton, Y. Shoham, and R. Steinberg, Eds., *Combinatorial Auctions*. MIT Press, 2006.

[128] M. H. Rothkopf, A. Pekeč, and R. Harstad, “Computationally manageable combinational auctions,” *Management Science*, vol. 44, no. 8, pp. 1131–1147, 1998.

Appendix A

Reproduction of Publications

A.1 Using Online Verification to Prevent Autonomous Vehicles from Causing Accidents [1]

Summary This work proposes a novel online verification technique that ensures legal safety of autonomous vehicles in urban traffic situations (see Problem 3), i.e., autonomous vehicles never cause self-inflicted accidents, although other road users may perform any behavior in accordance with traffic rules. We therefore introduce a safety layer for existing motion planners, e.g., those based on machine learning, which provide intended trajectories that are kinematically feasible but may not be legally safe. Our safety layer offers three main features: (1) online situation assessment, i.e., every situation is considered on-the-fly and all possible future evolutions according to legal safety are taken into account. (2) Fail-safe operation, i.e., it is ensured that the autonomous vehicle always has a fallback solution available in hazardous situations. (3) Correct by construction, i.e., independent of the existing motion planner and the number of specified traffic rules, legal safety is guaranteed.

Based on reachability analysis and specified traffic rules, our technique first computes the set of all future behaviors of other road users that comply with legal safety. Then, we compute the drivable area of the autonomous vehicle, also using reachability analysis, from which we determine driving corridors for the subsequent fail-safe motion planning.

We demonstrate the benefits of our approach in different urban traffic scenarios that we recorded in real traffic. The selected traffic situations are among the situations in which most accidents occur in urban traffic, i.e., a left-turn maneuver at an intersection, a critical situation with a jaywalking pedestrian, and a lane-change maneuver in dense traffic. Furthermore, we test our safety layer in combination with different motion planners that provide intended trajectories. Our results indicate that legal safety can be guaranteed and at the same time the driving behavior is not conservative.

Author Contributions **S. M.**, C. P., and M. K. developed the verification technique during replanning. M. K. developed the concept and algorithms for the set-based prediction. **S. M.** and C. P. developed the concept and algorithms for the drivable area computation, driving corridor identification, and fail-safe trajectory planning. M. A. developed the main concept of online verification by integrating set-based prediction and fail-safe trajectory generation. M. A. developed the underlying algorithms for reachability analysis and led the research project. **S. M.**, C. P., and M. K. designed, conducted, and evaluated the experiments and collected the data. **S. M.**, C. P., and M. K. wrote the article and the supplementary information.

Attachments The Supplementary Information, the Supplementary Data File, and Supplementary Videos are available at `https://www.nature.com/articles/s42256-020-0225-y#Sec15`.

ARTICLES
https://doi.org/10.1038/s42256-020-0225-y

nature machine intelligence

Check for updates

Using online verification to prevent autonomous vehicles from causing accidents

Christian Pek[1,2] ✉, Stefanie Manzinger[1,2] ✉, Markus Koschi[1,2] ✉ and Matthias Althoff[1]

Ensuring that autonomous vehicles do not cause accidents remains a challenge. We present a formal verification technique for guaranteeing legal safety in arbitrary urban traffic situations. Legal safety means that autonomous vehicles never cause accidents although other traffic participants are allowed to perform any behaviour in accordance with traffic rules. Our technique serves as a safety layer for existing motion planning frameworks that provide intended trajectories for autonomous vehicles. We verify whether intended trajectories comply with legal safety and provide fallback solutions in safety-critical situations. The benefits of our verification technique are demonstrated in critical urban scenarios, which have been recorded in real traffic. The autonomous vehicle executed only safe trajectories, even when using an intended trajectory planner that was not aware of other traffic participants. Our results indicate that our online verification technique can drastically reduce the number of traffic accidents.

Safety remains a major challenge in the realization of autonomous vehicles. Unsafe decisions by autonomous vehicles can endanger human lives and cause tremendous economic loss in terms of product liability. Although autonomous driving is becoming a reality, recent accidents involving autonomous driving systems have raised major concerns in various institutions[1], and policy makers continue to debate about adequate safety levels for certifying autonomous vehicles[2]. To achieve widespread acceptance, safety concerns must be resolved to the full satisfaction of all road users. So far, automotive safety relies primarily on simulation and testing. However, due to the infinitely many unique real-world scenarios, these techniques cannot ensure strict safety levels[3,4], especially when using machine learning for motion planning[5].

We call for a paradigm shift from accepting residual collision risks to ensuring safety through formal verification. Formal verification describes the process of proving that a system always fulfils a desired formal specification[6]. However, in the context of safe motion planning, specifying all unsafe scenarios and proper reactions of autonomous vehicles is a tedious task[6]. Although it cannot be excluded that autonomous vehicles are involved in accidents, such as when a following car deliberately provokes a rear-end collision, self-inflicted accidents can and should be eliminated. What can we expect from human drivers to avoid self-inflicted accidents? Based on the Vienna Convention on Road Traffic, which serves as a foundation for safe driving in 78 countries, human drivers 'shall avoid any behaviour likely to endanger or obstruct traffic' (article 7 of ref. [7]). Inspired by this general rule, we demand that motions of autonomous vehicles must be collision-free under the premise that other traffic participants are allowed to perform all legal behaviours, that is, all dynamically feasible behaviours that do not violate traffic rules. Following refs. [8,9], we refer to this specification as 'legal safety'.

In contrast to related work, our holistic approach computes all legal behaviours of other traffic participants and collision-free fallback plans for the autonomous vehicle. Our solution serves as a safety layer for existing motion planning frameworks. These frameworks generate intended trajectories but cannot guarantee legal safety. However, in combination with our verification technique, legal safety is ensured. Our technique provides the following three key features:

1. Online situation assessment: The safety of each traffic situation is assessed online during operation of the autonomous vehicle by rigorously predicting all legal future evolutions of the scenario (blue areas, Fig. 1) while accounting for measurement uncertainties. In contrast to classical testing approaches, even previously unseen traffic environments can be handled, that is, scenarios with arbitrary road geometries and number of traffic participants.
2. Fail-safe operation: Our approach ensures that the autonomous vehicle always has a fail-safe trajectory to a standstill in designated safe areas, which serves as a fallback plan in the case where a safety-critical situation occurs (see the fail-safe trajectory in Fig. 1).
3. Correct by construction: Regardless of the provided motion planning framework, which may include machine learning components, our verification technique ensures that the autonomous vehicle operates in compliance with legal safety at all times. Furthermore, our safety guarantees hold even if certain traffic rules are not yet included in our technique, because, from the set of all dynamically feasible behaviours, we only remove the behaviours that are illegal according to the considered traffic rules.

At present, verification is performed during the design process—that is, offline, before systems are deployed[10]. However, offline verification is not suitable for autonomous vehicles, as these vehicles operate in highly uncertain environments in which each scenario is unique. For this reason, online verification approaches have been introduced that verify safety properties during operation of the autonomous vehicles (section II-C of ref. [11]), for example, through logical reasoning[12,13] or avoiding inevitable collision states[14,15]. In the case where a trajectory is classified as unsafe, these approaches usually do not provide an alternative safe plan for the vehicle. In the field of control, popular safety techniques are robust model predictive control approaches[16–18] and correct-by-construction controllers,

[1]Cyber-Physical Systems Group, Department of Informatics, Technical University of Munich, Garching, Germany. [2]These authors contributed equally. Christian Pek, Stefanie Manzinger, Markus Koschi. ✉e-mail: christian.pek@tum.de; stefanie.manzinger@tum.de; markus.koschi@tum.de

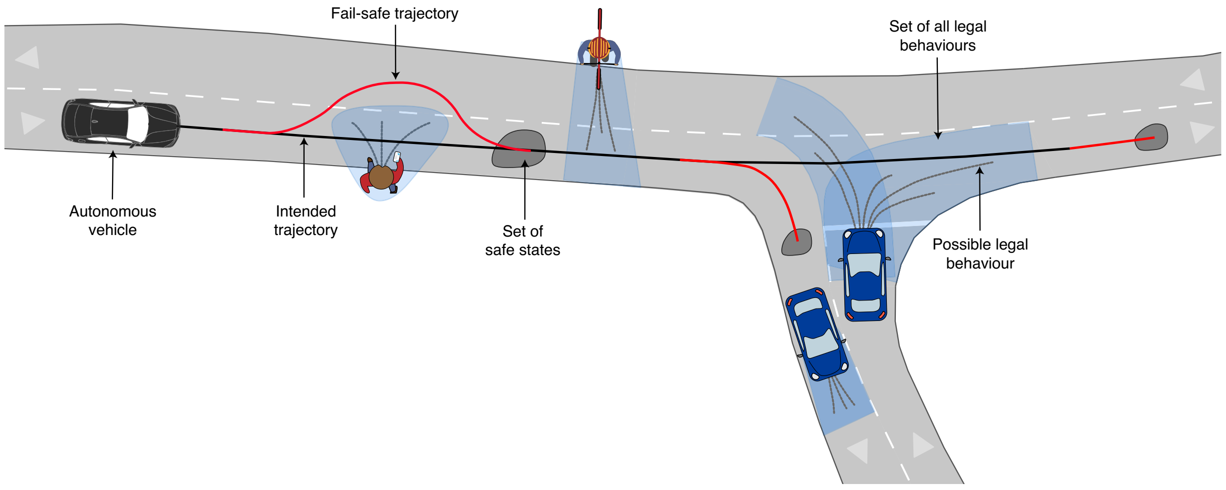


Fig. 1 | Verification of legal safety. Intended trajectories (black line) are usually planned by only considering the most likely behaviours (grey lines) of other traffic participants. Our online verification technique ensures that the autonomous vehicle is safe in accordance with legal safety by maintaining fail-safe trajectories (red lines) at all times. These fail-safe trajectories are collision-free against the set of all legal behaviours (blue areas) of other traffic participants and safeguard the autonomous vehicle along its intended trajectory to safe states (grey areas).

for example, involving barrier certificates[19], Lyapunov functions[20] or automatic controller synthesis[21]. These approaches ensure that the vehicle avoids unsafe states or is kept within an invariant set of safe states[22,23] at all times. Closely related recent approaches incorporate reachability analysis to compute the set of states that a system is able to reach over time. Thus, it can be verified that unsafe states are not reached during operation[9,24–26]. However, these existing approaches are often computationally intractable, do not generalize to arbitrary traffic scenarios or do not provide the required prediction of unsafe sets in dynamic environments.

In the context of autonomous driving, the time-variant unsafe sets are commonly defined as the future occupied positions of other traffic participants, which can be obtained by motion prediction[27]. Existing prediction approaches usually compute a countable set of most likely behaviours by applying probabilistic[28–30] or machine learning methods[31–33]. The safety of autonomous vehicles is guaranteed only if no traffic participant deviates from the few predicted behaviours, but such deviations often occur in real traffic. By incorporating reachability analysis, predictions are able to consider an infinite number of possible future behaviours of dynamic obstacles[9,34–37]. Yet, allowing for all dynamically feasible behaviours of other traffic participants overly limits the manoeuvrability of the autonomous vehicle. Therefore, our reachability-based prediction only considers behaviours that are dynamically feasible in the road network and that do not violate a set of formalized traffic rules (blue areas, Fig. 1).

The motion planner for fail-safe trajectories must cope with small and convoluted solution spaces. Commonly used trajectory planning techniques either discretize the input or state space of the autonomous vehicle[38,39] or apply variational techniques in continuous space[40–42]. The former methods suffer from discretization effects, such that narrow passageways in the solution space may not be found[43] or safe terminal states may not be reached[44]. Although variational-based methods overcome these limitations, the non-convexity of the motion planning problem due to nonlinear vehicle dynamics and collision avoidance poses a major challenge. As a result, variational-based techniques are often computationally complex[45–47] or must be guided through the solution space to work in dense traffic situations[48,49], for example, by specifying driving corridors that represent temporal tactical decisions, such as overtaking an obstacle on the left or right. Approaches for obtaining driving corridors generally do not consider the dynamics of the autonomous vehicle[50–52] and thus may not be able to reason about the drivability of driving corridors. Our approach combines reachability analysis with convex optimization to determine drivable fail-safe trajectories within dynamics-aware driving corridors in arbitrary traffic scenarios (fail-safe trajectories, Fig. 1).

Results

Our verification technique ensures legal safety over consecutive verification cycles. A new verification cycle $c \in \mathbb{N}_+$ begins whenever an intended trajectory I_c is provided by the intended trajectory planner of the existing motion planning framework, where c is incremented by one for each received intended trajectory. The autonomous vehicle can only start executing a new intended trajectory I_c that is starting at t_c if I_c is successfully verified as legally safe. A trajectory is legally safe if it (1) is collision-free against the predicted occupancy sets (that is, occupied positions) that result from all legal behaviours of other traffic participants and (2) leads the autonomous vehicle to a safe terminal state.

Typically, the time horizon T_{I_c} of I_c is several seconds for planning anticipatory motions. However, the predicted occupancy sets of the surrounding traffic participants become increasingly large for longer time horizons due to growing uncertainties regarding their future behaviours. Thus, I_c is often not safe over its entire time horizon T_{I_c}. For the safety verification (Fig. 2a), we therefore do not consider the entire intended trajectory I_c, but only a short part of I_c lasting from t_c until $t_c + \Delta_c^{\mathrm{safe}}$, where $\Delta_c^{\mathrm{safe}} \in \mathbb{R}_+$. We regard this part of I_c as legally safe and refer to it as I_c^{safe} if it is collision-free against the predicted occupancy sets within its entire time duration Δ_c^{safe}. Because I_c^{safe} does not ensure that the autonomous vehicle remains legally safe for $t > t_c + \Delta_c^{\mathrm{safe}}$, we compute a consecutive fail-safe trajectory F_c (the index of F indicates the corresponding intended trajectory I). The fail-safe trajectory F_c needs to smoothly continue I_c^{safe}, be collision-free against the predicted occupancy sets for its entire time horizon T_{F_c}, and transition the autonomous vehicle to a standstill in safe areas. We say that I_c is verified successfully if I_c^{safe} and F_c exist and are computed prior to t_c. The concatenation of I_c^{safe} and F_c represents the verified trajectory and is denoted as $I_c^{\mathrm{safe}} \parallel F_c$.

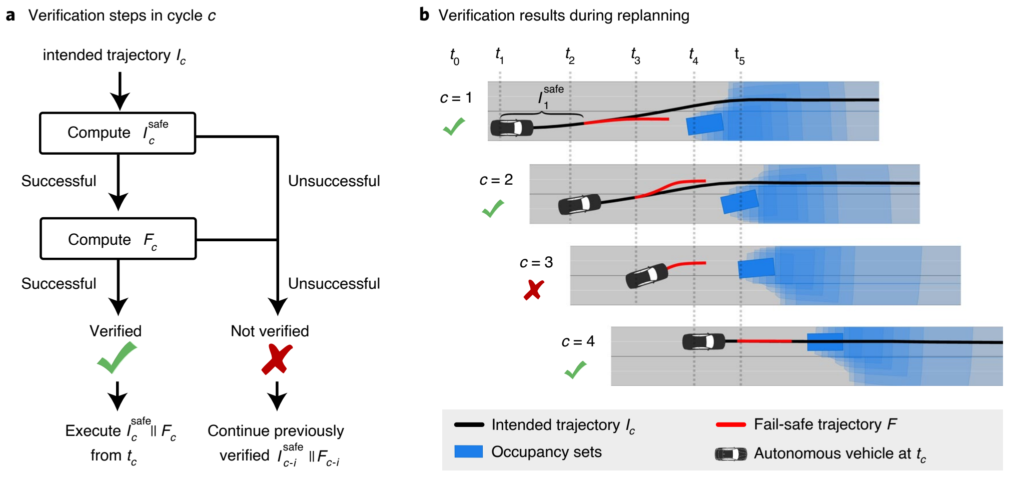

a Verification steps in cycle c
intended trajectory I_c
Compute I_c^{safe}
Successful
Unsuccessful
Compute F_c
Successful
Unsuccessful
Verified
Not verified
Execute $I_c^{safe} \| F_c$ from t_c
Continue previously verified $I_{c-i}^{safe} \| F_{c-i}$
b Verification results during replanning
t_0 t_1 t_2 t_3 t_4 t_5
$c = 1$
I_1^{safe}
$c = 2$
$c = 3$
$c = 4$
Intended trajectory I_c
Fail-safe trajectory F
Occupancy sets
Autonomous vehicle at t_c

Fig. 2 | Verification during replanning. a, In each verification cycle c, the given intended trajectory I_c is verified by computing the safe part I_c^{safe} and the fail-safe trajectory F_c. **b**, If the verification result of cycle c is successful (as in $c \in \{1, 2, 4\}$), the verified trajectory $I_c^{safe} \| F_c$ is executed starting at t_c. If the verification result is unsuccessful (as in $c = 3$), the verified trajectory I_{c-i}^{safe} and F_{c-i} of a previous successful verification cycle $c - i$ is executed until a new intended trajectory is successfully verified again (as in $c = 4$).

Let us explain the verification procedure during replanning using Fig. 2. Initially, at t_0, we assume that the autonomous vehicle is in a safe state (for example, parked). Immediately after the autonomous vehicle successfully verifies a given intended trajectory I_1 in verification cycle $c = 1$ (that is, I_1^{safe} and F_1 are obtained), the vehicle is allowed to engage in the autonomous driving mode at time t_1 and starts executing I_1^{safe} of the verified trajectory $I_1^{safe} \| F_1$ (see the result of $c = 1$ in Fig. 2b). The intended trajectory planner can then provide new intended trajectories I_c, $c > 1$, for verification. If a new trajectory I_c is successfully verified, the autonomous vehicle can transition from the previously verified trajectory to I_c^{safe} of the new verified trajectory $I_c^{safe} \| F_c$ at time t_c (see Fig. 2a and the result of $c \in \{2, 4\}$ in Fig. 2b). If the intended trajectory I_c cannot be verified, the most recently verified trajectory $I_{c-i}^{safe} \| F_{c-i}$ of cycle $c - i$, $i \in \{1, \ldots, c - 1\}$, continues to be executed (see Fig. 2a and the result of $c = 3$ in Fig. 2b). While moving along $I_{c-i}^{safe} \| F_{c-i}$, the fail-safe trajectory F_{c-i} is only executed if no new intended trajectory can be successfully verified before the final time of I_{c-i}^{safe}. This previously verified trajectory $I_{c-i}^{safe} \| F_{c-i}$ remains collision-free as long as other traffic participants do not violate traffic rules, because our set-based prediction has already anticipated all their legal future behaviours. Thus, legal safety is ensured regardless of the verification result.

Experiments on real data. For all verification cycles c in our experiments, the starting time of fail-safe trajectories F_c is equal to the starting time of the next intended trajectory I_{c+1}, that is, $t_c + \Delta_c^{safe} = t_{c+1}$ (see result for $c = 2$ in Fig. 2b). This is achieved by choosing a constant replanning rate $\Delta t = t_{c+1} - t_c$ (meaning that new intended trajectories should be executed at rate Δt) that is set to the constant duration of I_c^{safe} as $\Delta t = \Delta_c^{safe}$ for all c. Consequently, when executing a verified trajectory $I_c^{safe} \| F_c$, the transition to the fail-safe trajectory F_c may only occur at t_{c+1}. Thus, in each time interval $[t_c, t_{c+1}]$, the autonomous vehicle either executes I_c^{safe} completely or a part of F_{c-i} of a previously verified $I_{c-i}^{safe} \| F_{c-i}$. In other words, only if the current verification result is not successful do the autonomous vehicles transition from the safe part of an intended trajectory to a fail-safe trajectory.

In urban environments, most accidents occur at intersections and with pedestrians[53]. To demonstrate that our proposed verification technique allows autonomous vehicles to handle these crucial cases, we created two scenarios by recording real traffic with a BMW 7 series vehicle. By post-processing the real-world recordings, as described in the Supplementary Information, and applying our verification technique offline, we obtained the results presented below. For each of the two scenarios we illustrate an overview of the traffic situation using recorded images from the BMW 7 series vehicle and show the verification results of selected verification cycles c (Figs. 3 and 4). In addition, we demonstrate for both scenarios that our method guarantees legal safety for arbitrary intended trajectory planners (Fig. 5). In the Supplementary Information, we further provide a scenario illustrating safe lane changes (where the third most accidents occur[53]), further results including videos, detailed computation times (177 ms on average), all used parameters and software to visualize the verification results for all verification cycles.

Scenario I: left-turn at an urban intersection. In countries where vehicles drive on the right (we apply this throughout this Article), left turns at intersections are among the most hazardous manoeuvres, because the autonomous vehicle must consider the right of way of oncoming vehicles and yield to potential cyclists in their dedicated lane (Fig. 3a). The behaviour of oncoming vehicles or cyclists may change rapidly over time. For example, vehicles may accelerate or decelerate, and cyclists may even stop and dismount, which increases the uncertainty about the future evolution of the traffic scenario. Under all circumstances, the autonomous vehicle must yield to oncoming traffic while not disrupting the traffic flow due to overly conservative behaviour.

Our method accomplishes this challenge by safeguarding the opportunistic intended trajectory with fail-safe trajectories that (1) comply with the right of way and (2) never stop the autonomous vehicle in the intersection area. Because our prediction accounts for all legal behaviours of other traffic participants, our verification technique can decide whether a left turn manoeuvre can be completed before oncoming traffic can enter the intersection. Thus, the autonomous vehicle automatically respects the right of way.

As illustrated in Fig. 3b at $t_1 = 0$ s, the autonomous vehicle first approaches the intersection along its intended trajectory, that is, I_c^{safe}, $c \in \{1, \ldots, 4\}$, is executed. From $t_5 = 2.4$ s to $t_{10} = 5.4$ s, our

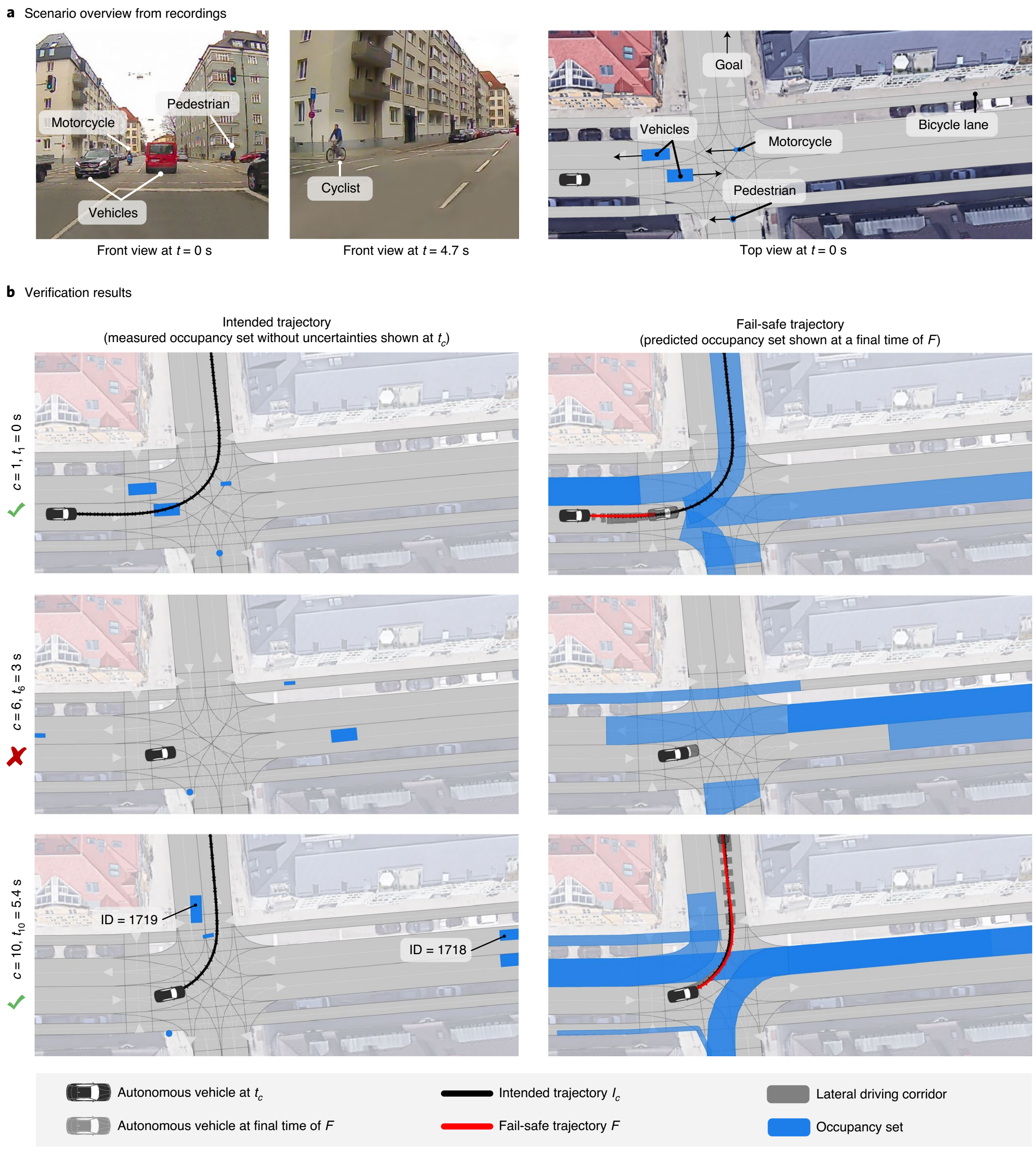


Fig. 3 | Results of Scenario I (urban intersection). a, Camera images and top view of the scenario. **b**, Verification results of selected verification cycles *c*. The intended trajectory I_c is only shown if it is successfully verified. Credit: Google, GeoBasis-DE/BKG (satellite images).

approach automatically detects that the intended trajectories lead to an unsafe situation in which a collision with the oncoming vehicle within the intersection area cannot be excluded before the cyclist has definitely passed. The fail-safe trajectory thus stops the autonomous vehicle at the intersection (see fail-safe trajectory at $t_6 = 3$ s in Fig. 3b). Immediately after the cyclist has passed, our verification technique successfully verifies an intended trajectory and the autonomous vehicle continues its left turn before oncoming traffic, as shown in Fig. 3b at $t_{10} = 5.4$ s. Note that, in this figure, the fail-safe trajectory overlays the occupancy sets, because the occupancy sets are shown at the final time of the fail-safe trajectory (see Supplementary Fig. 8 for the occupancy sets at intermediate times). Figure 3b also demonstrates that our prediction incorporates traffic rules. Consider the occupancy set of the oncoming vehicle with

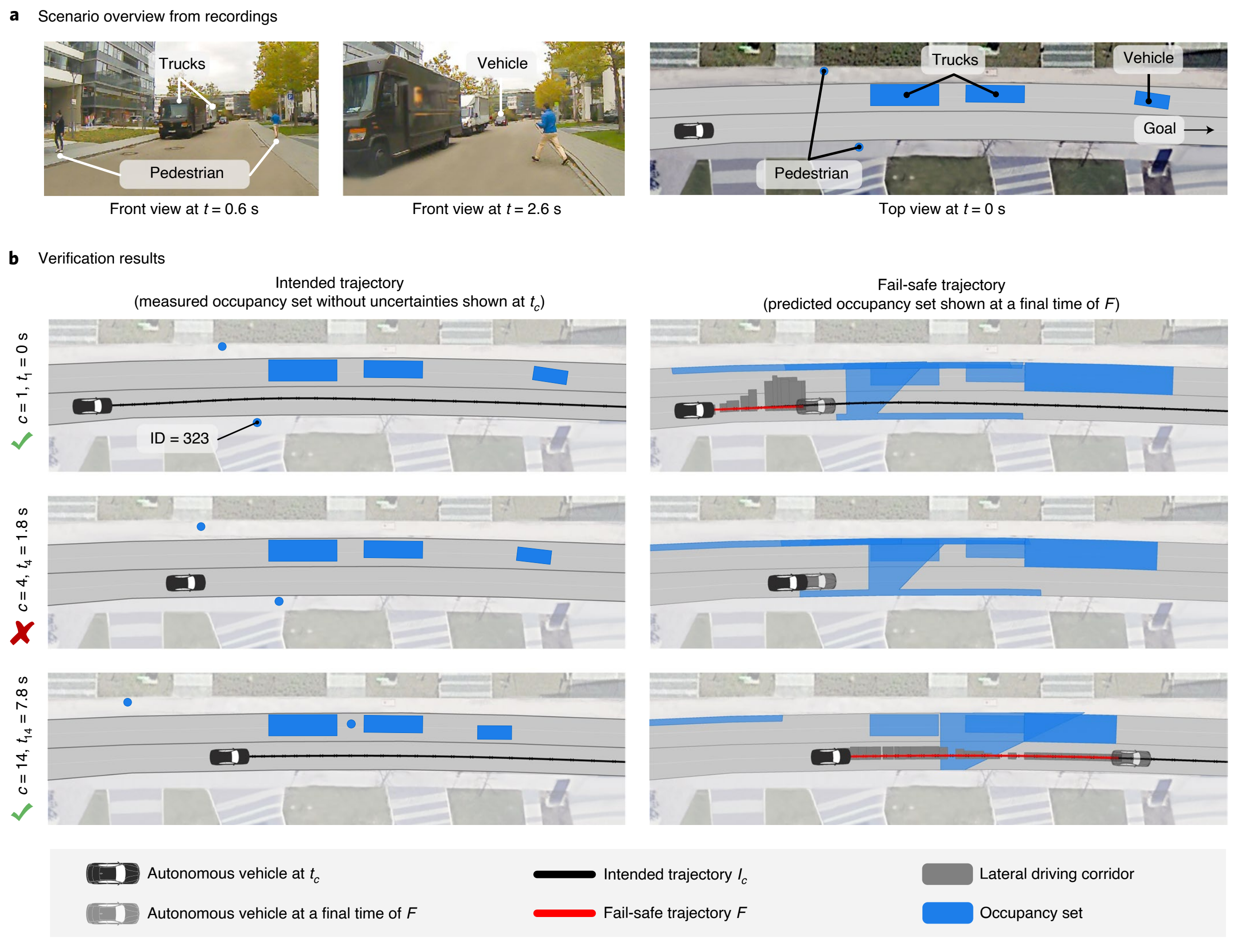


Fig. 4 | Results of Scenario II (jaywalking pedestrian). **a**, Camera images and top view of the scenario. **b**, Verification results of selected verification cycles c. The intended trajectory I_c is only shown if it is successfully verified. Credit: Google, GeoBasis-DE/BKG (satellite images).

ID 1718 at $t_{10} = 5.4$ s. The legal safe distance forbids vehicles to turn after the autonomous vehicle in a way that obstructs the autonomous vehicle. Therefore, the vehicle with ID 1718 is only allowed to continue straight or turn left, but may not yet turn right.

Scenario II: jaywalking pedestrian. Vulnerable road users pose a special challenge to autonomous vehicles, because they often exhibit unexpected changes in behaviour. In particular, pedestrians can quickly change their walking direction, which makes it difficult for autonomous vehicles to react in time. Even though it is illegal for pedestrians to jaywalk, that is, to cross the road in the presence of traffic, pedestrians are occasionally inattentive and cross directly in front of passing vehicles. If the prediction of the autonomous vehicle does not include this behaviour, a fatal accident could occur.

In the first verification cycle $c = 1$ presented in Fig. 4, the pedestrian with ID 323 (in a blue jacket) is walking on the sidewalk and is only looking at his cell phone (Fig. 4a). To anticipate that this inattentive pedestrian may jaywalk, we broaden the set of considered legal behaviours for this pedestrian by relaxing the constraints in its prediction. As a result, the autonomous vehicle computes the future occupancies of this pedestrian for both crossing the road and walking partially on the road parallel to the sidewalk (see occupancy set in Fig. 4b for the fail-safe trajectory at $t_1 = 0$ s; note that occupancy sets of pedestrians are not visualized outside of the road). The resulting fail-safe trajectory F_1 (starting at t_2) ensures that the autonomous vehicle remains behind the pedestrian.

In the next verification cycles $c \in \{2, 3, 4\}$, the autonomous vehicle cannot verify the new intended trajectories. In fact, each intended trajectory collides with the jaywalking pedestrian. Thus, by automatically executing the first computed fail-safe trajectory F_1, the autonomous vehicle slows down to avoid a collision with the pedestrian with ID 323 (see $t_4 = 1.8$ s in Fig. 4b). After the pedestrian crosses, the autonomous vehicle accelerates to the desired velocity, and the fail-safe trajectory implies that the autonomous vehicle is able to pass before the pedestrian may walk back towards the lane of the autonomous vehicle (see $t_{14} = 7.8$ s in Fig. 4b).

As demonstrated in this scenario, our verification technique offers its users, such as mobility providers, the flexibility to define the legal behaviours differently for specific types of traffic participants. For example, when driving past a school, one may wish to anticipate that any child or even any pedestrian may cross the road.

Legal safety for arbitrary intended trajectories. We apply our verification technique to three different intended trajectory planners (for details see Supplementary Information):

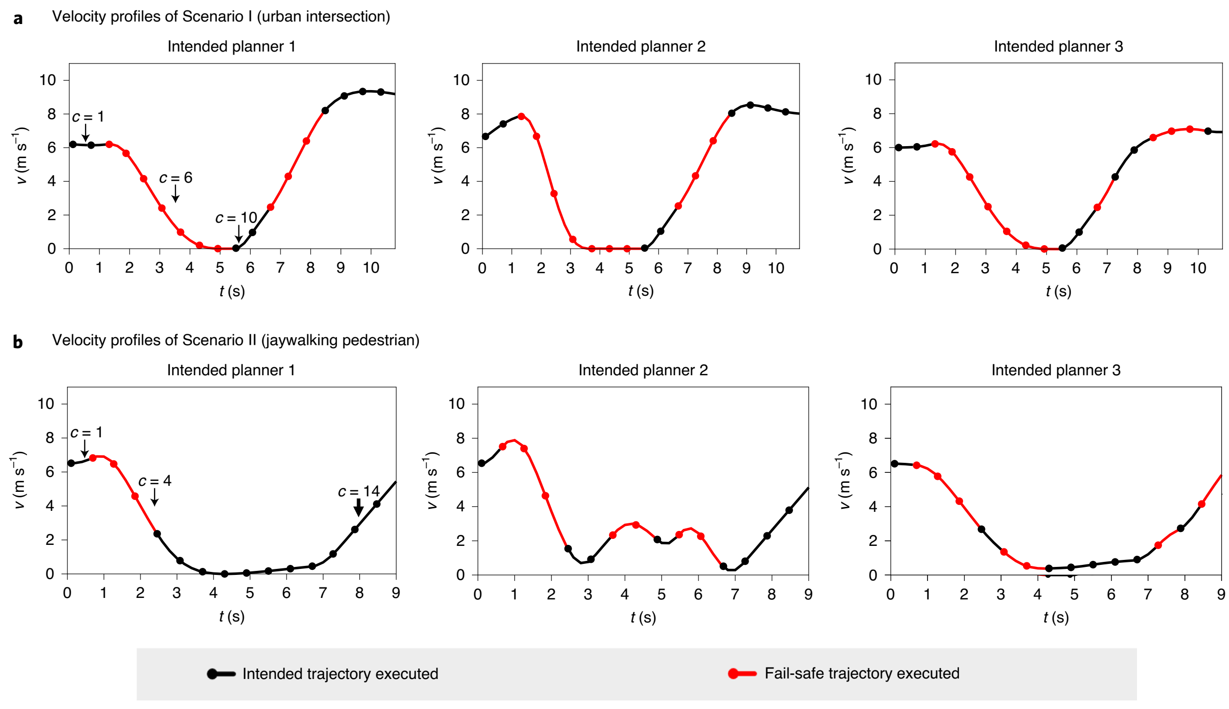


Fig. 5 | Results of the verification technique with different intended planners. a, Executed velocity profiles in Scenario I (results of cycles $c\in\{1, 6, 10\}$ are labelled). **b**, Executed velocity profiles in Scenario II (results of cycles $c\in\{1, 4, 14\}$ are labelled).

- Planner 1 uses continuous optimization to plan trajectories that are collision-free with regard to the most likely behaviour of other traffic participants. This planner is also used as an intended trajectory planner for the previous results of Scenarios I and II.
- Planner 2 is based on Planner 1 with the modification that other traffic participants are ignored. With this planner, we mimic a reinforcement learning approach that has not yet learned collision avoidance.
- Planner 3[39] samples in a discrete state space to plan trajectories that are collision-free with regard to the most likely behaviour of other traffic participants.

Figure 5 illustrates the velocity profiles of the autonomous vehicle in Scenarios I and II for each intended trajectory planner. In Scenario I, our verification technique intervenes independently of the applied intended trajectory planner so that the autonomous vehicle stops in front of the intersection (Fig. 5a). Although Planner 2 is not aware of other traffic participants, our verification technique enables the autonomous vehicle to safely turn left. Because Planner 2 tries to reach the desired velocity ($8\,\mathrm{m\,s^{-1}}$) more aggressively than Planners 1 and 3 (see the results of verification cycles $c\in\{1, 2\}$ in Fig. 5a), the subsequently executed fail-safe trajectories cause a rapid deceleration of the autonomous vehicle (peak, $-6\,\mathrm{m\,s^{-2}}$) (see the results of verification cycles $c\in\{3, \ldots, 8\}$ for Intended Planner 2 in Fig. 5a). However, the execution of fail-safe trajectories for Planner 2 causes only a short delay, as the stopping time at the intersection is less than 2 s.

In Scenario II, the intended trajectory planners are not aware of the pedestrian's intention to jaywalk. Therefore, fail-safe trajectories are executed to slow down the autonomous vehicle (see the results of verification cycles $c\in\{2, 3, 4\}$ in Fig. 5b) until Planners 1 and 3 react to the pedestrian. Planner 2 requires permanent guidance to avoid a collision with the pedestrian. Although the type of executed trajectory, that is, I_c^{safe} or F_{c-i}, continuously alternates, the average velocity of the autonomous vehicle with Planner 2 is 5% higher than that with Planner 1 ($6.36\,\mathrm{m\,s^{-1}}$ and $6.09\,\mathrm{m\,s^{-1}}$, respectively).

In summary, we are able to guarantee legal safety for different intended trajectory planners, even when using a planner that ignores other traffic participants. Furthermore, the resulting velocity profiles are smooth and continuous, as fail-safe trajectories are planned with full consideration of the vehicle's dynamics.

Discussion

Certification is the main obstacle to achieving commercial success with the proposed verification technique. Regulatory guidelines have already been prepared for various domains, such as railway systems, industrial robots and aviation systems, but only limited regulations exist for motion planning of autonomous vehicles (for example, ISO 26262 and ISO 21448). We have prepared the ground for certification by formulating legal safety and presenting a verification technique that ensures that this specification is met during operation of the autonomous vehicle. Moreover, the safety guarantees are maintained when adapting our considered set of traffic rules to new requirements. If legal safety becomes a recognized standard for autonomous vehicles, mobility providers can certify our proposed verification technique for usage in their vehicles. As a result, we expect that societal trust in autonomous vehicles will increase and that testing efforts can be significantly reduced, even if motion planning frameworks for generating intended trajectories are changed.

Legal safety is a promising novel safety approach inspired by traffic regulations that is suitable for certification. Related concepts, such as responsibility-sensitive safety[54], not-at-fault driving[26] and compositional and contract-based verification[55], share our premise to avoid (self-inflicted) accidents, but differ substantially to our proposed solution. Responsibility-sensitive safety assumes that other traffic participants act according to common-sense rules

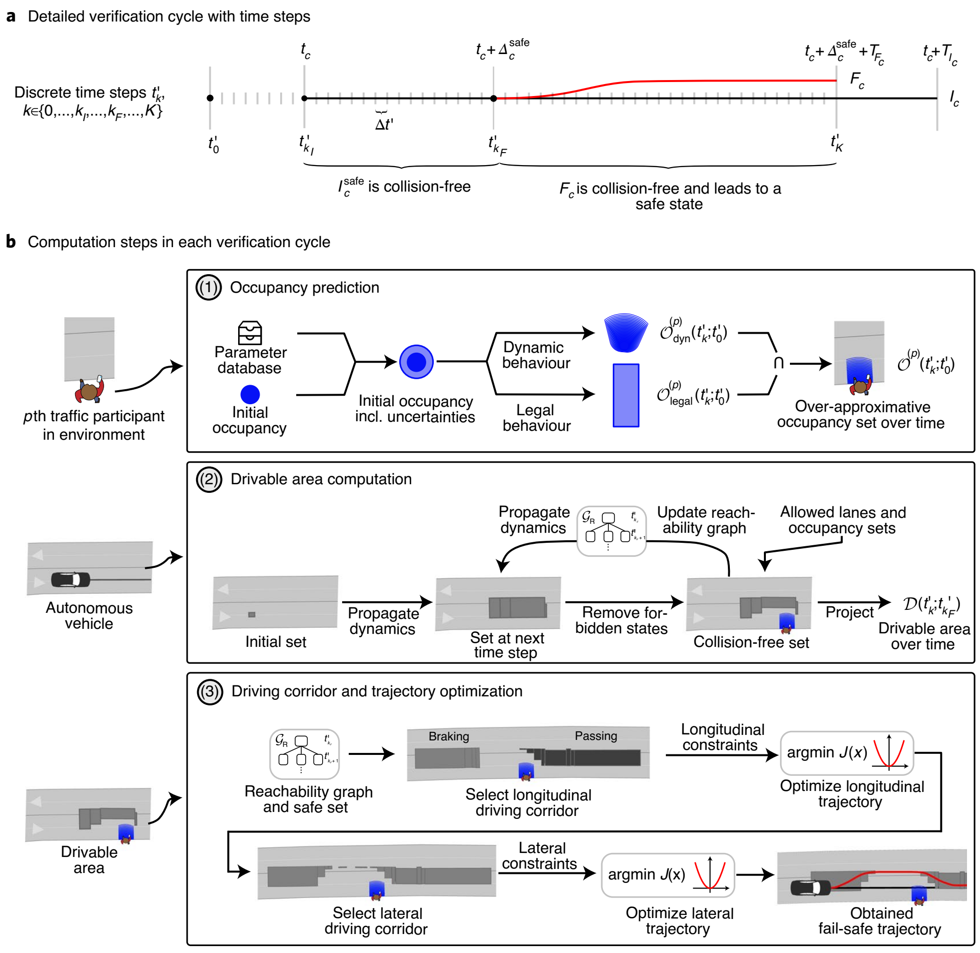

a Detailed verification cycle with time steps
Discrete time steps t'_k, $k\in\{0,\ldots,k_I,\ldots,k_F,\ldots,K\}$
t_c
$t_c+\Delta_c^{\mathrm{safe}}$
$t_c+\Delta_c^{\mathrm{safe}}+T_{F_c}$
$t_c+T_{I_c}$
F_c
I_c
$\Delta t'$
t'_0
t'_{k_I}
t'_{k_F}
t'_K
I_c^{safe} is collision-free
F_c is collision-free and leads to a safe state
b Computation steps in each verification cycle
(1) Occupancy prediction
pth traffic participant in environment
Parameter database
Initial occupancy
Initial occupancy incl. uncertainties
Dynamic behaviour
Legal behaviour
$\mathcal{O}^{(p)}_{\mathrm{dyn}}(t'_k;t'_0)$
$\mathcal{O}^{(p)}_{\mathrm{legal}}(t'_k;t'_0)$
$\cap$
$\mathcal{O}^{(p)}(t'_k;t'_0)$
Over-approximative occupancy set over time
(2) Drivable area computation
Autonomous vehicle
Initial set
Propagate dynamics
Set at next time step
Remove forbidden states
Collision-free set
Propagate dynamics
Update reachability graph
Allowed lanes and occupancy sets
Project
$\mathcal{D}(t'_k;t'_{k_F})$
Drivable area over time
(3) Driving corridor and trajectory optimization
Drivable area
Reachability graph and safe set
Braking
Passing
Select longitudinal driving corridor
Longitudinal constraints
argmin J(x)
Optimize longitudinal trajectory
Select lateral driving corridor
Lateral constraints
argmin J(x)
Optimize lateral trajectory
Obtained fail-safe trajectory

Fig. 6 | Computation steps of the verification technique. a, Time discretization t'_k in one verification cycle c. **b**, Overview of the computation steps for verifying an arbitrary intended trajectory. (1) We compute occupancy sets, that is, all legally occupied positions of other traffic participants over time. (2) The drivable area of the autonomous vehicle is computed to determine fail-safe manoeuvres. (3) Longitudinal and lateral driving corridors are selected from the reachability graph, and longitudinal and lateral trajectories are optimized such that a fail-safe trajectory is obtained.

and defines appropriate responses by the autonomous vehicle based on safe distances. However, despite the execution of appropriate responses, self-inflicted accidents cannot be excluded, because other traffic participants may behave differently than expected. Our approach addresses this problem by considering all legal behaviours. Not-at-fault driving computes a single trajectory that is split into moving, braking and stopped phases and is provably collision-free against a given prediction. By contrast, we allow intended trajectories to be planned independently of fail-safe trajectories, for example, using a most likely prediction to optimize comfort. In ref. [55], a finite number of offline-verified, local models are fitted online to the current traffic situation. However, this approach may result in unsafe behaviours if no valid composition of these local models can be found for the current situation. Our verification technique evaluates the safety of situations online and always provides fail-safe trajectories to eliminate self-inflicted accidents. The detailed computation steps of our verification technique are described in the Methods and are visualized in Fig. 6.

Methods

Formal verification is often believed to cause performance drops (for example, lower average velocities resulting in longer travel times) and conservative behaviour in robotic systems[56,57]. However, we believe that autonomous vehicles can offer high performance and ensure legal safety at the same time. This has motivated us to improve on our previous work on set-based predictions[58–60], fail-safe trajectory planning[61] and trajectory planning using reachable sets[62]. Further to our previous work, we present the following innovations:

1. Our proposed verification technique ensures legal safety in complex traffic scenarios and in a computationally efficient way. In particular, by embedding driving corridors[62] into fail-safe trajectory planning[61], we generalize the computation of possible fail-safe manoeuvre options to different traffic situations and can consider multiple safe terminal sets.
2. On various urban scenarios that have been recorded in real traffic including measurement uncertainties, the applicability of the proposed verification technique is demonstrated. In addition, our results indicate that non-conservative driving behaviour can be achieved despite the over-approximative, set-based prediction.
3. The temporal interplay over subsequent verification cycles of our verification technique with the intended trajectory planner of the autonomous vehicle is presented in detail.
4. Further experiments with three different intended trajectory planners validate that our verification technique is able to ensure legal safety for arbitrary intended trajectory planners.

In the following paragraphs, we present the inputs of our verification technique, preliminaries for the reachability analysis, an overview of the algorithmic steps and the safety guarantees for our verification technique. Additional details are provided in the Supplementary Information.

Inputs of the verification technique. Our verification technique is integrated between the motion planning layer and the control layer of the autonomous vehicle (see planning frameworks in refs. [63,64]). In each verification cycle c, our verification technique receives as inputs the intended trajectory I_c and the environment model. The intended trajectories must be kinematically feasible and branch off the previously verified trajectory $I_{c-i}^{\mathrm{safe}} \parallel F_{c-i}$. The environment model must contain the lanes of the road, pedestrian crossings and areas in which the autonomous vehicle is not allowed to stop, which are used to obtain the designated safe areas. For all safety-relevant traffic participants, the environment model must contain their type (that is, vehicle, motorcycle, bicycle or pedestrian) and their current states (that is, a set containing the exact state and bounded measurement uncertainties). If the type of traffic participant is unknown or uncertain, our verification technique can predict the set of future behaviours for all possible types in parallel.

Preliminaries of the verification technique. The motion of the pth traffic participant is governed by the differential equation $\dot{x}^{(p)}(t) = f^{(p)}\big(x^{(p)}(t), u^{(p)}(t)\big)$, where $x^{(p)}$ is the state and $u^{(p)}$ is the input. The admissible states and inputs are bounded by the respective sets $\mathcal{X}^{(p)}(t) \subset \mathbb{R}^{n^{(p)}}$ and $\mathcal{U}^{(p)}(t) \subset \mathbb{R}^{m^{(p)}}$. A possible solution of the differential equation at time t is denoted by $\chi^{(p)}\big(t; x^{(p)}(\tau_0), u^{(p)}(\cdot)\big)$, when starting at state $x^{(p)}(\tau_0) \in \mathcal{X}_0^{(p)}$, where $\mathcal{X}_0^{(p)}$ is the set of states at an initial time τ_0 including measurement uncertainties, and using input trajectory $u^{(p)}(\cdot)$. The reachable set $\mathcal{R}^{e^{(p)}}(t;\tau_0) \subseteq \mathcal{X}^{(p)}(t)$ describes the set of states that are reachable by the pth traffic participant at a certain point in time $t \geq \tau_0$ when starting in $\mathcal{X}_0^{(p)}$ and applying all admissible inputs $\mathcal{U}^{(p)}(t)$:

$$\mathcal{R}^{e^{(p)}}(t;\tau_0) = \Big\{\chi^{(p)}\big(t; x^{(p)}(\tau_0), u^{(p)}(\cdot)\big) \;\Big|\; x^{(p)}(\tau_0) \in \mathcal{X}_0^{(p)}, \forall \tilde{\tau} \in [\tau_0, t] : \chi^{(p)}\big(\tilde{\tau}; x^{(p)}(\tau_0), u^{(p)}(\cdot)\big) \in \mathcal{X}^{(p)}(\tilde{\tau}), u^{(p)}(\tilde{\tau}) \in \mathcal{U}^{(p)}(\tilde{\tau})\Big\} \tag{1}$$

For brevity, we omit the superscript (p) when referring to the autonomous vehicle. In each verification cycle c, we compute the reachable set of other traffic participants to predict their future movement and that of the autonomous vehicle to obtain its drivable area.

As illustrated in Fig. 6a, we introduce the discrete points in time t'_k for each verification cycle c, where $k \in \{0, \ldots, k_I, \ldots, k_F, \ldots K\} \subseteq \mathbb{N}_0$; for brevity, the notation of t'_k does not reflect its dependency on c. Time t'_0 is the initial time of the prediction, that is, the point in time at which the most recently available environment model has been recorded. Time t'_{k_I} corresponds to the start time of the intended trajectory I_c (that is, $t'_{k_I} = t_c$), t'_{k_F} corresponds to the start time of the fail-safe trajectory F_c (that is, $t'_{k_F} = t_c + \Delta_c^{\mathrm{safe}}$) and t'_K corresponds to the final time of the fail-safe trajectory (that is, $t'_K = t_c + \Delta_c^{\mathrm{safe}} + T_{F_c}$). Without loss of generality, we assume that the times t'_k are multiples of the time step size $\Delta t' \in \mathbb{R}_+$, that is, $t'_k = t'_0 + k\Delta t'$.

Recall that we set Δ_c^{safe} to the replanning rate Δt in our experiments. To minimize the interventions of our verification technique, that is, how often a fail-safe trajectory is executed, the duration Δ_c^{safe} can be dynamically adjusted to optimize the length of I_c^{safe} as described in ref. [65]. To avoid that new intended trajectories cannot be verified solely due to a timeout, intended trajectories I_c should be provided prior to $t_c - \Delta^{\mathrm{verify}}$, where $\Delta^{\mathrm{verify}} \in \mathbb{R}_+$ is the required computation time of our verification method.

Occupancy prediction. The goal in the first step of our verification technique is to over-approximate the area $\mathcal{L}^e(t)$ that exactly encloses the occupied positions of the surrounding traffic participants for all their legal behaviours. Therefore, we first compute all dynamically feasible behaviours and subsequently remove illegal behaviours.

All dynamically feasible behaviours of other traffic participants are obtained using reachability analysis as defined in equation (1). For each pth traffic participant, the environmental model provides the initial states $\mathcal{X}_0^{(p)}$ at t'_0, which are described by a set due to measurement uncertainties (Fig. 6b, step (1)). The dynamics of each traffic participant are abstracted by a second-order integrator model with bounded velocities and accelerations. We compute the reachable set $\mathcal{R}^{(p)}(t;t'_0)$ as a tight over-approximation of the exact reachable set, that is, $\mathcal{R}^{(p)}(t;t'_0) \supseteq \mathcal{R}^{e^{(p)}}(t;t'_0)$, and only for the position domain to allow for an efficient computation. For collision checks with planned trajectories of the autonomous vehicle, we introduce $\mathcal{O}_{\mathrm{dyn}}^{(p)}(t;t'_0)$ as the dynamics-based occupancy set resulting from the over-approximative reachable set $\mathcal{R}^{(p)}(t;t'_0)$ by considering the dimensions of the pth traffic participant (Fig. 6b, step (1)).

Next, we remove behaviours that are not allowed according to traffic rules. Therefore, we formalize a set of traffic rules that is most relevant for motion planning (and which can be easily extended). Let $v^{(p)}$ and $a^{(p)}$ denote the velocity and acceleration of the pth predicted traffic participant, respectively, and $\diamondsuit^{\mathrm{veh}}$ denotes that the parameter $\diamondsuit \in \{\overline{v}, \overline{a}\}$ bounding the velocity or acceleration is applicable for vehicles and motorcycles, while $\diamondsuit^{\mathrm{cyc}}$ is for bicycles and $\diamondsuit^{\mathrm{ped}}$ is for pedestrians (the values of the parameters are stored in a database generated offline, can be updated online, and are provided in the Supplementary Information). The considered traffic rules for vehicles, motorcycles and bicycles are as follows:

- Maximum velocity is bounded (article 13.2 of ref. [7]): $v^{(p)} \leq v_{\mathrm{limit}} f_S^{(p)}$, where v_{limit} is the legal speed limit of the road and $f_S^{(p)} \geq 1$ is a parameterized speeding factor to consider slight over-speeding. If no speed limit is available, such as for bicycles, $v^{(p)} \leq \overline{v}^{\mathrm{veh/cyc}}$.
- Driving backward is not allowed (article 14.2 of ref. [7]): $v^{(p)} \geq 0$.
- Absolute acceleration is bounded (due to tyre friction): $|a^{(p)}| \leq \overline{a}^{\mathrm{veh/cyc}}$.
- Leaving the road is forbidden (article 14.1 of ref. [7]).
- A safe distance to the autonomous vehicle must be maintained when driving behind it or merging in front of it (articles 13.5 and 11.2d of ref. [7]).
- Changing lanes is only allowed if the new lane has the same driving direction as the previous one (article 11.2c of ref. [7]).

Note that, according to article 11.2c of ref. [7], overtaking in a lane not appropriate to the direction of traffic is only allowed if not endangering or interfering with oncoming traffic. Because such a legal overtaking manoeuvre does not interfere with the motion planning of the autonomous vehicle, we neglect it in our prediction without compromising legal safety.

Although pedestrians are generally not allowed to obstruct vehicular traffic, for example, to jaywalk (article 7.1 of ref. [7]), vehicles are required to take precautions to avoid endangering pedestrians (article 21.1 of ref. [7]). Thus, the considered traffic rules for pedestrians are as follows:

- Absolute velocity is bounded (for example, based on ISO 13855): $|v^{(p)}| \leq \overline{v}^{\mathrm{ped}}$.
- Absolute acceleration is bounded (due to physical capabilities): $|a^{(p)}| \leq \overline{a}^{\mathrm{ped}}$.
- Entering the road is forbidden (articles 7.1 and 20.2 of ref. [7]) except
    - on pedestrian crossings (articles 20.6b and 21.2 of ref. [7])
    - when walking toward the road; then, crossing the road is allowed perpendicularly with a deviation of angle α based on the current heading of the pedestrian (articles 20.6c,d of ref. [7])
    - when walking parallel to the road; then, occupying the strip of the road edge with a width of d_{slack} is allowed, for example, to avoid obstacles on the sidewalk (articles 20.2a, 20.3 and 20.4 of ref. [7]).

In summary, our set of traffic rules either constrains the dynamics of other traffic participants (for example, their maximum velocity), which are considered by $\mathcal{O}_{\mathrm{dyn}}^{(p)}(t;t'_0)$, or constrains the allowed regions in the environment (for example, certain lanes or pedestrian crossings), which are given by the environment model and are denoted by $\mathcal{O}_{\mathrm{legal}}^{(p)}(t;t'_0)$. The resulting over-approximative occupancy set of the pth traffic participant is $\mathcal{O}^{(p)}(t;t'_0) = \mathcal{O}_{\mathrm{dyn}}^{(p)}(t;t'_0) \cap \mathcal{O}_{\mathrm{legal}}^{(p)}(t;t'_0)$ (Fig. 6b, step (1)). To verify that I_c^{safe} and F_c are collision-free, we compute the occupancy sets for consecutive time intervals $[t'_k, t'_{k+1}]$ until the final time of F_c, that is, $\forall\, k \in \{k_I, \ldots, K\}$. Note that the time intervals $[t'_k, t'_{k+1}]$ can be of different duration for each k, for example, in case I_c^{safe} and F_c are discretized differently. The predicted occupancy sets of all traffic participants are given by $\mathcal{L}([t'_k, t'_{k+1}]) = \bigcup_p \bigcup_{t \in [t'_k, t'_{k+1}]} \mathcal{O}^{(p)}(t;t'_0)$.

Note that, regardless of how many traffic rules we consider, our prediction always over-approximates the exact set of all legal behaviours, that is, $\mathcal{L}(t) \supseteq \mathcal{L}^e(t)$. The reason is that only behaviours defined as illegal are removed from the over-approximation of all dynamically feasible behaviours. The fewer traffic rules we consider, the more cautiously the autonomous vehicle behaves, because it respects more behaviours than actually allowed according to all traffic rules. However, the autonomous vehicle definitely remains collision-free when other traffic participants adhere to all traffic rules, as prescribed by legal safety. If a collision occurs nonetheless, we can verifiably argue that another traffic participant must have violated traffic rules and that the collision is not self-inflicted by the autonomous vehicle. Nevertheless, we account for humans' tendency to violate traffic rules, such as the speed limit. Therefore, we continuously monitor whether any traffic participant performs a behaviour that is not included in the set of legal behaviours. Whenever violations are detected, this behaviour is automatically added to the prediction result; for example, if another vehicle illegally changes lanes, we no longer exclude this behaviour from our prediction of this vehicle. As a result, our verification technique will attempt to find a new fail-safe trajectory in case the previous one is no longer collision-free. Furthermore, if a traffic participant appears likely to misbehave, such behaviours can be included in our prediction by disabling the corresponding constraint, as demonstrated in Scenario II.

Drivable area computation. To obtain possible sequences of high-level fail-safe manoeuvres (for example, overtaking other vehicles on their left or right), we compute the drivable area of the autonomous vehicle at discrete points in time t'_k with $k \geq k_F$ by projecting its reachable set $\mathcal{R}^e(t'_k; t'_{k_F})$ defined in equation (1) onto the position domain (Fig. 6b, step (2)). As for the prediction of other traffic participants, we abstract the dynamics of the autonomous vehicle using two second-order integrator models in the longitudinal and lateral directions with bounded velocities and accelerations in a road-aligned coordinate system[66]. For computational efficiency, the reachable set is approximated through the union of base sets $\mathcal{B}_k^{(i)}$, $i \in \mathbb{N}_0$, such that $\mathcal{R}^e(t'_k; t'_{k_F}) \approx \bigcup_i \mathcal{B}_k^{(i)}$ holds. The base sets $\mathcal{B}_k^{(i)}$ are the Cartesian products of convex polytopes describing reachable position–velocity pairs in the longitudinal and lateral directions. We use convex polytopes, because they are closed under required set operations such as Minkowski sum, linear mapping and intersection. The projection of base sets $\mathcal{B}_k^{(i)}$ onto the position domain yields axis-aligned rectangles $\mathcal{D}_k^{(i)}$ that represent the drivable area $\mathcal{D}(t'_k; t'_{k_F}) := \bigcup_i \mathcal{D}_k^{(i)}$. The projection of the reachable set onto the position domain can be computed efficiently, because we only need to determine the minimum and maximum position coordinates of the convex polytopes of the base sets $\mathcal{B}_k^{(i)}$.

The state $x(t'_{k_F})$ of the fail-safe trajectory F_c at its start time t'_{k_F} is provided by the final state of I_c^{safe}. We enclose $x(t'_{k_F})$ with a base set such that $x(t'_{k_F}) \in \mathcal{B}_{k_F}^{(0)}$ holds. The reachable set of consecutive points in time t'_{k+1}, $k \geq k_F$, is computed as illustrated in Fig. 6b (step (2)). First, we propagate each base set $\mathcal{B}_k^{(i)}$ of the previous time step forward in time considering all admissible inputs. Second, we remove states outside the set of admissible states $\mathcal{X}(t'_{k+1})$, that is, positions in which the autonomous vehicle collides with the predicted occupancy sets $\mathcal{L}([t'_k, t'_{k+1}])$ or the area $\mathcal{Q}$ outside of the road, to obtain $\mathcal{R}(t'_{k+1}; t'_{k_F}) \approx \bigcup_j \mathcal{B}_{k+1}^{(j)}$ at time t'_{k+1}. Third, we store each base set $\mathcal{B}_{k+1}^{(j)}$ in a directed graph $\mathcal{G}_{\mathcal{R}}$. In $\mathcal{G}_{\mathcal{R}}$, each set $\mathcal{B}_{k+1}^{(j)}$ is associated with exactly one node and an edge indicates that base set $\mathcal{B}_{k+1}^{(j)}$ is reachable from $\mathcal{B}_k^{(i)}$ within one time step. The procedure is repeated until the final time step t'_K is reached.

Driving corridor and trajectory optimization. We generate drivable fail-safe trajectories through continuous optimization. As convex optimization problems can be solved efficiently with global convergence, we convexify the inherently non-convex optimization problem by separating the longitudinal and lateral motion of the autonomous vehicle. However, longitudinal motion planning requires prior knowledge on the lateral motion and vice versa, as both subsystems are dynamically coupled. To overcome this issue, we obtain driving corridors from the drivable area that provide spatio-temporal position constraints for the optimization problems. We refer to the driving corridors for longitudinal and lateral optimization as the longitudinal and lateral driving corridors, respectively. To ensure legal safety for an infinite time horizon, we constrain the driving corridors to end in a safe terminal state based on the designated safe areas, for example, a standstill in the rightmost lane sufficiently far from an intersection. As illustrated in Fig. 6b (step (3)), our motion planner first optimizes the longitudinal trajectory within a longitudinal driving corridor, followed by optimizing the lateral trajectory in a suitable lateral driving corridor. Currently, we constrain fail-safe trajectories to be kinematically feasible, collision-free with respect to road boundaries and the predicted occupancy sets, respect the speed limit and end in a safe state. Further constraints can be imposed to consider additional properties, for example, rules on overtaking or stopping at the boundaries of the field of view of the vehicle.

We represent collision avoidance constraints by a minimum and maximum value on the longitudinal or lateral positions at each point in time. To obtain these limits, we exploit that a connected set in the position domain projected onto either the longitudinal or lateral direction yields an interval. Consequently, we define a longitudinal corridor and a lateral driving corridor for fail-safe motion planning as a temporal sequence of connected sets that are subsets of the drivable area $\mathcal{D}(t'_k; t'_{k_F})$ from time t'_{k_F} to the final time t'_K.

To determine longitudinal driving corridors, we perform a search on the reachability graph $\mathcal{G}_{\mathcal{R}}$ backwards in time starting from the set of safe terminal states (Fig. 6b, step (3)). There may be multiple longitudinal driving corridors, because the drivable area can be disconnected due to surrounding traffic participants. We select the longitudinal driving corridor with the greatest cumulative drivable area from t'_{k_F} to t'_K for trajectory planning (other heuristics can also be applied). For the longitudinal trajectory optimization, we use a fourth-order integrator model with jounce as input and bounded longitudinal velocity, acceleration and jerk. In addition to the collision avoidance constraints from the boundary of the longitudinal driving corridor, the autonomous vehicle must come to a standstill at the final time t'_K. To improve comfort, we choose a quadratic cost function that minimizes acceleration, jerk and jounce as well as deviations from the desired velocity.

The computation and selection of lateral driving corridors are performed similarly to the computation and selection of longitudinal driving corridors with the addition that the connected sets of the lateral driving corridor must provide a unique passing side for each obstacle. The lateral trajectories of the autonomous vehicle are optimized with respect to a linearized kinematic single-track model with limits on the steering actuators. Analogously to planning in the longitudinal direction, the position constraints for collision avoidance are obtained from the boundaries of the lateral driving corridor. We select a quadratic cost function to minimize the lateral distance and orientation deviation from a given reference path and to punish high curvature rates for comfort.

In the case that trajectory optimization is infeasible using the selected lateral or longitudinal driving corridor, we select a driving corridor with the next highest cumulative drivable area for optimization until either a fail-safe trajectory is identified or no further driving corridors remain. In the rare event that no feasible fail-safe trajectory is found, the previously verified trajectory is further executed.

Guarantees of our verification technique. To comply with legal safety, autonomous vehicles must not collide with any legal behaviour of other traffic participants:

$$\forall t \geq t_0 : occ(x(t)) \cap (\mathcal{L}^e(t) \cup \mathcal{Q}) = \emptyset \qquad (2)$$

where the operator $occ(x)$ relates the state x of the autonomous vehicle to the set of occupied points in the position domain as $occ(x) : \mathcal{X} \rightarrow Pow(\mathbb{R}^n)$, where $Pow(\mathbb{R}^n)$ is the power set of $\mathbb{R}^n$.

Using the principle of induction, we sketch the proof that our technique ensures legal safety according to equation (2). For the base case ($c=1$), for $t \geq t_0$, the autonomous vehicle is initially in a safe state in which it can remain. Only if I_c can be successfully verified will the autonomous vehicle start executing $I_c^{\text{safe}}||F_c$ from t_c. This trajectory is collision-free at all discrete time steps $t'_k \in [t_c, t_c + \Delta_c^{\text{safe}} + T_{F_c}]$ against all legal behaviours $\mathcal{L}(t) \supseteq \mathcal{L}^e(t)$ of other traffic participants and the area $\mathcal{Q}$ outside the road. If no new intended trajectory can be successfully verified in a subsequent verification cycle before $t_c + \Delta_c^{\text{safe}} + T_{F_c}$, the fail-safe trajectory F_c transitions the autonomous vehicle to a standstill in a safe terminal state at $t_c + \Delta_c^{\text{safe}} + T_{F_c}$, which is legally safe for all future times. For the inductive step, assuming that the verification result of cycle $c=r$, for any $r \in \mathbb{N}_+$, ensures legal safety, we show that legal safety is also ensured regardless of the verification result of cycle $c+1$. If the verification is unsuccessful, the autonomous vehicle continues to execute the trajectory $I_{c-i}^{\text{safe}}||F_{c-i}, i \in \{0, \ldots, c-1\}$ of the previous cycle c that ensures legal safety by definition. If the verification is successful in cycle $c+1$, the autonomous vehicle executes $I_{c+1}^{\text{safe}}||F_{c+1}$ from t_{c+1}. In this case, we can apply the same reasoning as in the base case to demonstrate that legal safety is also ensured from t_{c+1} with the verified trajectory $I_{c+1}^{\text{safe}}||F_{c+1}$.

To ensure that the autonomous vehicle is collision-free along I^{safe} and F in continuous time and despite control disturbances and model uncertainties, we refer to the approach in ref. [67].

Data availability

All data gathered and reported in this study are available in the Supplementary data file. This includes the environment model, the intended trajectory and the verification result of each verification cycle for all scenarios.

Code availability

The code to visualize and analyse the gathered data and obtained results of this study are included in the Supplementary data file.

Received: 30 October 2019; Accepted: 4 August 2020;
Published online: 14 September 2020

References

1. Favarò, F., Eurich, S. & Nader, N. Autonomous vehicles' disengagements: trends, triggers and regulatory limitations. *Accid. Anal. Prev.* **110**, 136–148 (2018).
2. Anderson, J. M. et al. *Autonomous Vehicle Technology: A Guide for Policymakers* (Rand Corporation, 2016).
3. Koopman, P. & Wagner, M. Autonomous vehicle safety: an interdisciplinary challenge. *IEEE Intell. Transportation Syst. Mag.* **9**, 90–96 (2017).
4. Kalra, N. & Paddock, S. M. Driving to safety: how many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Res. A Policy Practice* **94**, 182–193 (2016).
5. Seshia, S. A., Sadigh, D. & Sastry, S. S. Towards verified artificial intelligence. Preprint at https://arxiv.org/abs/1606.08514 (2017).
6. Schwarting, W., Alonso-Mora, J. & Rus, D. Planning and decision-making for autonomous vehicles. *Annu. Rev. Control Robot. Autonomous Syst.* **1**, 187–210 (2018).
7. United Nations Economic Commission for Europe. *Convention on Road Traffic. United Nations Conference on Road Traffic* (United Nations, 1968); consolidated version of 2006.
8. Vanholme, B., Gruyer, D., Lusetti, B., Glaser, S. & Mammar, S. Highly automated driving on highways based on legal safety. *IEEE Trans. Intell. Transportation Syst.* **14**, 333–347 (2013).
9. Althoff, M. & Dolan, J. M. Online verification of automated road vehicles using reachability analysis. *IEEE Trans. Robotics* **30**, 903–918 (2014).
10. Koopman, P. & Wagner, M. Challenges in autonomous vehicle testing and validation. *SAE Int. J. Transportation Safety* **4**, 15–24 (2016).
11. Dahl, J., de Campos, G. R., Olsson, C. & Fredriksson, J. Collision avoidance: a literature review on threat-assessment techniques. *IEEE Trans. Intell. Vehicles* **4**, 101–113 (2019).
12. Tumova, J., Hall, G. C., Karaman, S., Frazzoli, E. & Rus, D. Least-violating control strategy synthesis with safety rules. In *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control* 1–10 (HSCC, 2013).
13. Kress-Gazit, H., Fainekos, G. E. & Pappas, G. J. Temporal-logic-based reactive mission and motion planning. *IEEE Trans. Robotics* **25**, 1370–1381 (2009).
14. Fraichard, T. & Asama, H. Inevitable collision states—a step towards safer robots? In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* 388–393 (IEEE, 2003).
15. Chan, N., Kuffner, J. & Zucker, M. Improved motion planning speed and safety using regions of inevitable collision. In *17th CISM-IFToMM Symposium on Robot Design, Dynamics and Control* 103–114 (Springer, 2008).
16. Koller, T., Berkenkamp, F., Turchetta, M. & Krause, A. Learning-based model predictive control for safe exploration. In *Proceedings of the 2018 IEEE International Conference on Decision and Control* 6059–6066 (IEEE, 2018).

17. Wabersich, K. P. & Zeilinger, M. N. Linear model predictive safety certification for learning-based control. In *Proceedings of the IEEE International Conference on Decision and Control* 7130–7135 (IEEE, 2018).
18. Sadraddini, S. & Belta, C. A provably correct MPC approach to safety control of urban traffic networks. In *Proceedings of the American Control Conference* 1679–1684 (2016).
19. Ames, A. D. et al. Control barrier functions: theory and applications. In *Proceedings of the 18th European Control Conference* 3420–3431 (IEEE, 2019).
20. Tedrake, R., Manchester, I. R., Tobenkin, M. & Roberts, J. W. LQR-trees: feedback motion planning via sums-of-squares verification. *Int. J. Robotics Res.* **29**, 1038–1052 (2010).
21. Li, W., Sadigh, D., Sastry, S. S. & Seshia, S. A. Synthesis for human-in-the-loop control systems. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems* 470–484 (Springer, 2014).
22. Jalalmaab, M., Fidan, B., Jeon, S. & Falcone, P. Guaranteeing persistent feasibility of model predictive motion planning for autonomous vehicles. In *Proceedings of the 2017 IEEE Intelligent Vehicles Symposium* 843–848 (IEEE, 2017).
23. Danielson, C., Weiss, A., Berntorp, K. & Di Cairano, S. Path planning using positive invariant sets. In *Proceedings of the 55th International Conference on Decision and Control* 5986–5991 (IEEE, 2016).
24. Herbert, S. L. et al. FaSTrack: a modular framework for fast and guaranteed safe motion planning. In *Proceedings of the 56th International Conference on Decision and Control* 1517–1522 (IEEE, 2017).
25. Falcone, P., Ali, M. & Sjöberg, J. Predictive threat assessment via reachability analysis and set invariance theory. *IEEE Trans. Intell. Transportation Syst.* **12**, 1352–1361 (2011).
26. Vaskov, S. et al. Towards provably not-at-fault control of autonomous robots in arbitrary dynamic environments. In *Proc. Robotics: Science and Systems* 1–9 (2019).
27. Lefèvre, S., Vasquez, D. & Laugier, C. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH J.* **1**, 1–14 (2014).
28. Gindele, T., Brechtel, S. & Dillmann, R. Learning driver behavior models from traffic observations for decision making and planning. *IEEE Intell. Transportation Syst. Mag.* **7**, 69–79 (2015).
29. Bahram, M., Hubmann, C., Lawitzky, A., Aeberhard, M. & Wollherr, D. A combined model- and learning-based framework for interaction-aware maneuver prediction. *IEEE Trans. Intell. Transportation Syst.* **17**, 1538–1550 (2016).
30. Deo, N., Rangesh, A. & Trivedi, M. M. How would surround vehicles move? A unified framework for maneuver classification and motion prediction. *IEEE Trans. Intell. Vehicles* **3**, 129–140 (2018).
31. Ghahramani, Z. Probabilistic machine learning and artificial intelligence. *Nature* **521**, 452–459 (2015).
32. Tang, C., Chen, J. & Tomizuka, M. Adaptive probabilistic vehicle trajectory prediction through physically feasible Bayesian recurrent neural network. In *Proceedings of the 2019 IEEE International Conference on Robotics and Automation* 3846–3852 (IEEE, 2019).
33. Pool, E. A. I., Kooij, J. F. P. & Gavrila, D. M. Context-based cyclist path prediction using recurrent neural networks. In *Proceedings of the 2019 IEEE Intelligent Vehicles Symposium* 824–830 (IEEE, 2019).
34. Wu, A. & How, J. Guaranteed infinite horizon avoidance of unpredictable, dynamically constrained obstacles. *Autonomous Robots* **32**, 227–242 (2012).
35. Bouraine, S., Fraichard, T. & Salhi, H. Provably safe navigation for mobile robots with limited field-of-views in dynamic environments. *Autonomous Robots* **32**, 267–283 (2012).
36. Yang, Y., Zhang, J., Cai, K. & Prandini, M. Multi-aircraft conflict detection and resolution based on probabilistic reach sets. *IEEE Trans. Control Syst. Technol.* **25**, 309–316 (2017).
37. Nager, Y., Censi, A. & Frazzoli, E. What lies in the shadows? Safe and computation-aware motion planning for autonomous vehicles using intent-aware dynamic shadow regions. In *Proceedings of the 2019 IEEE International Conference on Robotics and Automation* 5800–5806 (IEEE, 2019).
38. McNaughton, M., Urmson, C., Dolan, J. M. & Lee, J.-W. Motion planning for autonomous driving with a conformal spatiotemporal lattice. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation* 4889–4895 (IEEE, 2011).
39. Werling, M., Kammel, S., Ziegler, J. & Gröll, L. Optimal trajectories for time-critical street scenarios using discretized terminal manifolds. *Int. J. Robotics Res.* **31**, 346–359 (2012).
40. Zucker, M. et al. CHOMP: covariant Hamiltonian optimization for motion planning. *Int. J. Robotics Res.* **32**, 1164–1193 (2013).
41. Ziegler, J., Bender, P., Dang, T. & Stiller, C. Trajectory planning for Bertha—a local, continuous method. In *Proceedings of the 2014 IEEE Intelligent Vehicles Symposium* 450–457 (IEEE, 2014).
42. Hult, R., Zanon, M., Gros, S. & Falcone, P. An MIQP-based heuristic for optimal coordination of vehicles at intersections. In *Proceedings of the 2018 IEEE International Conference on Decision and Control* 2783–2790 (IEEE, 2018).
43. Sun, Z., Hsu, D., Jiang, T., Kurniawati, H. & Reif, J. H. Narrow passage sampling for probabilistic roadmap planning. *IEEE Trans. Robotics* **21**, 1105–1115 (2005).
44. LaValle, S. M. in *Planning Algorithms* 79–80 (Cambridge Univ. Press, 2006).
45. Schouwenaars, T., De Moor, B., Feron, E. & How, J. Mixed integer programming for multi-vehicle path planning. In *Proceedings of the 2001 European Control Conference* 2603–2608 (IEEE, 2001).
46. Qian, X., Altché, F., Bender, P., Stiller, C. & de La Fortelle, A. Optimal trajectory planning for autonomous driving integrating logical constraints: an MIQP perspective. In *Proceedings of the IEEE 19th International Conference on Intelligent Transportation Systems* 205–210 (IEEE, 2016).
47. Park, J., Karumanchi, S. & Iagnemma, K. Homotopy-based divide-and-conquer strategy for optimal trajectory planning via mixed-integer programming. *IEEE Trans. Robotics* **31**, 1101–1115 (2015).
48. Gutjahr, B., Gröll, L. & Werling, M. Lateral vehicle trajectory optimization using constrained linear time-varying MPC. *IEEE Trans. Intell. Transportation Syst.* **18**, 1586–1595 (2016).
49. Zhan, W., Chen, J., Chan, C.-Y., Liu, C. & Tomizuka, M. Spatially-partitioned environmental representation and planning architecture for on-road autonomous driving. In *Proceedings of the 2017 IEEE Intelligent Vehicles Symposium* 632–639 (IEEE, 2017).
50. Mohy-ud-Din, H. & Muhammad, A. Detecting narrow passages in configuration spaces via spectra of probabilistic roadmaps. In *Proceedings of the 2010 ACM Symposium on Applied Computing* 1294–1298 (ACM, 2010).
51. Do, Q. H., Mita, S. & Yoneda, K. Narrow passage path planning using fast marching method and support vector machine. In *Proceedings of the 2014 IEEE Intelligent Vehicles Symposium* 630–635 (IEEE, 2014).
52. Bender, P., Taş, Ö. S., Ziegler, J. & Stiller, C. The combinatorial aspect of motion planning: maneuver variants in structured environments. In *Proceedings of the 2015 IEEE Intelligent Vehicles Symposium* 1386–1392 (IEEE, 2015).
53. Archer, J. & Vogel, K. *The Traffic Safety Problems in Urban Areas. Technical Report* (KTH Stockholm, 2000).
54. Shalev-Shwartz, S., Shammah, S. & Shashua, A. On a formal model of safe and scalable self-driving cars. Preprint at https://arxiv.org/pdf/1708.06374.pdf (2018).
55. Liebenwein, L. et al. Compositional and contract-based verification for autonomous driving on road networks. In *Robotics Research*, *Springer Proceedings in Advanced Robotics* Vol. 10, 163–181 (Springer, 2020).
56. Trautman, P. & Krause, A. Unfreezing the robot: navigation in dense, interacting crowds. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* 797–803 (IEEE, 2010).
57. Menéndez-Romero, C., Winkler, F., Dornhege, C. & Burgard, W. Maneuver planning for highly automated vehicles. In *Proceedings of the 2017 IEEE Intelligent Vehicles Symposium* 1458–1464 (IEEE, 2017).
58. Althoff, M. & Magdici, S. Set-based prediction of traffic participants on arbitrary road networks. *IEEE Trans. Intell. Vehicles* **1**, 187–202 (2016).
59. Koschi, M. & Althoff, M. SPOT: a tool for set-based prediction of traffic participants. In *Proceedings of the 2017 IEEE Intelligent Vehicles Symposium* 1686–1693 (IEEE, 2017).
60. Koschi, M., Pek, C., Beikirch, M. & Althoff, M. Set-based prediction of pedestrians in urban environments considering formalized traffic rules. In *Proceedings of the 21st International Conference on Intelligent Transportation Systems* 2704–2711 (IEEE, 2018).
61. Pek, C. & Althoff, M. Computationally efficient fail-safe trajectory planning for self-driving vehicles using convex optimization. In *Proceedings of the 2018 IEEE International Conference on Intelligent Transportation Systems* 1447–1454 (IEEE, 2018).
62. Manzinger, S., Pek, C. & Althoff, M. Using reachable sets for trajectory planning of automated vehicles. *IEEE Trans. Intell. Vehicles* https://doi.org/10.1109/TIV.2020.3017342 (2020).
63. Paden, B., Čáp, M., Yong, S. Z., Yershov, D. & Frazzoli, E. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Trans. Intell. Vehicles* **1**, 33–55 (2016).
64. González, D., Pérez, J., Milanés, V. & Nashashibi, F. A review of motion planning techniques for automated vehicles. *IEEE Trans. Intell. Transportation Syst.* **17**, 1135–1145 (2016).
65. Magdici, S., Ye, Z. & Althoff, M. Determining the maximum time horizon for vehicles to safely follow a trajectory. In *Proceedings of the 20th International Conference on Intelligent Transportation Systems* 1893–1899 (IEEE, 2017).
66. Héry, E., Masi, S., Xu, P. & Bonnifait, P. Map-based curvilinear coordinates for autonomous vehicles. In *Proceedings of the 20th International Conference on Intelligent Transportation Systems* 1–7 (IEEE, 2017).
67. Schürmann, B. et al. Ensuring drivability of planned motions using formal methods. In *Proceedings of the 20th International Conference on Intelligent Transportation Systems* 1661–1668 (IEEE, 2017).

Acknowledgements
We thank S. Kaster for his support in implementing the prediction and S. Steyer for providing the object detection and tracking algorithms. We also thank C. Schürmann for the voice-overs in the Supplementary Videos. This work was partially supported by the BMW Group within the CAR@TUM project, the German Federal Ministry of Economics and Technology through the research initiative Ko-HAF, and the German Research Foundation (DFG) under grants AL 1185/4-2 and AL 1185/3-2.

Author contributions
C.P., S.M. and M.K. developed the verification technique during replanning. M.K. developed the concept and algorithms for the set-based prediction. C.P. and S.M. developed the concept and algorithms for the drivable area computation, driving corridor identification and fail-safe trajectory planning. M.A. developed the main concept of online verification by integrating set-based prediction and fail-safe trajectory generation. He also developed the underlying algorithms for reachability analysis and led the research project. C.P., S.M. and M.K. designed and conducted the experiments and collected the data. The Article and the Supplementary Information were written by C.P., S.M. and M.K.

Competing interests
The authors declare no competing interests.

Additional information
Supplementary information is available for this paper at https://doi.org/10.1038/s42256-020-0225-y.

Correspondence and requests for materials should be addressed to C.P., S.M. or M.K.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

A.2 Using Reachable Sets for Trajectory Planning of Automated Vehicles [2]

Summary In critical traffic situations, the solution space for trajectory planning of autonomous vehicles is often small and cluttered, making it difficult to solve Problem 1 in time. We therefore propose to explore the non-convex solution space for trajectory planning using reachable sets. From the computed reachable sets, we extract driving corridors that represent spatio-temporal constraints. The obtained driving corridors guide the trajectory planner, which is based on continuous optimization, through the complex solution space. To obtain a convex optimization problem, we convexify the inherently non-convex optimization problem by separating the longitudinal and lateral dynamics of the autonomous vehicle. The collision avoidance constraints obtained from the driving corridor are represented as admissible position intervals in the longitudinal and lateral directions. The computational effort of the trajectory planner is low, since convex optimization problems can be solved efficiently to global optimality.

The combination of reachability analysis and convex optimization makes it possible to identify driving corridors in arbitrary traffic situations. The driving corridors can be restricted so that desired sets of goal states for the autonomous vehicle are reached, which facilitates tactical decision making, e.g., if the vehicle is supposed to reach a highway exit. Our method is particularly suitable for dealing with cluttered traffic situations, since the computation times typically decrease with increasing criticality of the traffic scene, i.e., the solution space for trajectory planning is shrinking.

We have successfully evaluated the drivability of the trajectories planned by our novel motion planning method in real vehicle tests. Moreover, we have compared our method with popular motion planning methods based on sampling and mixed-integer quadratic programming. The experiments reveal that our method is able to detect narrow passageways in the solution space for trajectory planning without tedious parameter tuning. Furthermore, our method has reliable runtimes in our experiments. We further demonstrate the general applicability of our method in a complex scenario in which the autonomous vehicle has to maneuver through dense traffic. In our last experiment, we analyze the runtime behavior of our method as a function of the size of the available solution space for trajectory planning. Our findings confirm that the runtime of our novel motion planning method has the tendency to decrease given that the solution space for trajectory planning shrinks.

Author Contributions M. A. initiated the idea of motion planning within reachable sets. **S. M.** predominantly developed the concept and algorithms with the support of C. P. **S. M.** predominantly wrote the article with the support of C. P. **S. M.** and C. P. designed, conducted, and evaluated the experiments. M. A. led the research project, provided feedback, and helped improving the manuscript.

Journal article The author's version of the article is reprinted in this thesis. The final version of record is available at `http://dx.doi.org/10.1109/TIV.2020.3017342`.

Attachments The video attachment of this publication is available at `https://ieeexplore.ieee.org/document/9170864/media#media`.

Using Reachable Sets for Trajectory Planning of Automated Vehicles

Stefanie Manzinger, Christian Pek, and Matthias Althoff

***Abstract*—The computational effort of trajectory planning for automated vehicles often increases with the complexity of the traffic situation. This is particularly problematic in safety-critical situations, in which the vehicle must react in a timely manner. We present a novel motion planning approach for automated vehicles, which combines set-based reachability analysis with convex optimization to address this issue. This combination makes it possible to find driving maneuvers even in small and convoluted solution spaces. In contrast to existing work, the computation time of our approach typically decreases, the more complex situations become. We demonstrate the benefits of our motion planner in scenarios from the CommonRoad benchmark suite and validate the approach on a real test vehicle.**

I. Introduction

EXISTING motion planning techniques for automated vehicles may still fail to obtain comfortable and safe motions in complex traffic scenarios with small and convoluted solution spaces (i.e., the portion of the state space enclosing feasible solutions is reduced by a large number of obstacles [1]). Yet, automated vehicles have to cope with arbitrarily complex scenarios in the real world. Why do planning techniques still struggle in complex scenarios?

The complexity mainly arises from the non-convexity of the motion planning problem: the presence of obstacles in the environment partitions the search space of the motion planning problem into different homotopy classes [1]–[3] (see Fig. 1). Homotopy classes describe "sets of trajectories that can be transformed into each other by gradual bending and stretching without colliding with obstacles" [4]. Figuratively speaking, planners have to decide when and how to pass obstacles, e.g., on the left or right side. The temporal orders of such tactical decisions can be represented by driving corridors (spatio-temporal constraints) and are a crucial element in generating collision avoidance constraints.

Given strict real-time constraints, many motion planning techniques encounter difficulties in determining suitable driving corridors for three reasons: a) scenarios in which α tactical decisions can be taken to avoid σ obstacles may already have up to α^{σ} possible driving corridors. This circumstance is particularly problematic in safety-critical situations, in which

Manuscript received Month Date, Year; revised Month Date, Year. This work was supported by the German Research Foundation (grants AL 1185/4-2 and AL 1185/3-2).

Stefanie Manzinger and Matthias Althoff are with the Department of Informatics, Technical University of Munich, D-85748 Garching, Germany (e-mail: stefanie.manzinger@in.tum.de, althoff@in.tum.de)

Christian Pek is with the Division of Robotics, Perception and Learning, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden (e-mail: pek2@kth.se); his contribution was carried out at the Department of Informatics, Technical University of Munich, D-85748 Garching, Germany.

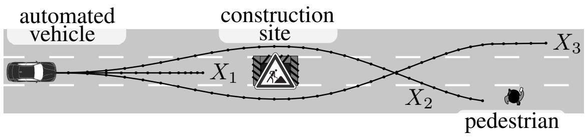


Fig. 1: Each trajectory $X_i, i \in \{1, 2, 3\}$, belongs to a different homotopy class. The automated vehicle needs to decide whether to pass the construction site (X_2 or X_3) or not (X_1). In the former case, the automated vehicle also has to decide on a passing side (left: X_2, right: X_3).

the automated vehicle must react in a timely manner to avoid a collision. b) Besides the sheer number of corridors, identifying which ones are drivable in reasonable time may often be intractable [5]. Deciding whether a driving corridor with a drivable trajectory exists can be reduced to a mixed-integer problem [6] and is therefore NP-complete [7]. c) Multiple driving corridors may exist that may not reach desired goal regions of the vehicle.

Set-based reachability analysis is particularly well suited for determining driving corridors in complex scenarios, since the number of set operations often decreases if the solution space becomes smaller [8]. The reachable set of an automated vehicle is the set of states the vehicle is able to reach over time starting from an initial set of states. Consequently, reachable sets make it possible to explore search spaces and homotopy classes efficiently in continuous space and to detect even narrow passages [9].

In this work, we show that by combining reachability analysis with an optimization-based trajectory planner, we are able to compute driving corridors and trajectories online in arbitrary scenarios. Our experiments demonstrate that our approach becomes faster with increased complexity of the motion planning problem. Below, we review existing planning techniques and present their shortcomings for the motion planning of automated vehicles in complex scenarios.

A. Literature Overview

1) Motion Planning Techniques: Various motion planning approaches are discussed in [10], [11]. We briefly review the most relevant publications for our work. Discretization-based planners, such as *rapidly exploring random trees* [12]–[14] or *state lattices* [15]–[18], connect partial motions toward a goal region in a kinematically feasible way. However, because the search space is discretized, solutions may not be found in small and convoluted solution spaces.

Continuous optimization techniques are applied to overcome discretization effects [19]. They minimize a cost function with respect to a set of state and input constraints [20]–[23]. Since many motion planning problems are non-convex, optimization problems may get stuck in local minima and may not be solvable in a computationally efficient way [24]. Convex optimization problems, however, have global convergence, for which efficient solving techniques exist [24]–[26]. Yet, convex planning problems can only be formulated for most traffic scenarios if the motion is separated into longitudinal and lateral components [27]–[30]. As a result, collision avoidance constraints can usually only be obtained if the driving corridor is already known [6].

Machine learning approaches have also been successfully applied to motion planning, e.g., [31]–[34]; however, these techniques are not yet suitable in safety-critical situations, since they are difficult to verify [35].

2) Driving Corridor Techniques: The extraction of driving corridors has been studied for some time. Naive approaches address the problem of determining driving corridors through sampling [36]–[39], combinatorial enumerations [5], [40], or support vector machines [41], [42]. However, sampling-based approaches usually struggle to detect narrow passages due to discretization effects, and enumerating all possible sequences of tactical decisions is not feasible under hard real-time constraints for complex situations. Pre-defining rules for tactical decision-making, e.g., using ontologies [43]–[45] or state-machines [46]–[49], may fail in unforeseen traffic situations. Some approaches reduce the complexity of maneuver planning by considering only a discrete set of admissible actions [50]–[53], but this diminishes the expressiveness of the planner.

Reachable sets have already been used to obtain driving corridors and to determine the non-existence of maneuvers [9], [54], [55]. The approaches in [56]–[58] combine reachability analysis with planning approaches to obtain collision-free motions. Nevertheless, they may not be applicable in arbitrary traffic situations, cannot constrain corridors to end in certain terminal states, or cannot extract high-level maneuvers with certain desired properties, such as comfort.

B. Contribution and Outline

This paper proposes a novel method for motion planning of automated vehicles by combining reachability analysis with convex optimization. Unlike previous work, our approach simultaneously

1) handles small and convoluted solution spaces. Our experiments reveal that the computational effort of our approach decreases with increased complexity of the scenario;
2) identifies collision-free driving corridors in arbitrary traffic situations using reachable sets;
3) plans optimal trajectories in candidate driving corridors with low computational effort; and
4) constrains driving corridors to end in certain terminal states, e.g., standstill in safe areas.

This paper is structured as follows: after presenting preliminaries in Sec. II, we introduce our framework in Sec. III. The computation of reachable sets is described in Sec. IV, followed by the determination of driving corridors and collision avoidance constraints in Sec. V-VI. We evaluate our concept in Sec. VII and finish with conclusions in Sec. VIII.

II. Preliminaries and Problem Statement

A. System Dynamics

We introduce different discrete-time linear systems to model the dynamics of the automated vehicle, subsequently denoted as ego vehicle. The linear models are a trade-off between the required computational time for the planning task and the accuracy compared to the real behavior. To formally verify planned trajectories despite control disturbances and model uncertainties, we refer to the approaches presented in [59], [60], which is not discussed in this work since we focus on motion planning. Additionally, in Sec. VII, we demonstrate by vehicle tests that planned trajectories are drivable despite the use of different models.

Let us introduce the subscript $\mathrm{sys} \in \{\mathrm{lon}, \mathrm{lat}, \mathcal{R}\}$ to distinguish the discrete-time linear systems for the longitudinal and lateral trajectory planning, and reachability analysis:

$$x_{\mathrm{sys},k+1} = A_{\mathrm{sys},k} x_{\mathrm{sys},k} + B_{\mathrm{sys},k} u_{\mathrm{sys},k}, \tag{1}$$

where $x_{\mathrm{sys},k} \in \mathbb{R}^{n_{\mathrm{sys},x}}$ is the state, $u_{\mathrm{sys},k} \in \mathbb{R}^{n_{\mathrm{sys},u}}$ is the input, and $k \in \mathbb{N}_0$ is the discrete time step corresponding to the time $t_k = k\Delta t$, where $\Delta t \in \mathbb{R}^+$ is the time increment. Without loss of generality, the initial time step is $k_0 = 0$ and the final time step is k_f. $A_{\mathrm{sys},k} \in \mathbb{R}^{n_{\mathrm{sys},x} \times n_{\mathrm{sys},x}}$ is the system matrix and $B_{\mathrm{sys},k} \in \mathbb{R}^{n_{\mathrm{sys},x} \times n_{\mathrm{sys},u}}$ is the input matrix. Systems in the form of (1) are subject to the convex sets of admissible states $\mathcal{X}_{\mathrm{sys},k} \subset \mathbb{R}^{n_{\mathrm{sys},x}}$ and admissible control inputs $\mathcal{U}_{\mathrm{sys},k} \subset \mathbb{R}^{n_{\mathrm{sys},u}}$, each at time step k. We use X_{sys} and U_{sys} to denote a possible state and input trajectory. Subsequently, we assume that the chosen reference point for describing the systems (1) coincides for all $\mathrm{sys} \in \{\mathrm{lon}, \mathrm{lat}, \mathcal{R}\}$.

B. Coordinate Systems and Reference Path

Let us introduce the global Cartesian coordinate frame as $F^{\mathtt{G}}$, the local curvilinear coordinate frame as $F^{\mathtt{L}}$, and the vehicle-fixed coordinate frame as $F^{\mathtt{V}}$, as shown in Fig. 2. The frame $F^{\mathtt{L}}$ is aligned with a given reference path $\Gamma : \mathbb{R} \to \mathbb{R}^2$, which is the centerline of a lane and can be provided by a route planning module, for example. In $F^{\mathtt{L}}$, a global position $(s_x, s_y)^T$ is expressed in terms of the arc length s_ζ and the orthogonal deviation s_η from $\Gamma(s_\zeta)$ (see Fig. 2). The orientation and curvature of $\Gamma(s_\zeta)$ are denoted by $\theta_\Gamma(s_\zeta)$ and $\kappa_\Gamma(s_\zeta)$, respectively. The transformation from $F^{\mathtt{L}}$ to $F^{\mathtt{G}}$ is denoted by $T^{\mathtt{G}}_{\mathtt{L}}(s_\zeta)$ (see [61]). The transformation from $F^{\mathtt{V}}$ to $F^{\mathtt{G}}$ is $T^{\mathtt{G}}_{\mathtt{V}}(\theta)$, where θ is the heading of the ego vehicle. We use a left-sided superscript $\{\mathtt{G}, \mathtt{L}, \mathtt{V}\}$ to indicate the reference coordinate system of a variable, e.g., ${}^{\mathtt{G}}v$ means velocity v of the ego vehicle in $F^{\mathtt{G}}$; for brevity, we omit the superscript if the coordinate system can be inferred from the context.

C. Reachability Analysis

As motivated in Sec. I, we use reachability analysis for identifying driving corridors. We therefore describe the dynamics of the ego vehicle with the discrete-time linear system $x_{\mathcal{R},k+1} = A_{\mathcal{R}} x_{\mathcal{R},k} + B_{\mathcal{R}} u_{\mathcal{R},k}$. Before introducing the one-step reachable set of the ego vehicle dynamics, we define the set $\mathcal{F}_k$ of forbidden states.

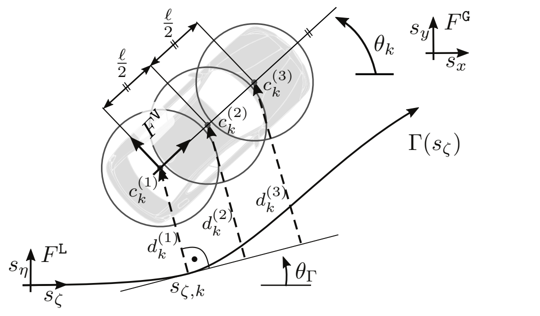


Fig. 2: The shape of the ego vehicle is approximated by three circles with center $c_k^{(i)}$, $i \in \{1,2,3\}$, to obtain collision avoidance constraints.

Definition 1 (Set of Forbidden States) *The occupied space of the ego vehicle in state $x_{\mathcal{R},k}$ is denoted by ${}^{\mathtt{G}}\mathcal{Q}(x_{\mathcal{R},k}) \subset \mathbb{R}^2$ and the set of occupied positions of all obstacles (e.g., other cars and pedestrians), as well as the space outside the road by ${}^{\mathtt{G}}\mathcal{O}_k$. The set of forbidden states at time step k is*

$$\mathcal{F}_k = \left\{x_{\mathcal{R},k} \in \mathcal{X}_{\mathcal{R},k} \mid {}^{\mathtt{G}}\mathcal{Q}(x_{\mathcal{R},k}) \cap {}^{\mathtt{G}}\mathcal{O}_k \neq \emptyset\right\}.$$

Definition 2 (One-step Reachable Set) *The initial reachable set is $\mathcal{R}^{\mathsf{e}}_{k_0} = \mathcal{X}_{\mathcal{R},k_0}$, where $\mathcal{X}_{\mathcal{R},k_0}$ is the set of initial states of the ego vehicle including measurement uncertainties. The one-step reachable set $\mathcal{R}^{\mathsf{e}}_{k+1}$ is then defined as the set of all states that can be reached from an initial set of states $\mathcal{R}^{\mathsf{e}}_k \subseteq \mathcal{X}_{\mathcal{R},k}$ within one time step without intersecting $\mathcal{F}_{k+1}$:*

$$\mathcal{R}^{\mathsf{e}}_{k+1} = \Big\{x_{\mathcal{R},k+1} \in \mathcal{X}_{\mathcal{R},k+1} \,\Big|\, \exists x_{\mathcal{R},k} \in \mathcal{R}^{\mathsf{e}}_k,\ \exists u_{\mathcal{R},k} \in \mathcal{U}_{\mathcal{R},k}: \\ x_{\mathcal{R},k+1} = A_{\mathcal{R}} x_{\mathcal{R},k} + B_{\mathcal{R}} u_{\mathcal{R},k} \wedge x_{\mathcal{R},k+1} \notin \mathcal{F}_{k+1}\Big\}.$$

Before defining the drivable area representing the collision-free, reachable positions of the ego vehicle, let us introduce the projection operator $\mathrm{proj}_{\diamondsuit}(x)$.

Definition 3 (Projection) *The operator $\mathrm{proj}_{\diamondsuit}(x)$ maps the state $x \in \mathcal{X}$ to its elements $\diamondsuit$, e.g., $\mathrm{proj}_{(m_1,m_3)}(x) = (m_1,m_3)^T$ for $x = (m_1,m_2,m_3)^T$. Using the same notation, we project a set of states $\mathcal{X}$: $\mathrm{proj}_{\diamondsuit}(\mathcal{X}) = \{\mathrm{proj}_{\diamondsuit}(x) \mid x \in \mathcal{X}\}$.*

Definition 4 (Drivable Area) *We obtain the drivable area $\mathcal{D}^{\mathsf{e}}_k$ of the ego vehicle as the projection of its reachable set $\mathcal{R}^{\mathsf{e}}_k$ onto the position domain: $\mathcal{D}^{\mathsf{e}}_k = \mathrm{proj}_{(s_\zeta,s_\eta)}(\mathcal{R}^{\mathsf{e}}_k)$.*

The superscript e of $\mathcal{R}^{\mathsf{e}}_k$ and $\mathcal{D}^{\mathsf{e}}_k$ denotes the exact reachable set and drivable area, respectively. However, it is generally computationally intractable to compute the exact reachable set of a system [62]. For this reason, we aim to compute accurate approximations $\mathcal{R}_k \approx \mathcal{R}^{\mathsf{e}}_k$ and $\mathcal{D}_k \approx \mathcal{D}^{\mathsf{e}}_k$ (see Sec. IV).

D. Convex Trajectory Planning

For trajectory planning, we describe the kinematics of the ego vehicle along $\Gamma(s_\zeta)$ using the rear axis center as the reference point, e.g., as shown in [29], [30]. In principle, the reference point can be chosen differently; however, our choice has the advantage that the slip angle can be assumed to be zero at low speeds and generally neglected at high speeds [29]. To formulate our trajectory planning problems as convex optimization problems—which can be efficiently solved with global convergence [24]—we separate the longitudinal and lateral motion of the ego vehicle.

Following [29], we assume that the orientation θ of the ego vehicle is close to the reference path's orientation $\theta_\Gamma(s_\zeta)$, consequently, the trigonometric functions can be approximated as $\sin(\Delta) \approx \Delta$ and $\cos(\Delta) \approx 1$, and $s_\eta \kappa_\Gamma(s_\zeta) \ll 1$. In this way, we linearize the kinematics of the ego vehicle as in [29]:

$$\dot{s}_\zeta = v\,\frac{\cos(\theta - \theta_\Gamma(s_\zeta))}{1 - s_\eta \kappa_\Gamma(s_\zeta)} \approx v, \tag{2}$$

$$\dot{s}_\eta = v \sin(\theta - \theta_\Gamma(s_\zeta)) \approx v(\theta - \theta_\Gamma(s_\zeta)). \tag{3}$$

Using (2) and (3), we can derive discrete-time linear systems of structure (1), which is detailed in [29], [30] and Sec. VII-A.

To model collision avoidance for the full-dimensional vehicle, we over-approximate its shape by three circles with equal radius r similar to [29], [30], [63]. The centers $c_k^{(i)}$, $i \in \{1,2,3\}$, of the circles are selected such that $c_k^{(1)}$ and $c_k^{(3)}$ coincide with the rear and front axis centers, respectively (see Fig. 2). The distance between the centers of adjacent circles is $\ell/2$, where ℓ is the wheelbase. With this, we formally define the longitudinal and lateral trajectory planning problems.

Definition 5 (Longitudinal Trajectory Planning Problem) *The longitudinal trajectory planning problem is to minimize the convex cost function $J_{\mathrm{lon}}(x_{\mathrm{lon}}, u_{\mathrm{lon}})$ for all $k \in \{k_0,\dots,k_f\}$:*

$$\min_{u_{\mathrm{lon}}} \ J_{\mathrm{lon}}(x_{\mathrm{lon}}, u_{\mathrm{lon}}) \tag{4a}$$

$$\text{s.t.} \quad x_{\mathrm{lon},k+1} = A_{\mathrm{lon},k}\, x_{\mathrm{lon},k} + B_{\mathrm{lon},k}\, u_{\mathrm{lon},k}, \tag{4b}$$

$$u_{\mathrm{lon},k} \in \mathcal{U}_{\mathrm{lon},k}, \quad x_{\mathrm{lon},k} \in \mathcal{X}_{\mathrm{lon},k}, \tag{4c}$$

$$\underline{s}_{\zeta,k+1} \leq \ \mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k+1}) \ \leq \overline{s}_{\zeta,k+1}, \tag{4d}$$

where (4b) *describes the longitudinal dynamics of the ego vehicle subject to the convex constraints* (4c)-(4d)*. The linear constraint* (4d) *models collision avoidance, where $s_{\zeta,k} = \mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k})$ is the longitudinal position of ${}^{\mathtt{L}}c_k^{(1)}$ (see Fig. 2). The minimum and maximum bounds on $s_{\zeta,k}$ are denoted by $\underline{s}_{\zeta,k}$ and $\overline{s}_{\zeta,k}$, respectively.*

Definition 6 (Lateral Trajectory Planning Problem) *Given that the longitudinal trajectory X_{lon} is obtained a priori by solving* (4)*, the lateral trajectory planning problem is to minimize the convex cost function $J_{\mathrm{lat}}(x_{\mathrm{lat}}, u_{\mathrm{lat}})$ for all $k \in \{k_0,\dots,k_f\}$:*

$$\min_{u_{\mathrm{lat}}} \ J_{\mathrm{lat}}(x_{\mathrm{lat}}, u_{\mathrm{lat}}) \tag{5a}$$

$$\text{s.t.} \quad x_{\mathrm{lat},k+1} = A_{\mathrm{lat},k}\, x_{\mathrm{lat},k} + B_{\mathrm{lat},k}\, u_{\mathrm{lat},k}, \tag{5b}$$

$$u_{\mathrm{lat},k} \in \mathcal{U}_{\mathrm{lat},k}, \quad x_{\mathrm{lat},k} \in \mathcal{X}_{\mathrm{lat},k}, \tag{5c}$$

$$\forall i \in \{1,2,3\}: \ \underline{d}^{(i)}_{k+1} \leq d^{(i)}_{k+1}(x_{\mathrm{lat},k+1}, x_{\mathrm{lon},k+1}) \leq \overline{d}^{(i)}_{k+1}, \tag{5d}$$

where (5b) *describes the lateral dynamics of the ego vehicle subject to the convex constraints* (5c)-(5d)*, and $x_{\mathrm{lon},k} \in X_{\mathrm{lon}}$. To model lateral collision avoidance, we introduce the lateral distance $d_k^{(i)}(x_{\mathrm{lat},k}, x_{\mathrm{lon},k})$ of the i-th circle's center to $\Gamma(s_{\zeta,k})$ at time step k (see Fig. 2) and restrict the minimum and maximum deviations $\underline{d}_k^{(i)}$ and $\overline{d}_k^{(i)}$ from $\Gamma(s_{\zeta,k})$, respectively.*

E. Problem Statement

The drivable area $\mathcal{D}_k$ is used to explore the collision-free solution space for trajectory planning. However, the drivable area is generally non-convex and may be disconnected due to the presence of obstacles (see Fig. 3a), which renders the direct usage of the drivable area unsuitable to obtain bounds for the convex collision avoidance constraints (4d) and (5d). To overcome this issue, we determine driving corridors that are subsets of the drivable area, particularly, a driving corridor for the longitudinal and lateral trajectory planning problems referred to as longitudinal and lateral driving corridor, respectively. To this end, we exploit that the projection of a connected set $\mathcal{C} \subset \mathbb{R}^2$ onto either the longitudinal or lateral position domain yields an interval from which we can obtain the bounds in (4d) and (5d).

Definition 7 (Connected Set [64]) *A set $\mathcal{C} \subset \mathbb{R}^2$ is connected if there do not exist open sets $\mathcal{W}_1, \mathcal{W}_2 \subseteq \mathbb{R}^2$ such that $\mathcal{W}_1 \cup \mathcal{W}_2$ contains $\mathcal{C}$ and with $\mathcal{C} \cap \mathcal{W}_1$ and $\mathcal{C} \cap \mathcal{W}_2$ disjoint and non-empty.*

A longitudinal driving corridor is composed of connected sets (see Fig. 3b).

Definition 8 (Longitudinal Driving Corridor) *A longitudinal driving corridor C_{lon} is a sequence of connected sets $\mathcal{C}_k \subseteq \mathcal{D}_k$ over time steps $k \in \{k_0, \ldots, k_f\}$. We refer to the longitudinal driving corridor at time step k as $C_{\mathrm{lon},k}$.*

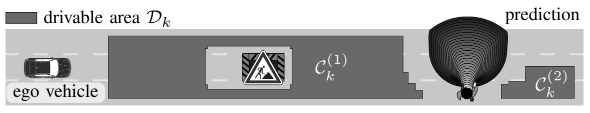
A lateral driving corridor is also composed of connected sets. However, a connected set may have holes, e.g., $\mathcal{C}_k^{(1)}$ in Fig. 3b; thus, the passing sides for obstacles may be ambiguous. Therefore, we demand that lateral driving corridors must provide a single passing side for each obstacle at the longitudinal positions $\mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k})$ of the longitudinal trajectory X_{lon} (see Fig. 3d).

Definition 9 (Lateral Driving Corridor) *A lateral driving corridor C_{lat} is a sequence of connected sets $\mathcal{C}_k \subseteq \mathcal{D}_k$ over time steps $k \in \{k_0, \ldots, k_f\}$. For each $\mathcal{C}_k$, it must hold that $\{s_{\eta,k} \,|\, (s_{\zeta,k}, s_{\eta,k})^T \in \mathcal{C}_k, s_{\zeta,k} = \mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k}), x_{\mathrm{lon},k} \in X_{\mathrm{lon}}\}$ is a non-empty interval (unique passing side at $\mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k})$, see Fig. 3d). We refer to the lateral driving corridor at time step k as $C_{\mathrm{lat},k}$.*

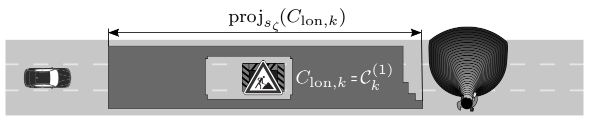
The goal of this work is to present an efficient algorithm to extract longitudinal and lateral driving corridors according to Def. 8 and 9 from the drivable area. Furthermore, we elaborate on the extraction of collision avoidance constraints for the full-dimensional vehicle using the proposed driving corridors.

III. Framework

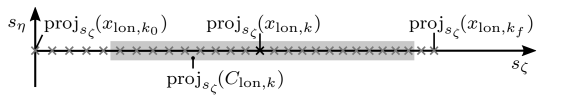
Let us introduce our motion planning approach summarized in Alg. 1 using the scenario in Fig. 3, in which the ego vehicle approaches a construction site and a pedestrian. Our motion planning approach is integrated between the behavioral layer and the control layer of the ego vehicle, e.g., see planning framework in [11]. We assume that our method receives as input the environment model including the admissible lanes leading to the specified target, a desired reference path through the road network, and all safety-relevant traffic participants and their predicted future behavior. Our algorithm outputs a reference trajectory for the ego vehicle that is passed to a vehicle controller.



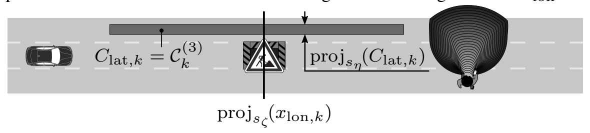
(a) Computation of the drivable area for consecutive time steps k. We depict the drivable area $\mathcal{D}_k$ at time step k.


$\mathrm{proj}_{s_\zeta}(C_{\mathrm{lon},k})$
$C_{\mathrm{lon},k} = \mathcal{C}_k^{(1)}$

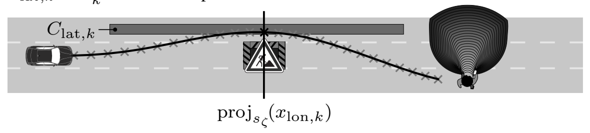
(b) Identification of the longitudinal driving corridors C_{lon} as a time series of connected sets. We depict the longitudinal driving corridor $C_{\mathrm{lon},k} = \mathcal{C}_k^{(1)}$ at time step k.


s_η
$\mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k_0})$
$\mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k})$
$\mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k_f})$
s_ζ
$\mathrm{proj}_{s_\zeta}(C_{\mathrm{lon},k})$

(c) Optimization of the longitudinal trajectory X_{lon} subject to the longitudinal position constraints obtained from the longitudinal driving corridor C_{lon}.


$C_{\mathrm{lat},k} = \mathcal{C}_k^{(3)}$
$\mathrm{proj}_{s_\eta}(C_{\mathrm{lat},k})$
$\mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k})$

(d) Identification of the lateral driving corridors C_{lat} as a time series of connected sets with unique passing side. We depict the lateral driving corridor $C_{\mathrm{lat},k} = \mathcal{C}_k^{(3)}$ at time step k.



(e) Optimization of the lateral trajectory X_{lat} subject to the lateral position constraints obtained from the lateral driving corridor C_{lat}.

Fig. 3: Motion planning with reachable sets.

As a first step, we compute the one-step reachable sets of the ego vehicle for consecutive time steps (see Alg. 1, line 2) and store them in a data structure $\mathcal{G}_{\mathcal{R}}$ (see Sec. IV). Since our trajectory planner requires position constraints to determine drivable trajectories, we project the reachable set onto the position domain to obtain the drivable area (see Fig. 3a).

We continue with identifying possible longitudinal driving corridors within the drivable area according to Def. 8 (see Alg. 1, line 3). The drivable area $\mathcal{D}_k$ can be disconnected, e.g., $\mathcal{C}_k^{(1)}$ is not connected with $\mathcal{C}_k^{(2)}$ in Fig. 3a, since its computation requires the removal of colliding states (see Def. 2 and 4). Because of this, there may be multiple longitudinal driving corridors (stored in a data structure $\mathcal{G}_{\mathcal{C},\mathrm{lon}}$, see Sec. V). We select a suitable one for trajectory optimization by ranking the longitudinal driving corridors according to a user-defined criterion (see Alg. 1, line 4). The highest-ranked driving corridor C_{lon}, e.g., $C_{\mathrm{lon},k}$ in Fig. 3b, is selected for planning the longitudinal trajectory X_{lon} as proposed in Sec. II-D (see Alg. 1, line 5). If the optimization problem is infeasible, we select the next available longitudinal driving corridor. The

Algorithm 1 ConvexTrajectoryPlanning

```
Input: set of initial states $\mathcal{X}_{\mathcal{R},k_0}$, set of obstacles $\mathcal{O}_k$ for $k_0$ to $k_f$
Output: longitudinal and lateral trajectories $X_\mathrm{lon}$ and $X_\mathrm{lat}$
 1: $X_\mathrm{lon},\ X_\mathrm{lat} \leftarrow \emptyset$
 2: $\mathcal{G}_\mathcal{R} \leftarrow$ computeReachableSet($\mathcal{X}_{\mathcal{R},k_0}, \mathcal{O}_k$)
 3: $\mathcal{G}_{\mathcal{C},\mathrm{lon}} \leftarrow$ IdentifyCorridors($\mathcal{G}_\mathcal{R}$)
 4: for all $C_\mathrm{lon}$ in $\mathcal{G}_{\mathcal{C},\mathrm{lon}}$.assessCorridor( ) do
 5:     $X_\mathrm{lon} \leftarrow$ LongitudinalOptimization($C_\mathrm{lon}$)
 6:     if $X_\mathrm{lon} \neq \emptyset$ then
 7:         $\mathcal{G}_{\mathcal{C},\mathrm{lat}} \leftarrow$ IdentifyCorridors($\mathcal{G}_\mathcal{R}$, $X_\mathrm{lon}$, $C_\mathrm{lon}$)
 8:         for all $C_\mathrm{lat}$ in $\mathcal{G}_{\mathcal{C},\mathrm{lat}}$.assessCorridor( ) do
 9:             $X_\mathrm{lat} \leftarrow$ LateralOptimization($C_\mathrm{lat}$, $X_\mathrm{lon}$, $C_\mathrm{lon}$)
10:             if $X_\mathrm{lat} \neq \emptyset$ then
11:                 return $X_\mathrm{lon}, X_\mathrm{lat}$
12:             end if
13:         end for
14:     end if
15: end for
```

longitudinal trajectory X_lon (see Fig. 3c) is used to identify lateral driving corridors (see Alg. 1, line 7).

Analogous to longitudinal planning, multiple lateral driving corridors may exist (stored in a data structure $\mathcal{G}_{\mathcal{C},\mathrm{lat}}$, see Sec. V). We also rank them according to a user-defined criterion (see Alg. 1, line 8). The highest-ranked lateral driving corridor C_lat, e.g., $C_{\mathrm{lat},k}$ in Fig. 3d, is used to optimize the lateral trajectory X_lat as proposed in Sec. II-D (Alg. 1, line 9). If the optimization problem is infeasible, we use the next available driving corridor C_lat to obtain X_lat. If the optimization is successful, the final trajectory is returned (see Fig. 3e and Alg. 1, line 11). Trajectories can be planned for each available driving corridor as long as time permits. Thus, it is possible to obtain several maneuver options to choose from. In the rare event that no drivable trajectories are found, we always keep a fail-safe trajectory available [30], [65].

IV. Reachable Set Computation

After presenting the vehicle model for reachability analysis in Sec. IV-A, we introduce the representation of reachable sets (see Sec. IV-B) and our proposed algorithm (see Sec. IV-C) inspired by [9].

A. Vehicle Model for Reachability Analysis

We deliberately select a simple but realistic dynamic model for the reachable set computation to reduce computational complexity. The dynamics of the ego vehicle are described by two second-order integrator models in the curvilinear coordinate frame F^L with ${}^\mathrm{L}c_k^{(1)}$ as the reference point (see Fig. 2). The state $x_\mathcal{R} = (s_\zeta, v_\zeta, s_\eta, v_\eta)^T$ is composed of the position s and velocity v in the longitudinal ζ-direction and the lateral η-direction. The input $u_\mathcal{R} = (a_\zeta, a_\eta)^T$ is given by the acceleration a in ζ- and η-directions. Adding bounds on the accelerations and velocities, the dynamics are

$$x_{\mathcal{R},k+1} = \begin{pmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{pmatrix} x_{\mathcal{R},k} + \begin{pmatrix} \frac{1}{2}\Delta t^2 & 0 \\ \Delta t & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ 0 & \Delta t \end{pmatrix} u_{\mathcal{R},k}, \quad (6a)$$

$$\underline{v}_\zeta \leq v_{\zeta,k} \leq \overline{v}_\zeta, \quad \underline{v}_\eta \leq v_{\eta,k} \leq \overline{v}_\eta, \quad (6b)$$

$$\underline{a}_\zeta \leq a_{\zeta,k} \leq \overline{a}_\zeta, \quad \underline{a}_\eta \leq a_{\eta,k} \leq \overline{a}_\eta. \quad (6c)$$

The formulation of the dynamics in F^L makes it possible to consider speed limits and to avoid driving backwards. This entails that model (6a) deviates increasingly from a real vehicle for larger curvatures $\kappa_\Gamma(s_\zeta)$, e.g., the ego vehicle would be able to make a turn with an arbitrarily high velocity. However, we compensate for this by setting appropriate constraints (6b)-(6c), e.g., we use a conservative parametrization of the admissible accelerations and velocities.

B. Reachable Set Representation

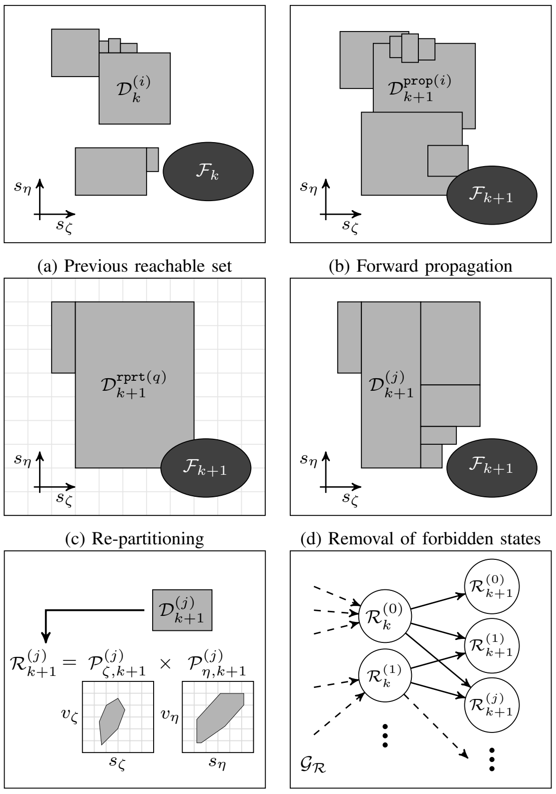
To improve computational efficiency, the reachable set (see Def. 2) of (6) is approximated by the union $\mathcal{R}_k$ of base sets $\mathcal{R}_k^{(i)} = \mathcal{P}_{\zeta,k}^{(i)} \times \mathcal{P}_{\eta,k}^{(i)}$, $i \in \mathbb{N}_0$, which are the Cartesian products of two convex polytopes $\mathcal{P}_{\zeta,k}^{(i)}$ and $\mathcal{P}_{\eta,k}^{(i)}$ in the (s_ζ, v_ζ) and (s_η, v_η) plane (see Fig. 4e), respectively [9]. Hence, each base set represents pairs of reachable positions and velocities. We select polytopes for the representation of the reachable set, because the required set operations can be efficiently performed on polytopes. The projection of sets $\mathcal{R}_k^{(i)}$ onto the position domain yields axis-aligned rectangles $\mathcal{D}_k^{(i)}$ (see Fig. 4a) whose union represents the drivable area $\mathcal{D}_k$ (see Def. 4).

C. Algorithm

The initial base set $\mathcal{R}_{k_0}^{(0)}$ encloses the set of initial states $\mathcal{X}_{\mathcal{R},k_0}$ of the ego vehicle including measurement uncertainties, such that $\mathcal{X}_{\mathcal{R},k_0} \subseteq \mathcal{R}_{k_0}^{(0)}$. The reachable set $\mathcal{R}_k$ of consecutive time steps is computed as explained below and illustrated in Fig. 4. To simplify the notation, we also denote the collection of base sets $\mathcal{R}_k^{(i)}$ with $\mathcal{R}_k$ (not only the union of base sets), i.e., $\mathcal{R}_k = \{\mathcal{R}_k^{(0)}, \mathcal{R}_k^{(1)}, \ldots\}$. Similarly, we simplify the notation for the drivable area $\mathcal{D}_k$.

1) Forward Propagation: The base sets $\mathcal{R}_k^{(i)} \in \mathcal{R}_k$ of time step k (see Fig. 4a) are propagated forward in time according to the system model (6) considering all admissible inputs. The forward propagation is computed similarly to [9] and a detailed description can be found in the Appendix A. We denote the propagated reachable set with $\mathcal{R}_{k+1}^\mathtt{prop}$, and refer to the propagated drivable area as $\mathcal{D}_{k+1}^\mathtt{prop}$, see Fig. 4b.

2) Re-Partitioning: With the aim to reduce the number of axis-aligned rectangles, the propagated drivable area $\mathcal{D}_{k+1}^\mathtt{prop}$ is merged and re-partitioned into a set $\mathcal{D}_{k+1}^\mathtt{rprt}$ of new axis-aligned rectangles with disjoint interiors using sweep line algorithms [66] (see Fig 4c). To avoid generating a large number of new axis-aligned rectangles with only slightly displaced edges, we enlarge each propagated set $\mathcal{D}_{k+1}^{\mathtt{prop}(i)}$ to a grid prior to merging and re-partitioning as presented in [9] (see Fig 4c).


$\mathcal{D}_k^{(i)}$
$\mathcal{F}_k$
s_η
s_ζ
$\mathcal{D}_{k+1}^{\mathtt{prop}(i)}$
$\mathcal{F}_{k+1}$
$\mathcal{D}_{k+1}^{\mathtt{rprt}(q)}$
$\mathcal{D}_{k+1}^{(j)}$
$\mathcal{R}_{k+1}^{(j)} = \mathcal{P}_{\zeta,k+1}^{(j)} \times \mathcal{P}_{\eta,k+1}^{(j)}$
v_ζ
v_η
$\mathcal{R}_k^{(0)}$
$\mathcal{R}_k^{(1)}$
$\mathcal{R}_{k+1}^{(0)}$
$\mathcal{R}_{k+1}^{(1)}$
$\mathcal{R}_{k+1}^{(j)}$
$\mathcal{G}_\mathcal{R}$

(a) Previous reachable set (b) Forward propagation (c) Re-partitioning (d) Removal of forbidden states (e) Creation of base sets (f) Update of reachability graph

Fig. 4: Steps in the reachable set computation.

3) Removal of Forbidden States: The forbidden positions are removed from $\mathcal{D}_{k+1}^{\mathtt{rprt}}$ to obtain only collision-free configurations of the ego vehicle as shown in Fig. 4d. Since $\mathcal{D}_{k+1}^{\mathtt{rprt}} \setminus \mathrm{proj}_{(s_\zeta,s_\eta)}(\mathcal{F}_{k+1})$ cannot be represented by axis-aligned rectangles in general, we under-approximate the result by $\mathcal{D}_{k+1}$. The under-approximation ensures that the remaining drivable area $\mathcal{D}_{k+1}$ is collision-free. A detailed description of this computation step is provided in Sec. IV-D.

4) Creation of Base Sets: The reachable set $\mathcal{R}_{k+1}$ is obtained by determining the reachable velocities for the collision-free drivable area $\mathcal{D}_{k+1}$ (see Fig. 4e). From the propagated base sets $\mathcal{R}_{k+1}^{\mathtt{prop}(i)}$, we can determine the velocities for which the ego vehicle reaches a set of positions $\mathcal{D}_{k+1}^{(j)}$. After introducing the set $\mathrm{overlap}(\mathcal{D}_{k+1}^{(j)}) = \{i \in \mathbb{N}_0 \,|\, \exists \mathcal{R}_{k+1}^{\mathtt{prop}(i)} \in \mathcal{R}_{k+1}^{\mathtt{prop}} : \mathcal{D}_{k+1}^{(j)} \cap \mathrm{proj}_{(s_\zeta,s_\eta)}(\mathcal{R}_{k+1}^{\mathtt{prop}(i)}) \neq \emptyset\}$, the new polytopes $\mathcal{P}_{\zeta,k+1}^{(j)}$ are

$$\mathcal{P}_{\zeta,k+1}^{(j)} = \mathrm{convexhull}\left(\bigcup_{i \in \mathrm{overlap}(\mathcal{D}_{k+1}^{(j)})} \hat{\mathcal{P}}_{\zeta,k+1}^{(i)} \right),$$

with

$$\hat{\mathcal{P}}_{\zeta,k+1}^{(i)} = \mathcal{P}_{\zeta,k+1}^{\mathtt{prop}(i)} \cap \left\{ \begin{pmatrix} s_\zeta \\ v_\zeta \end{pmatrix} \in \mathbb{R}^2 \middle| s_\zeta \geq \inf\left(\mathrm{proj}_{s_\zeta}(\mathcal{D}_{k+1}^{(j)})\right), \right.$$
$$\left. s_\zeta \leq \sup\left(\mathrm{proj}_{s_\zeta}(\mathcal{D}_{k+1}^{(j)})\right) \right\}.$$

$\mathcal{P}_{\eta,k+1}^{(j)}$ is determined similarly. Note that we compute the convex hull to ensure that we obtain convex polytopes.

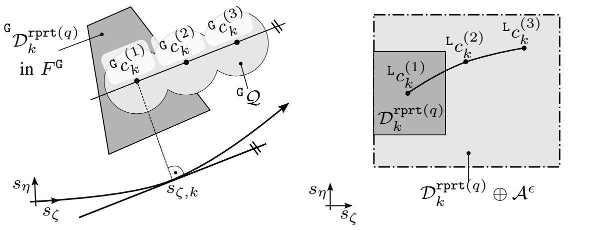
5) Update of Reachability Graph: The base sets $\mathcal{R}_{k+1}^{(j)}$ are stored in a directed graph $\mathcal{G}_\mathcal{R}$ to trace back the temporal and spatial sequence of reachable states (see Fig. 4f). In $\mathcal{G}_\mathcal{R}$, each base set $\mathcal{R}_{k+1}^{(j)}$ is assigned to exactly one node and an edge indicates that $\mathcal{R}_{k+1}^{(j)}$ is reachable from $\mathcal{R}_k^{(i)}$ for consecutive time steps (this also holds for $\mathcal{D}_k^{(i)}$ and $\mathcal{D}_{k+1}^{(j)}$, since $\mathcal{D}_k^{(i)} = \mathrm{proj}_{(s_\zeta,s_\eta)}(\mathcal{R}_k^{(i)})$). Note that a set $\mathcal{R}_k^{(i)}$ may reach several sets $\mathcal{R}_{k+1}^{(j)}$ in the next point in time.

D. Removal of Forbidden States

To remove the forbidden states (see Fig. 4d), we need to consider the occupancy ${}^{\mathtt{G}}\mathcal{Q}(x_{\mathcal{R},k})$ of the ego vehicle that is over-approximated by three circles with radius r (see Sec. II). Let us therefore introduce the Minkowski sum of two sets $\mathcal{W}_1$ and $\mathcal{W}_2$ as $\mathcal{W}_1 \oplus \mathcal{W}_2 = \{w_1 + w_2 \,|\, w_1 \in \mathcal{W}_1, w_2 \in \mathcal{W}_2\}$. By dilating the occupancies ${}^{\mathtt{G}}\mathcal{O}_k$ with a circle with radius r using the Minkowski sum, it is sufficient to test only ${}^{\mathtt{G}}c_k^{(i)}$, $i \in \{1,2,3\}$, for collisions [63]. To determine the center ${}^{\mathtt{G}}c_k^{(i)}$ of the i-th circle for ${}^{\mathtt{L}}c_k^{(1)} = (s_{\zeta,k}, s_{\eta,k})^T$, we assume that the heading θ_k of the ego vehicle is aligned with $\theta_\Gamma(s_{\zeta,k})$:

$${}^{\mathtt{G}}c_k^{(i)} = T_{\mathtt{L}}^{\mathtt{G}}(s_{\zeta,k}) \begin{pmatrix} s_{\zeta,k} \\ s_{\eta,k} \end{pmatrix} + T_{\mathtt{V}}^{\mathtt{G}}(\theta_\Gamma(s_{\zeta,k})) \begin{pmatrix} \frac{i-1}{2}\ell \\ 0 \end{pmatrix}. \quad (7)$$

Computing (7) for all $(s_{\zeta,k}, s_{\eta,k})^T \in \mathcal{D}_k^{\mathtt{rprt}}$ to determine the collision-free drivable area is intractable, since the transformation matrix $T_{\mathtt{V}}^{\mathtt{G}}(\theta_\Gamma(s_{\zeta,k}))$ continuously varies along the reference path $\Gamma(s_\zeta)$ with changing $s_{\zeta,k}$ (see Fig. 5a). To reduce computational complexity, we strictly over-approximate the reachable positions of centers ${}^{\mathtt{L}}c_k^{(i)}$ by enlarging each $\mathcal{D}_k^{\mathtt{rprt}(q)} \in \mathcal{D}_k^{\mathtt{rprt}}$ with a rectangular shape $\mathcal{A}^\epsilon$ (see Fig. 5b): $\mathcal{D}_k^{\mathtt{rprt}(q)} \oplus \mathcal{A}^\epsilon$. The size of $\mathcal{A}^\epsilon$ depends on the local curvature κ_Γ of the reference path and $\mathcal{D}_k^{\mathtt{rprt}(q)}$; however, we omit the


${}^{\mathtt{G}}\mathcal{D}_k^{\mathtt{rprt}(q)}$ in $F^{\mathtt{G}}$
${}^{\mathtt{G}}c_k^{(1)}$
${}^{\mathtt{G}}c_k^{(2)}$
${}^{\mathtt{G}}c_k^{(3)}$
${}^{\mathtt{G}}\mathcal{Q}$
s_η
s_ζ
$s_{\zeta,k}$
${}^{\mathtt{L}}c_k^{(1)}$
${}^{\mathtt{L}}c_k^{(2)}$
${}^{\mathtt{L}}c_k^{(3)}$
$\mathcal{D}_k^{\mathtt{rprt}(q)}$
$\mathcal{D}_k^{\mathtt{rprt}(q)} \oplus \mathcal{A}^\epsilon$

(a) Occupancy ${}^{\mathtt{G}}\mathcal{Q}$ of the ego vehicle for an arbitrary position ${}^{\mathtt{G}}c_k^{(1)}$ within ${}^{\mathtt{G}}\mathcal{D}_k^{\mathtt{rprt}(q)}$ given that $\theta_k = \theta_\Gamma(s_{\zeta,k})$. (b) Reachable positions of centers ${}^{\mathtt{L}}c_k^{(i)}$, $i \in \{1,2,3\}$, are enclosed by enlarging $\mathcal{D}_k^{\mathtt{rprt}(q)}$ with $\mathcal{A}^\epsilon$ in $F^{\mathtt{L}}$.

Fig. 5: Consideration of the shape of the ego vehicle in the reachable set computation.

dependency in the notation for brevity. A detailed explanation on the computation of $\mathcal{A}^{\epsilon}$ is given in the Appendix B. Using $\mathcal{A}^{\epsilon}$, the removal of forbidden states from $\mathcal{D}_k^{\mathtt{rprt}}$ can be either performed in the global coordinate frame $F^{\mathtt{G}}$ or in the curvilinear frame $F^{\mathtt{L}}$. As a result, we under-approximate $\mathcal{D}_k^{\mathtt{rprt}} \setminus \mathrm{proj}_{(s_\zeta, s_\eta)}(\mathcal{F}_k)$ with a new set of axis-aligned rectangles $\mathcal{D}_k$, such that for all $\mathcal{D}_k^{(j)} \in \mathcal{D}_k$: $\mathcal{D}_k^{(j)} \oplus \mathcal{A}^{\epsilon}$ is collision-free.

V. Identification of Driving Corridors

After introducing necessary preliminaries, we elaborate on the identification of longitudinal and lateral driving corridors.

A. Preliminaries

As stated in Def. 8-9, a driving corridor is a temporal sequence of connected sets in the position domain. We denote the n-th connected set at time step k by $\mathcal{C}_k^{(n)} = \{\mathcal{D}_k^{(i)}, \mathcal{D}_k^{(j)}, \dots\}$, $n \in \mathbb{N}$ (see Fig. 3a). To determine longitudinal and lateral driving corridors, we create corresponding graphs $\mathcal{G}_{\mathcal{C}}$, in which connected sets $\mathcal{C}_k^{(n)}$ are stored as a node denoted by $\mathtt{n}(\mathcal{C}_k^{(n)})$. An edge between nodes $\mathtt{n}(\mathcal{C}_k^{(n)})$ and $\mathtt{n}(\mathcal{C}_{k+1}^{(m)})$ in $\mathcal{G}_{\mathcal{C}}$ indicates that at least one set $\mathcal{D}_k^{(i)} \in \mathcal{C}_k^{(n)}$ reaches a set $\mathcal{D}_{k+1}^{(j)} \in \mathcal{C}_{k+1}^{(m)}$ within one time step so that the connected set $\mathcal{C}_k^{(n)}$ reaches $\mathcal{C}_{k+1}^{(m)}$. A path in $\mathcal{G}_{\mathcal{C}}$ from k_0 to k_f is a possible driving corridor.

B. Longitudinal Driving Corridors

Alg. 2 identifies longitudinal driving corridors backwards in time using the reachability graph $\mathcal{G}_{\mathcal{R}}$. Let us first explain the procedure, afterwards we apply it to the example in Fig. 6: we obtain the drivable area $\mathcal{D}_{k_f}$ of the final time step k_f (see Alg. 2, line 2) through the reachability graph $\mathcal{G}_{\mathcal{R}}$. As an option, we can exclude states of the reachable set beforehand to constrain driving corridors to end in a predefined set of terminal states (e.g., a standstill in safe areas). Then, we identify connected sets $\mathcal{C}_{k_f}^{(n)}$ within the drivable area $\mathcal{D}_{k_f}$ using a sweep line algorithm [67] (see Alg. 2, line 6). For each $\mathcal{C}_{k_f}^{(n)}$, we add a new node $\mathtt{n}(\mathcal{C}_{k_f}^{(n)})$ to $\mathcal{G}_{\mathcal{C}}$ (see Alg. 2, line 7) and determine the temporally preceding connected sets by calling the FindPaths function (see Alg. 2, line 8).

The FindPaths function updates graph $\mathcal{G}_{\mathcal{C}}$ by determining connected sets $\mathcal{C}_{k-1}^{(s)}$, $s \in \mathbb{N}$, reaching the provided set $\mathcal{C}_k^{(n)}$ within one time step (see Alg. 2, lines 11-22). To this end, we obtain the union of parents $\mathcal{D}_{k-1}^{\text{parents}} = \{\mathcal{D}_{k-1}^{(q)}, \mathcal{D}_{k-1}^{(p)}, \dots\}$ for all provided sets $\mathcal{D}_k^{(i)} \in \mathcal{C}_k^{(n)}$ through the reachability graph $\mathcal{G}_{\mathcal{R}}$ (see Alg. 2, line 12). Next, we also cluster the parent sets $\mathcal{D}_{k-1}^{\text{parents}}$ to connected sets $\mathcal{C}_{k-1}^{(s)}$ (see Alg. 2, line 16). For each connected parent set $\mathcal{C}_{k-1}^{(s)}$:

1) a new node $\mathtt{n}(\mathcal{C}_{k-1}^{(s)})$ is added to $\mathcal{G}_{\mathcal{C}}$ (see Alg. 2, line 17),
2) an edge from $\mathtt{n}(\mathcal{C}_{k-1}^{(s)})$ to $\mathtt{n}(\mathcal{C}_k^{(n)})$ is created to store the fact that $\mathcal{C}_{k-1}^{(s)}$ reaches $\mathcal{C}_k^{(n)}$ (see Alg. 2, line 18),
3) the FindPaths function is called (see Alg. 2, line 19).

The FindPaths function terminates as soon as all paths from the provided set $\mathcal{C}_{k_f}^{(n)}$ to the initial set $\mathcal{C}_{k_0}^{(1)}$ are found. One may also set a timeout or other criteria for early stopping, e.g., a maximum number of driving corridors.

Algorithm 2 IdentifyCorridors

Input: reachability graph $\mathcal{G}_{\mathcal{R}}$
Optional Input: X_{lon} and C_{lon} to obtain lateral driving corridors (otherwise longitudinal driving corridors are obtained).
Output: graph $\mathcal{G}_{\mathcal{C}}$ containing possible driving corridors

```
 1: $\mathcal{G}_{\mathcal{C}}$.INIT( )
 2: $\mathcal{D}_{k_f} \leftarrow \mathcal{G}_{\mathcal{R}}$.DRIVABLEAREA($k_f$)
 3: if $X_{\mathrm{lon}} \neq \emptyset$ then                ▷ only for lateral driving corridors
 4:     $\mathcal{D}_{k_f} \leftarrow$ EXCLUDESETS($\mathcal{D}_{k_f}$, $X_{\mathrm{lon}}$, $C_{\mathrm{lon}}$)
 5: end if
 6: for all $\mathcal{C}_{k_f}^{(n)}$ in CONNECTEDSETS($\mathcal{D}_{k_f}$) do
 7:     $\mathcal{G}_{\mathcal{C}}$.ADDNODE($\mathtt{n}(\mathcal{C}_{k_f}^{(n)})$)
 8:     $\mathcal{G}_{\mathcal{C}} \leftarrow$ FINDPATHS($\mathcal{G}_{\mathcal{R}}$, $\mathcal{G}_{\mathcal{C}}$, $\mathcal{C}_{k_f}^{(n)}$, $X_{\mathrm{lon}}$, $C_{\mathrm{lon}}$)
 9: end for
10: return $\mathcal{G}_{\mathcal{C}}$

11: function FINDPATHS($\mathcal{G}_{\mathcal{R}}$, $\mathcal{G}_{\mathcal{C}}$, $\mathcal{C}_k^{(n)}$, $X_{\mathrm{lon}}$, $C_{\mathrm{lon}}$)
12:     $\mathcal{D}_{k-1}^{\text{parents}} \leftarrow \mathcal{G}_{\mathcal{R}}$.GETPARENTS($\mathcal{C}_k^{(n)}$)
13:     if $X_{\mathrm{lon}} \neq \emptyset$ then            ▷ only for lateral driving corridors
14:         $\mathcal{D}_{k-1}^{\text{parents}} \leftarrow$ EXCLUDESETS($\mathcal{D}_{k-1}^{\text{parents}}$, $X_{\mathrm{lon}}$, $C_{\mathrm{lon}}$)
15:     end if
16:     for all $\mathcal{C}_{k-1}^{(s)}$ in CONNECTEDSETS($\mathcal{D}_{k-1}^{\text{parents}}$) do
17:         $\mathcal{G}_{\mathcal{C}}$.ADDNODE($\mathtt{n}(\mathcal{C}_{k-1}^{(s)})$)
18:         $\mathcal{G}_{\mathcal{C}}$.ADDEDGE($\mathtt{n}(\mathcal{C}_{k-1}^{(s)}), \mathtt{n}(\mathcal{C}_k^{(n)})$)
19:         $\mathcal{G}_{\mathcal{C}} \leftarrow$ FINDPATHS($\mathcal{G}_{\mathcal{R}}$, $\mathcal{G}_{\mathcal{C}}$, $\mathcal{C}_{k-1}^{(s)}$, $X_{\mathrm{lon}}$, $C_{\mathrm{lon}}$)
20:     end for
21:     return $\mathcal{G}_{\mathcal{C}}$
22: end function

23: function EXCLUDESETS($\mathcal{D}'_k$, $X_{\mathrm{lon}}$, $C_{\mathrm{lon}}$)
24:     $\mathcal{D}'_k \leftarrow \mathcal{D}'_k \cap C_{\mathrm{lon},k}$
25:     $\mathcal{D}'_k \leftarrow$ FILTER($\mathcal{D}'_k$, $X_{\mathrm{lon}}$.AT($k$))
26:     return $\mathcal{D}'_k$
27: end function
```

Example *At time step $k_f = 2$, the connected sets are $\mathcal{C}_2^{(1)}$, $\mathcal{C}_2^{(2)}$, and $\mathcal{C}_2^{(3)}$ (see Fig. 6b). Let us assume that* FindPaths *is invoked with $\mathcal{C}_2^{(2)}$. The obtained parents of $\mathcal{C}_2^{(2)}$ are $\mathcal{D}_1^{\text{parents}} = \{\mathcal{D}_1^{(0)}, \dots, \mathcal{D}_1^{(4)}\}$ (see Fig. 6a-6b) that are partitioned into two connected sets $\mathcal{C}_1^{(1)}$ and $\mathcal{C}_1^{(2)}$ (see Fig. 6b). Assume that we continue with $\mathcal{C}_1^{(1)}$: we first add the node $\mathtt{n}(\mathcal{C}_1^{(1)})$ to $\mathcal{G}_{\mathcal{C}}$ (see Fig. 6c), followed by creating an edge from $\mathtt{n}(\mathcal{C}_1^{(1)})$ to $\mathtt{n}(\mathcal{C}_2^{(2)})$. Then, the* FindPaths *function is first invoked for $\mathcal{C}_1^{(1)}$. After the* FindPaths *function terminates for $\mathcal{C}_1^{(1)}$, it is invoked for $\mathcal{C}_1^{(2)}$. Eventually, we obtain the paths $(\mathcal{C}_0^{(1)}, \mathcal{C}_1^{(1)}, \mathcal{C}_2^{(2)})$ and $(\mathcal{C}_0^{(1)}, \mathcal{C}_1^{(2)}, \mathcal{C}_2^{(2)})$ for $\mathcal{C}_2^{(2)}$ (see Fig. 6c).*

C. Lateral Driving Corridors

Lateral driving corridors are obtained in a similar way to longitudinal driving corridors using Alg. 2, but with the addition that sets $\mathcal{D}_k^{(i)}$ are removed (see Alg. 2, ExcludeSets function, lines 23-27)

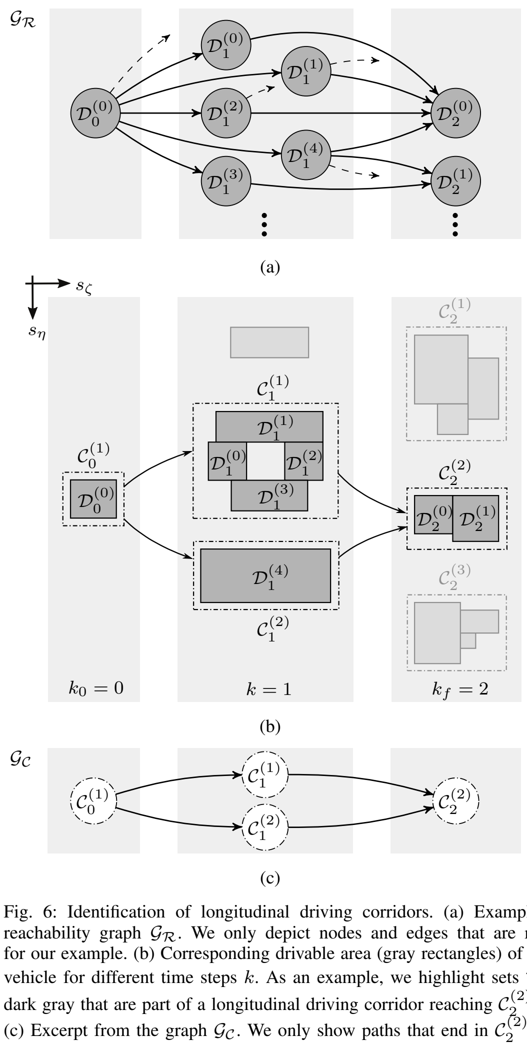


Fig. 6: Identification of longitudinal driving corridors. (a) Example of a reachability graph $\mathcal{G}_\mathcal{R}$. We only depict nodes and edges that are relevant for our example. (b) Corresponding drivable area (gray rectangles) of the ego vehicle for different time steps k. As an example, we highlight sets $\mathcal{D}_k^{(i)}$ in dark gray that are part of a longitudinal driving corridor reaching $\mathcal{C}_2^{(2)}$ at k_f. (c) Excerpt from the graph $\mathcal{G}_\mathcal{C}$. We only show paths that end in $\mathcal{C}_2^{(2)}$.

A1) that are not part of the longitudinal driving corridor C_lon (see Alg. 2, line 24),

A2) for which $\text{proj}_{s_\zeta}(x_{\text{lon},k}) \notin \text{proj}_{s_\zeta}(\mathcal{D}_k^{(i)})$, $x_{\text{lon},k} \in X_\text{lon}$, holds (see Alg. 2, line 25).

Addition A1 ensures that each identified lateral driving corridor is a subset of the longitudinal driving corridor. Since we do not update the reachability graph $\mathcal{G}_\mathcal{R}$ for computational efficiency, all parents, including those that are not part of the longitudinal driving corridor, are returned in Alg. 2, line 12. Addition A2 ensures that we obtain connected sets with a unique passing side at all states $x_{\text{lon},k}$ of the longitudinal trajectory X_lon.

Example *We continue the example in Fig. 6. Let us assume that the selected longitudinal driving corridor is $C_\text{lon} = (\mathcal{C}_0^{(1)}, \mathcal{C}_1^{(1)}, \mathcal{C}_2^{(2)})$. In Fig. 7, the longitudinal positions of X_lon are highlighted. For brevity, we only concentrate on the modifications in Alg. 2 for determining lateral driving*

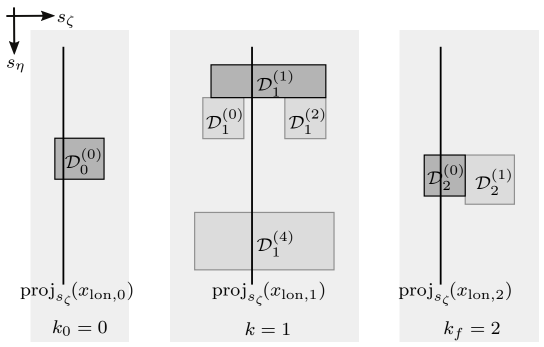


Fig. 7: Identification of lateral driving corridors. As an example, we highlight sets $\mathcal{D}_k^{(i)}$ in dark gray that are part of the lateral driving corridor reaching $\mathcal{D}_2^{(0)}$ at the final time step k_f.

corridors: at time step $k_f = 2$, the connected sets $\mathcal{C}_2^{(1)}$ and $\mathcal{C}_2^{(3)}$ are removed as they are not part of C_lon (addition A1, see Fig. 6b). We also exclude $\mathcal{D}_2^{(1)}$ from $\mathcal{C}_2^{(2)}$ due to addition A2 (see Fig. 7 and Alg. 2, line 4). Then, the FindPaths *function is invoked from $\{\mathcal{D}_2^{(0)}\}$. The parent sets of $\mathcal{D}_2^{(0)}$ are $\mathcal{D}_1^\text{parents} = \{\mathcal{D}_1^{(0)}, \mathcal{D}_1^{(1)}, \mathcal{D}_1^{(2)}, \mathcal{D}_1^{(4)}\}$ (see Fig. 6a). We remove $\mathcal{D}_1^{(4)}$ and $\{\mathcal{D}_1^{(0)}, \mathcal{D}_1^{(2)}\}$ due to addition A1 and A2, respectively (see Fig. 7 and Alg. 2, lines 13-15).*

VI. Determining Collision Avoidance Constraints from Driving Corridors

The longitudinal trajectory planner (see Def. 5) optimizes the longitudinal motion of the ego vehicle along the reference path $\Gamma(s_\zeta)$, i.e., $\forall k:\ \theta_k = \theta_\Gamma(s_{\zeta,k})$ and $s_{\eta,k} = 0$. The lateral trajectory planner (see Def. 6) optimizes the lateral motion of the ego vehicle, so that the ego vehicle is able to deviate from the reference path, i.e., for some k, it may hold that $\theta_k \neq \theta_\Gamma(s_{\zeta,k})$ or $s_{\eta,k} \neq 0$. To guarantee collision avoidance, we constrain all possible trajectories of the ego vehicle to stay within the longitudinal and lateral driving corridor C_lon and C_lat, which is described below.

A. Longitudinal Constraints

Longitudinal collision avoidance is realized by enforcing a minimum and maximum bound $\underline{s}_{\zeta,k}$ and $\overline{s}_{\zeta,k}$ on the longitudinal position $s_{\zeta,k}$ of the reference circle ${}^\text{L}c_k^{(1)}$ (center of rear circle, see Fig. 2) at each time step k. As the shape of the ego vehicle is considered during the reachable set computation, we conclude that the ego vehicle is collision-free if ${}^\text{L}c_k^{(1)}$ is located within the drivable area $\mathcal{D}_k$. Since $C_{\text{lon},k} \subseteq \mathcal{D}_k$, we obtain $\underline{s}_{\zeta,k}$ and $\overline{s}_{\zeta,k}$ for the longitudinal position constraint (4d) from the longitudinal driving corridor $C_{\text{lon},k}$:

$$\underline{s}_{\zeta,k} = \inf\left\{\text{proj}_{s_\zeta}(C_{\text{lon},k})\right\},\quad \overline{s}_{\zeta,k} = \sup\left\{\text{proj}_{s_\zeta}(C_{\text{lon},k})\right\}.$$

B. Lateral Constraints

To model lateral collision avoidance, we introduce the linearized lateral deviation $d_k^{(i)}$ of the i-th circle's center to the reference path at time step k (see Fig. 2):

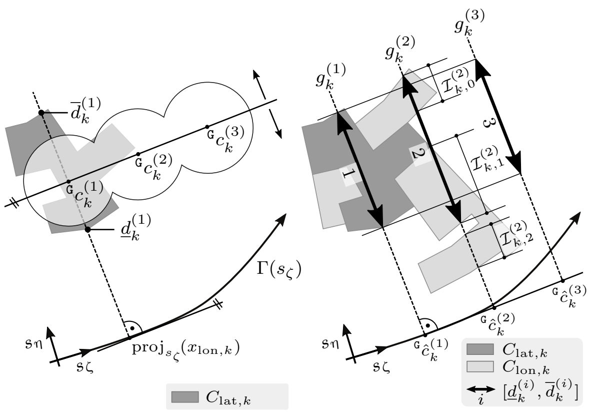
$$d_k^{(i)} = s_{\eta,k} + \frac{i-1}{2}\ell\left(\theta_k - \theta_\Gamma(\mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k}))\right), \tag{8}$$

where $x_{\mathrm{lon},k}$ is the k-th state of the longitudinal trajectory X_{lon} and $i \in \{1,2,3\}$. We are able to guarantee collision avoidance in the lateral direction by limiting the deviation (8) of centers ${}^{\mathtt{G}}c_k^{(i)}$ at each time step k (see (5d)). Below, we assume that $\theta_k = \theta_\Gamma(\mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k}))$ to obtain the limits on (8). Note that the limits obtained are still valid for $\theta_k \neq \theta_\Gamma(\mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k}))$ due to the small-angle approximation (see Sec. II). Below, we use these findings to obtain the admissible lateral deviations for the circles ${}^{\mathtt{G}}c_k^{(i)}$, $i \in \{1,2,3\}$.

1) Admissible Deviations for Reference Circle: Using a similar reasoning as for the longitudinal position constraints in Sec. VI-A, we obtain the minimum and maximum admissible deviations $\underline{d}_k^{(1)}$ and $\overline{d}_k^{(1)}$ of ${}^{\mathtt{G}}c_k^{(1)}$ from the reference path $\Gamma(s_\zeta)$ directly from C_{lat}. Figuratively speaking, we move the ego vehicle perpendicular to the reference path at $\mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k})$ and determine all positions of the lateral driving corridor $C_{\mathrm{lat},k}$ intersecting with ${}^{\mathtt{G}}c_k^{(1)}$ (see Fig. 8a):

$$\underline{d}_k^{(1)} = \inf\left\{\mathrm{proj}_{s_\eta}(C_{\mathrm{lat},k})\right\}, \quad \overline{d}_k^{(1)} = \sup\left\{\mathrm{proj}_{s_\eta}(C_{\mathrm{lat},k})\right\}.$$

2) Admissible Deviations for Center and Front Circles: The maximum and minimum admissible deviations $\underline{d}_k^{(i)}$ and $\overline{d}_k^{(i)}$, $i \in \{2,3\}$, for the center and front circle cannot be extracted directly from the lateral driving corridor C_{lat}. The reason is that $C_{\mathrm{lat},k}$ only contains collision-free, reachable positions of ${}^{\mathtt{L}}c_k^{(1)}$ in the neighborhood of $\mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k})$, $x_{\mathrm{lon},k} \in X_{\mathrm{lon}}$ (see addition A2 in Sec. V-C). Thus, ${}^{\mathtt{G}}c_k^{(2)}$ and ${}^{\mathtt{G}}c_k^{(3)}$ may be even located outside of $C_{\mathrm{lat},k}$ (see Fig. 8a). Yet, we can set the bounds $\underline{d}_k^{(i)}$ and $\overline{d}_k^{(i)}$, $i \in \{2,3\}$, to $\underline{d}_k^{(1)}$ and $\overline{d}_k^{(1)}$, respectively, since we have already considered the shape of the ego vehicle in the reachable set computation (see Sec. IV-D) for $\theta_k = \theta_\Gamma(\mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k}))$.


$\overline{d}_k^{(1)}$
${}^{\mathtt{G}}c_k^{(3)}$
${}^{\mathtt{G}}c_k^{(2)}$
${}^{\mathtt{G}}c_k^{(1)}$
$\underline{d}_k^{(1)}$
$\Gamma(s_\zeta)$
s_η
s_ζ
$\mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k})$
$C_{\mathrm{lat},k}$
$g_k^{(3)}$
$g_k^{(2)}$
$g_k^{(1)}$
$\mathcal{I}_{k,0}^{(2)}$
$\mathcal{I}_{k,1}^{(2)}$
$\mathcal{I}_{k,2}^{(2)}$
1
2
3
${}^{\mathtt{G}}\hat{c}_k^{(3)}$
${}^{\mathtt{G}}\hat{c}_k^{(2)}$
${}^{\mathtt{G}}\hat{c}_k^{(1)}$
$C_{\mathrm{lat},k}$
$C_{\mathrm{lon},k}$
$[\underline{d}_k^{(i)}, \overline{d}_k^{(i)}]$
i

(a) Admissible deviations $d_k^{(1)}$ of center ${}^{\mathtt{G}}c_k^{(1)}$ from $\Gamma(s_\zeta)$ are determined with the lateral driving corridor C_{lat}.

(b) Admissible deviations $d_k^{(i)}$ of centers ${}^{\mathtt{G}}c_k^{(i)}$, $i \in \{2,3\}$, from $\Gamma(s_\zeta)$ are determined with C_{lon}.

Fig. 8: We obtain the minimum and maximum admissible deviation $\underline{d}_k^{(i)}$ and $\overline{d}_k^{(i)}$ through driving corridors C_{lon} and C_{lat}.

However, this choice might be overly conservative for large time steps k, since we restrict the limits $\underline{d}_k^{(i)}$ and $\overline{d}_k^{(i)}$, $i \in \{2,3\}$, to the bounds of $d_k^{(1)}$. We aim to use the longitudinal driving corridor C_{lon} to extend the deviation limits of the center and front circles based on the following observation: the connected sets of the longitudinal driving corridor C_{lon} usually increase in size for greater time steps k (since more states are reachable), so that ${}^{\mathtt{L}}c_k^{(2)}$, ${}^{\mathtt{L}}c_k^{(3)} \in C_{\mathrm{lon},k}$ for ${}^{\mathtt{L}}c_k^{(1)} \in C_{\mathrm{lat},k}$. We can infer that the center and front circle are collision-free within $C_{\mathrm{lon},k}$, since a circle with radius r is guaranteed to be collision-free in the longitudinal driving corridor (see Sec. IV-D). We therefore continue with the determination of the admissible deviations from the reference path for which the center and front circle are contained in the longitudinal driving corridor.

Let us introduce the normal vector $\eta(\mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k}))$ of the reference path at the longitudinal position $\mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k})$. We further introduce the straight line $g_k^{(i)}$ that is parallel to $\eta(\mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k}))$ passing through ${}^{\mathtt{G}}\hat{c}_k^{(i)}$, where ${}^{\mathtt{G}}\hat{c}_k^{(i)}$ is the Cartesian position of the circle centers if the ego vehicle is located on $\Gamma(s_\zeta)$ at $\mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k})$ (see Fig. 8b). The point ${}^{\mathtt{G}}\hat{c}_k^{(i)}$ is obtained with (7) for ${}^{\mathtt{L}}c_k^{(1)} = (\mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k}), 0)^T$. The distances from ${}^{\mathtt{G}}\hat{c}_k^{(i)}$ to all positions in $C_{\mathrm{lon},k}$ intersecting $g_k^{(i)}$ are:

$$\mathcal{Y}_k^{(i)} = \Bigg\{ \overbrace{\eta(\mathrm{proj}_{s_\zeta}(x_{\mathrm{lon},k}))^T \left(T_{\mathtt{L}}^{\mathtt{G}}(s_{\zeta,k}) \begin{pmatrix} s_{\zeta,k} \\ s_{\eta,k} \end{pmatrix} - {}^{\mathtt{G}}\hat{c}_k^{(i)} \right)}^{\text{distance}} \in \mathbb{R} \,\Bigg|\, \begin{pmatrix} s_{\zeta,k} \\ s_{\eta,k} \end{pmatrix} \in C_{\mathrm{lon},k},\; \underbrace{T_{\mathtt{L}}^{\mathtt{G}}(s_{\zeta,k}) \begin{pmatrix} s_{\zeta,k} \\ s_{\eta,k} \end{pmatrix} \in g_k^{(i)}}_{\text{intersection points}} \Bigg\}.$$

If $\mathcal{Y}_k^{(i)} \neq \emptyset$, there exist sets $\mathcal{D}_k^{(j)} \in C_{\mathrm{lon},k}$ intersecting $g_k^{(i)}$ in $F^{\mathtt{G}}$ (see Fig. 8b, $g_k^{(2)}$). We unify the intersection distances $\mathcal{Y}_k^{(i)}$ to intervals $\mathcal{I}_{k,q}^{(i)}$, $q \in \mathbb{N}_0$, as illustrated in Fig. 8b, and consider only those intervals $\mathcal{I}_{k,q}^{(i)}$ where $[\underline{d}_k^{(1)}, \overline{d}_k^{(1)}] \cap \mathcal{I}_{k,q}^{(i)} \neq \emptyset$. The collection of the intervals considered is denoted with $\mathcal{I}_{\mathrm{v}}^{(i)}$. As illustrated in Fig. 8b, interval $\mathcal{I}_{k,2}^{(2)}$ is omitted and we only consider intervals $\mathcal{I}_{k,0}^{(2)}$ and $\mathcal{I}_{k,1}^{(2)}$, i.e., $\mathcal{I}_{\mathrm{v}}^{(2)} = \{\mathcal{I}_{k,0}^{(2)}, \mathcal{I}_{k,1}^{(2)}\}$. The deviation limits are then determined by

$$[\underline{d}_k^{(i)}, \overline{d}_k^{(i)}] = [\underline{d}_k^{(1)}, \overline{d}_k^{(1)}] \cup \mathcal{I}_{\mathrm{v}}^{(i)}, \quad i \in \{2,3\}.$$

VII. Experiments

After introducing implementation details in Sec. VII-A, we validate the drivability of our planned motions by real-world experiments in Sec. VII-B. In Sec. VII-C, we compare our method to motion planners based on discretization and continuous optimization. Subsequently, we demonstrate that our approach works in arbitrarily complex scenarios. A video of the experiments is attached to this paper.

A. Implementation Details

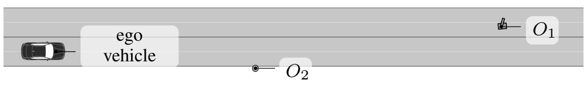
Our approach is implemented partly in Python and C++ on a computer with an Intel Core i7-6700HQ CPU and 16 GB of memory. We use the CommonRoad benchmark suite to model our scenarios [68]. The assessment criteria for driving corridor selection can be chosen arbitrarily. In this work, we select the paths in $\mathcal{G}_{\mathcal{C}}$ with the greatest cumulated area of connected sets, since connected areas of greater size generally yield less restrictive position constraints for trajectory optimization.

For longitudinal and lateral trajectory planning (see Def. 5 and 6), we apply the approach presented in [30], where system (4b) is a fourth-order integrator model with jounce as input and bounded velocity, acceleration, and jerk. System (5b) is a linearized kinematic single-track model with constraints on the steering actuators. We select a quadratic cost function for both the longitudinal and lateral optimization. The longitudinal cost function J_{lon} punishes deviations from a desired velocity, high accelerations, jerk, and inputs. The lateral cost function J_{lat} minimizes the lateral distance to $\Gamma(s_\zeta)$, deviations from $\theta_\Gamma(s_\zeta)$ and $\kappa_\Gamma(s_\zeta)$, as well as high curvature rates and inputs to obtain smooth trajectories.

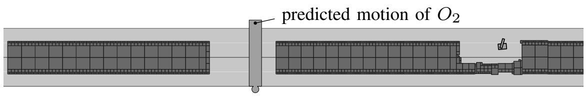
B. Drivability of Planned Motions

We have integrated our approach in a BMW 7-series test vehicle and applied it online to determine the driving corridor. Fig. 9a illustrates the real-world scenario with two lanes, in which we placed a static obstacles O_1 and a simulated pedestrian O_2 in the driveway of the vehicle. The obstacles are detected by LiDAR sensors and we use the set-based prediction *SPOT* [69] to predict the future motion of the pedestrian over time. Note that the prediction method can be replaced and is not part of our algorithm.

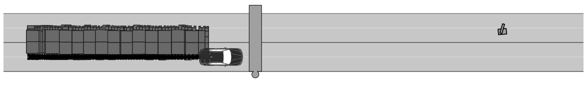
Fig. 9b shows the drivable area at the final time step, from which we infer that two maneuvers exist: a) stopping in front of the pedestrian, and b) swerving. The trajectory that ends in front of the pedestrian and its driving corridor are visualized in Fig. 9c. The swerving trajectory is illustrated in Fig. 9d, in which the vehicle first passes the pedestrian O_2 on the left and then evades the static obstacle O_1 by changing to the original lane. Fig. 9e shows the drivable area of the scenario at $t_{30} = 6.0\,\text{s}$. Even though the solution space is exceedingly small, we are able to detect even narrow passageways by using reachable sets for driving corridor identification. This is of utmost importance in safety-critical scenarios, in which evading may be the last possible feasible maneuver. Even though our approach uses different models for the reachability analysis and trajectory planning, we obtain drivable trajectories as illustrated in Fig. 9f and 9g, which show the nominal and measured velocity and curvature profiles of the executed double lane change maneuver.

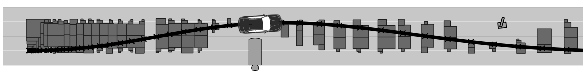


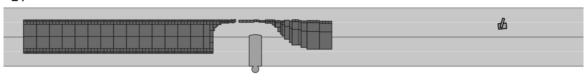
(a) Initial scenario with a static obstacle O_1 and a pedestrian O_2 at $t = 0\,\text{s}$.

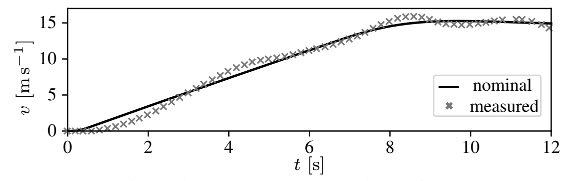


(b) Drivable area $\mathcal{D}_{70}$ at the final time step $t_{70} = 14\,\text{s}$.

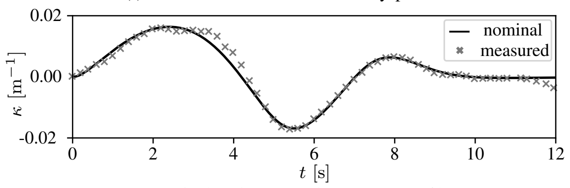

(c) Planned trajectory that ends in front of the pedestrian and the corresponding driving corridor C_{lat}.


(d) Planned evasive trajectory and the corresponding driving corridor C_{lat} at $t_{27} = 5.4\,\text{s}$.


(e) Small solution space within the drivable area $\mathcal{D}_{30}$ at $t_{30} = 6.0\,\text{s}$.



(f) Nominal and measured velocity profile.



(g) Nominal and measured curvature profile.

(h) Images of the vehicle tests with a BMW 7-series.

Fig. 9: Validation of our approach on a BMW 7-series test vehicle (scenario ZAM_Urban-8_1_S-1:2018b).

C. Comparison

1) Sampling-based Motion Planner: We compare our method with a popular sampling-based trajectory planner presented in [15] on the previous scenario to demonstrate the efficacy of our method to detect narrow passages in the solution space. The motion planner in [15] samples a finite set of trajectories that connect the initial state of the ego vehicle with different goal states. The longitudinal and lateral motions are planned separately using quartic and quintic polynomials in F^{L}. The Cartesian product of all longitudinal and lateral motions yields the set of candidate trajectories that are checked for drivability and collisions. Due to the pre-defined s-shape of the trajectories of the sampling-based planner, a double lane change maneuver that is similar to the one of our method (see Fig. 9d) can only be planned in an anticipatory way. We therefore remove the static obstacle O_1 and aim to sample at least one feasible trajectory that evades the pedestrian, which

TABLE I: Parameters for Sampling.

scenario	**goal state**	**min. value**	**max. value**	**step size**
baseline	t in [s]	0.40	10.0	0.20
& critical	$d^{(2)}$ in [m]	-4.00	4.00	1.00
baseline	v_ζ in [m/s]	6.00	18.00	1.5
critical	v_ζ in [m/s]	6.00	18.00	0.2

is particularly challenging due to the small solution space (see Fig. 9e). This scenario is referred to as baseline scenario. Then, we add an additional constraint to the motion planning to increase the difficulty and demand that the ego vehicle must keep a distance of $0.2\,\mathrm{m}$ to the pedestrian. We therefore dilate the predicted occupancies of the pedestrian with a circle with radius $0.2\,\mathrm{m}$ and refer to this scenario as critical scenario. Subsequently, we compare the performance of both planners.

Following [15], we compute 3969 trajectories using the sampling-based planner in the baseline scenario. The sampling parameters are uniformly distributed and can be found in Tab. I. From 3969 trajectories, we found two feasible trajectories evading the pedestrian. However, using the same sampling parameters, no feasible trajectory has been found in the critical scenario. We therefore gradually raise the number of goal states by approximately bisecting the step size for velocity sampling, i.e., we used step sizes $\{1.5, 0.8, 0.4, 0.2\}$m/s, until one feasible trajectory is found. The final sampling parameters are listed in Tab. I. Overall, we needed 26901 trajectories to obtain one feasible trajectory evading the pedestrian in the critical scenario (see Fig. 10). The computation times of the baseline and critical scenario are $3.2\,\mathrm{s}$ and $20.8\,\mathrm{s}$, respectively; thus, the runtime increases by factor 6.5. In contrast, our motion planning approach is able to solve both scenarios without the need for adapting parameters and the computation times do not substantially differ (baseline scenario: $0.078\,\mathrm{s}$, critical scenario: $0.076\,\mathrm{s}$).

This scenario particularly demonstrates that sampling-based motion planners struggle to detect narrow passage ways in the solution space for trajectory planning due to the discretization of the state space. In contrast, our method can automatically detect narrow passages without increased computational effort. One might argue that we have used a naiv sampling strategy, however, manually tuning or learning adaptive sampling strategies for different traffic situations is error-prone and difficult. In comparison, our method for driving corridor identification can be applied immediately without parameter tuning.

2) Mixed-Integer Quadratic Programming: We analyze the influence of an increasing number of obstacles on the runtime behavior of our method. For comparison, we select the motion planner proposed in [70] based on mixed-integer quadratic programming (MIQP). The reason for selecting a MIQP planner is two-fold: 1) it is a continuous optimization method, and thus, does not suffer from discretization effects, and 2) mixed-integer programming has gained increasing interest over the past years, e.g., see [6], [20], [70]–[72], as binary variables enable to incorporate discrete tactical decisions into the trajectory optimization.

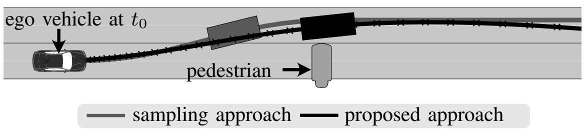
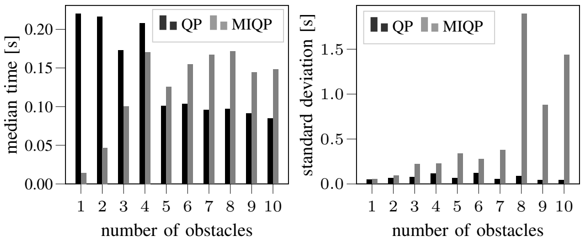


Fig. 10: Critical scenario at $t_{27} = 5.4\,\mathrm{s}$.



(a) Median comp. times (b) Standard deviation of comp. times

Fig. 11: Runtime behavior of our approach (label QP) and a MIQP planner [70] (label MIQP) in randomly generated scenarios with up to ten obstacles.

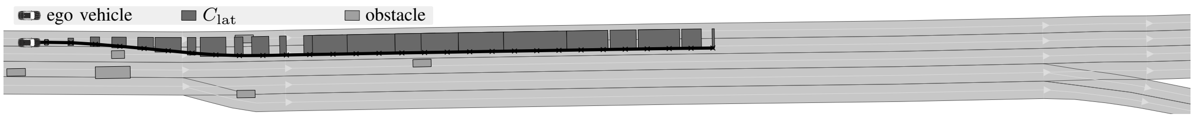
We consider the same road as in Fig. 10 and gradually increase the number $\sigma \in \{1, \ldots, 10\}$ of static obstacles in the environment. The obstacles are represented by rectangles, i.e., $[s_x - l_{\mathrm{obs}}/2, s_x + l_{\mathrm{obs}}/2] \times [s_y - w_{\mathrm{obs}}/2, s_y + w_{\mathrm{obs}}/2]$, that are randomly sampled, where $s_x \in [25, 180]\,\mathrm{m}$, $s_y \in [-1.75, 5.25]\,\mathrm{m}$, $l_{\mathrm{obs}} \in [1.0, 20.0]\,\mathrm{m}$, $w_{\mathrm{obs}} \in [0.5, 4.0]\,\mathrm{m}$. Since our motion planner uses a kinematic single-track model for optimization that is more restrictive compared to the third-order integrator models of [70], we only consider scenarios for the analysis that both planners could solve. Overall, we generated 50 scenarios for each σ. In all scenarios, trajectories are planned for 50 time steps with $\Delta t = 0.2\,\mathrm{s}$ and the initial velocity of the ego vehicle was 15 m/s. To compute the solution for the MIQP, we use the solver Gurobi [73], version 9.0.2.

As illustrated in Fig. 11a, the median computation times for both planners are comparable. While the MIQP planner has lower computation times for scenarios with up to 4 obstacles, our approach is faster for scenarios with more than 4 obstacles. Our method is already applicable for complex road geometries and obstacles with arbitrary shapes, whereas the MIQP planner is tailored to our experimental setup, i.e., lane boundaries are constant and obstacles have rectangular shapes. Modeling complex obstacle geometries, e.g., arbitrary polygonal shapes, increases the computation times of the MIQP planner [70].

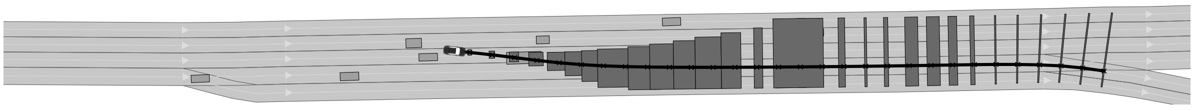
The MIQP planner solves many scenarios rather fast, however, the runtime behavior is highly volatile. As shown in Fig. 11b, the standard deviation of the computation times are particularly high for scenarios with multiple obstacles and have the tendency to increase with a raising number of obstacles. In contrast, the computation times of our method vary only slightly over the different scenarios. The experiment indicates that the runtime behavior of our method is reliable, which is of high importance for real-time applications.

D. Motion Planning in a Complex Scenario

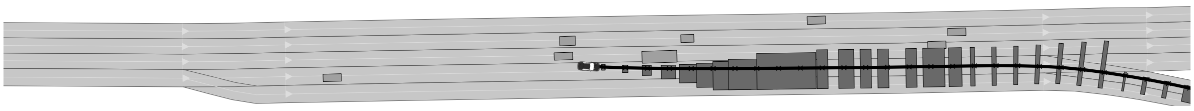
Let us consider a complex scenario, where the ego vehicle is in the far left lane and plans to take the highway off-



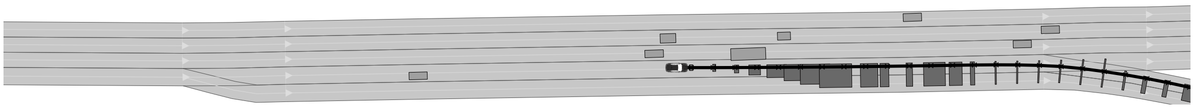
(a) Scenario at $t = 0\,\text{s}$.



(b) Scenario at $t = 3.6\,\text{s}$.



(c) Scenario at $t = 4.8\,\text{s}$.



(d) Scenario at $t = 5.6\,\text{s}$.

Fig. 12: Planned trajectories of the ego vehicle for different planning cycles (scenarios DEU_A9-3_{1,...,15}_T-1:2018b).

ramp (see Fig. 12a). Even for experienced human drivers, this maneuver is challenging as multiple lane changes need to be planned. In order to account for realistic movements by other traffic participants, this scenario is based on real traffic that we recorded using a BMW 7-series vehicle. Again, we use *SPOT* [69] to predict the future motion of the other traffic participants and assume that traffic participants will remain in their current lanes, and accelerate or decelerate only slightly. Since this maneuver requires longer planning horizons, we replan the trajectory for consecutive planning cycles. In each planning cycle, trajectories are planned for 30 time steps using $\Delta t = 0.2\,\text{s}$; a new planning cycle starts every $0.4\,\text{s}$.

Fig. 12 illustrates the planned trajectories and the corresponding lateral driving corridors at selected times t. Other traffic participants are shown in their measured state without uncertainties at time t. As can be seen in Fig. 12b, our method enables the ego vehicle to merge into small gaps while finally reaching the highway off-ramp (see Fig. 12d). The median computation times of the reachability analysis and of trajectory planning, including the driving corridor identification, are $\approx 131\,\text{ms}$ and $\approx 154\,\text{ms}$, respectively. This scenario highlights that driving corridors allow the ego vehicle to maneuver even in dense traffic with multiple road users.

E. Scenarios with a Small Solution Space

Next, we analyze the influence of a decreasing solution space. We therefore group the computation times of our approach (reachability analysis and trajectory planning including driving corridor identification) obtained in Sec. VII-C2 according to the size of the drivable area cumulated over all time steps. As shown in Fig. 13, the computation times tend to decrease given that the cumulative drivable area shrinks. Since the number of driving corridors can be limited (thus, the number of planned trajectories), the main factor for the decrease in computation time is the reachability analysis that we further examine below.

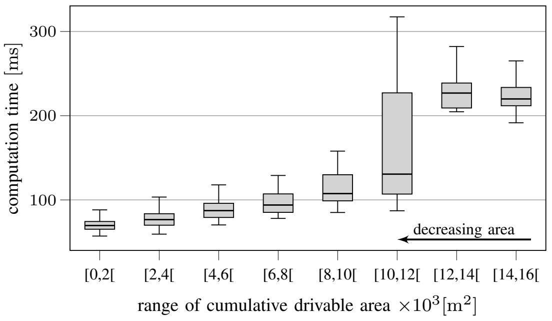


Fig. 13: The computation times of our approach have the tendency to decrease when the cumulative drivable area decreases. For better visibility, outliers of the boxplots are not shown.

We increase the criticality of the CommonRoad scenario USA_US101-6_1_T-1:2018b (see Fig. 14) featuring 29 dynamic obstacles. By gradually raising the initial velocity of the ego vehicle by $1.4\,\text{m/s}$, the traffic situations becomes more dangerous compared to the baseline scenario, when using, e.g., the Time-to-Collision or the Inter-Vehicle-Time, as thread-measure [74]. We emphasize that criticality measures are not part of the presented algorithm; an overview can be found in [74]. For each scenario, the reachable set is computed for 50 time steps with $\Delta t = 0.1\,\text{s}$ and repeated 20 times to account for fluctuations in the computation time. As shown in Tab. II, the median computation time of the reachable set decreases with increasing criticality of the traffic situation, since fewer set operations have to be performed, i.e., the number of base sets cumulated over all time steps k decreases. Also, the size of the cumulated drivable area is reduced by roughly 75%

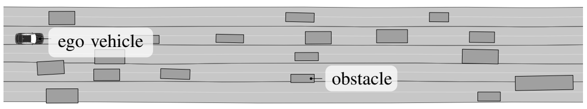


Fig. 14: Scenario USA_US101-6_1_T-1 of the CommonRoad benchmark suite. The obstacles and the ego vehicle are depicted at the initial time step.

TABLE II: Decreasing median computation times, number of base sets, and size of the drivable area cumulated over all time steps k, when increasing the initial velocity in scenario USA_US101-6_1_T-1.

Init. Vel.	**Comp. Time**	**No. Base Sets**	**Red. Drivable Area**
16.79 m/s	74.76 ms	2025	0.00 %
18.19 m/s	69.51 ms	1843	8.65 %
19.59 m/s	65.65 ms	1712	17.55 %
20.99 m/s	58.90 ms	1561	27.25 %
22.39 m/s	52.93 ms	1346	37.95 %
23.79 m/s	45.46 ms	1112	49.00 %
25.19 m/s	37.79 ms	899	59.47 %
26.59 m/s	31.30 ms	678	67.97 %
27.99 m/s	27.19 ms	535	75.40 %

when comparing the baseline scenario with the most-critical scenario. Thus, our approach is particularly suited for complex situations with small solution spaces, where fast reaction times are a necessity to avoid collisions.

VIII. Conclusions

This paper presents a novel motion planning approach for automated vehicles by combining reachability analysis with convex optimization. In contrast to most existing work on motion planning, our approach can be applied in arbitrarily complex traffic situations. We showed that set-based techniques have the potential to reduce the computation time of motion planners when the criticality of the scenario increases, i.e., the solution space becomes smaller and more convoluted. In real vehicle tests, we validated the extraction of collision avoidance constraints from the obtained driving corridors to plan drivable trajectories. Our results indicate that the presented approach can drastically enhance the safety of automated vehicles and their ability to determine complex driving maneuvers. Additionally, we are able to constrain driving corridors to end in predefined terminal states. Thus, automated vehicles can choose the most appropriate maneuver to achieve the desired driving task.

Appendix A
Forward Propagation

The forward propagation is performed similar to [9]. However, in [9], the absolute limits of the acceleration are restricted to be equal ($|\underline{a}_\zeta| = |\overline{a}_\zeta|$ and $|\underline{a}_\eta| = |\overline{a}_\eta|$), whereas we allow the maximum braking deceleration to be different from the maximum acceleration. According to [9], the lower and upper bounds of the reachable positions and velocities for system model (6) at $t = \Delta t$ are obtained by applying a bang-bang input with switching time $\gamma\Delta t$ for $\gamma \in [0, 1]$. Below, we present the solution only in the longitudinal direction, the solution in the lateral direction is obtained similarly. The lower bounds $s^{\mathrm{lo}}_\zeta(\gamma)$ and $v^{\mathrm{lo}}_\zeta(\gamma)$ on the position and velocity in the longitudinal direction are obtained by applying full acceleration until time $\gamma\Delta t$ followed by full deceleration:

$$\begin{aligned} s^{\mathrm{lo}}_\zeta(\gamma) = s_\zeta(0) + v_\zeta(0)\Delta t + \frac{1}{2}\underline{a}_\zeta\Delta t^2(1 - 2\gamma + \gamma^2) \\ + \frac{1}{2}\overline{a}_\zeta\Delta t^2(2\gamma - \gamma^2), \end{aligned} \tag{9a}$$

$$v^{\mathrm{lo}}_\zeta(\gamma) = v_\zeta(0) + \underline{a}_\zeta\Delta t(1 - \gamma) + \overline{a}_\zeta\gamma\Delta t. \tag{9b}$$

The upper bounds $s^{\mathrm{hi}}_\zeta(\gamma)$ and $v^{\mathrm{hi}}_\zeta(\gamma)$ on the position and velocity in the longitudinal direction are obtained by applying full deceleration until time $\gamma\Delta t$ followed by full acceleration for $(1 - \gamma)\Delta t$:

$$\begin{aligned} s^{\mathrm{hi}}_\zeta(\gamma) = s_\zeta(0) + v_\zeta(0)\Delta t + \frac{1}{2}\overline{a}_\zeta\Delta t^2(1 - 2\gamma + \gamma^2) \\ + \frac{1}{2}\underline{a}_\zeta\Delta t^2(2\gamma - \gamma^2), \end{aligned} \tag{10a}$$

$$v^{\mathrm{hi}}_\zeta(\gamma) = v_\zeta(0) + \overline{a}_\zeta\Delta t(1 - \gamma) + \underline{a}_\zeta\gamma\Delta t. \tag{10b}$$

From this, we can infer that a polytope $\mathcal{P}^{(i)}_{\zeta,k-1}$ can be propagated as shown in [9]:

$$\mathcal{P}^{(i)}_{\zeta,k} = \underbrace{\left(\begin{pmatrix}1 & \Delta t\\ 0 & 1\end{pmatrix}\mathcal{P}^{(i)}_{\zeta,k-1} \oplus \mathcal{P}_{u,\zeta}(\Delta t)\right)}_{\text{solution of (6a) + (6c)}} \cap \underbrace{\left(\mathbb{R} \times [\underline{v}_\zeta, \overline{v}_\zeta]\right)}_{\text{velocity constraints (6b)}},$$

where $\oplus$ denotes the Minkowski sum and $\mathcal{P}_{u,\zeta}(\Delta t)$ is the set of all states that can be reached from the initial state $(s_\zeta, v_\zeta)^T = (0, 0)^T$ when all admissible inputs are applied (6c). For computational reasons, $\mathcal{P}_{u,\zeta}(\Delta t)$ is over-approximated with a predefined number of halfspaces. Using (9), the linear equation for one halfspace at some γ is [9]:

$$\begin{pmatrix}\frac{\mathrm{d}s^{\mathrm{lo}}_\zeta}{\mathrm{d}\gamma}\\ \frac{\mathrm{d}v^{\mathrm{lo}}_\zeta}{\mathrm{d}\gamma}\end{pmatrix}^T\Bigg|_\gamma \begin{pmatrix}0 & 1\\ -1 & 0\end{pmatrix}^T\begin{pmatrix}s_\zeta\\ v_\zeta\end{pmatrix} \le \begin{pmatrix}\frac{\mathrm{d}s^{\mathrm{lo}}_\zeta}{\mathrm{d}\gamma}\\ \frac{\mathrm{d}v^{\mathrm{lo}}_\zeta}{\mathrm{d}\gamma}\end{pmatrix}^T\Bigg|_\gamma \begin{pmatrix}0 & 1\\ -1 & 0\end{pmatrix}^T\begin{pmatrix}s^{\mathrm{lo}}_\zeta\\ v^{\mathrm{lo}}_\zeta\end{pmatrix}\Bigg|_\gamma.$$

Similarly, we obtain the halfspace for (10).

Appendix B
Removal of Forbidden States

This section elaborates on the determination of the rectangular set $\mathcal{A}^\epsilon$ that over-approximates the reachable positions of centers ${}^{\mathtt{L}}c^{(i)}_k$ for the removal of forbidden states in the reachable set computation (see Sec. IV-D). Particularly, we present a solution for the following problem:

Problem 1 *Given that $\theta_k = \theta_\Gamma(\mathrm{proj}_{s_\zeta}({}^{\mathtt{L}}c^{(1)}_k))$, we aim to find the rectangular set $\mathcal{A}^\epsilon = [0, \overline{\epsilon}_\zeta] \times [-\underline{\epsilon}_\eta, \overline{\epsilon}_\eta]$ such that $\forall {}^{\mathtt{L}}c^{(1)}_k \in \mathcal{D}^{\mathtt{rprt}(q)}_k \forall i \in \{1, 2, 3\}$: ${}^{\mathtt{L}}c^{(i)}_k \in \mathcal{D}^{\mathtt{rprt}(q)}_k \oplus \mathcal{A}^\epsilon$.*

Once the necessary assumptions have been introduced for computing $\mathcal{A}^\epsilon$ in Appendix B-A, the longitudinal enlargement $\overline{\epsilon}_\zeta$ is derived in Appendix B-B, followed by the lateral enlargements $\overline{\epsilon}_\eta$ and $\underline{\epsilon}_\eta$.

A. Notation and Assumptions

We denote the respective minimum and maximum lateral position of $\mathcal{D}^{\mathtt{rprt}(q)}_k$ by $\underline{s}^{\mathtt{rprt}(q)}_{\eta,k} = \inf\left\{\mathrm{proj}_{s_\eta}(\mathcal{D}^{\mathtt{rprt}(q)}_k)\right\}$

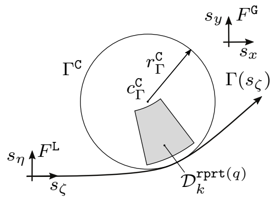


Fig. 15: Local approximation of reference path $\Gamma(s_\zeta)$ with circle $\Gamma^{\mathtt{C}}$.

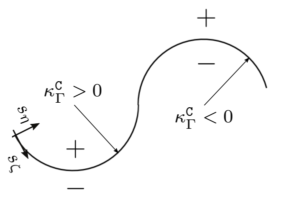
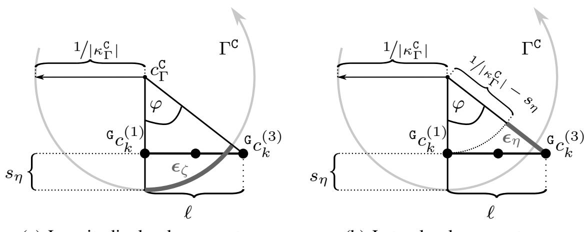

+
−
$\kappa_\Gamma^{\mathtt{C}} > 0$
$\kappa_\Gamma^{\mathtt{C}} < 0$
s_η
s_ζ
+
−

Fig. 16: Positive and negative domain of lateral coordinate s_η.


$1/|\kappa_\Gamma^{\mathtt{C}}|$
$\Gamma^{\mathtt{C}}$
$c_\Gamma^{\mathtt{C}}$
φ
${}^{\mathtt{G}}c_k^{(1)}$
${}^{\mathtt{G}}c_k^{(3)}$
ϵ_ζ
s_η
ℓ
$1/|\kappa_\Gamma^{\mathtt{C}}|$
$\Gamma^{\mathtt{C}}$
$1/|\kappa_\Gamma^{\mathtt{C}}| - s_\eta$
φ
ϵ_η
${}^{\mathtt{G}}c_k^{(1)}$
${}^{\mathtt{G}}c_k^{(3)}$
s_η
ℓ

(a) Longitudinal enlargement ϵ_ζ. (b) Lateral enlargement ϵ_η.

Fig. 17: Computation of longitudinal and lateral enlargements.

and $\overline{s}_{\eta,k}^{\mathtt{rprt}(q)} = \sup\left\{\operatorname{proj}_{s_\eta}(\mathcal{D}_k^{\mathtt{rprt}(q)})\right\}$. The minimum and maximum curvature of $\Gamma(s_\zeta)$ are denoted by $\underline{\kappa}_\Gamma$ and $\overline{\kappa}_\Gamma$. We assume that we can locally approximate the reference path $\Gamma(s_\zeta)$ with a circle $\Gamma^{\mathtt{C}}$ in proximity of $\mathcal{D}_k^{\mathtt{rprt}(q)}$ as shown in Fig. 15. The center, curvature, and radius of $\Gamma^{\mathtt{C}}$ are denoted as $c_\Gamma^{\mathtt{C}}$, $\kappa_\Gamma^{\mathtt{C}}$, and $r_\Gamma^{\mathtt{C}} = 1/|\kappa_\Gamma^{\mathtt{C}}|$, respectively. For $\kappa_\Gamma^{\mathtt{C}} \geq 0$, the lateral coordinates s_η are positive on the inner side of the curve (see Fig. 16) and for $\kappa_\Gamma^{\mathtt{C}} < 0$, the lateral coordinates s_η are negative on the inner side of the curve (see Fig. 16). Based on $\Gamma^{\mathtt{C}}$ and $\mathcal{D}_k^{\mathtt{rprt}(q)}$, we compute $\overline{\epsilon}_\zeta$, $\overline{\epsilon}_\eta$ and $\underline{\epsilon}_\eta$ to determine $\mathcal{A}^\epsilon$. We further assume that $\mathcal{D}_k^{\mathtt{rprt}(q)} \oplus \mathcal{A}^\epsilon$ lies completely inside the unique projection domain of $\Gamma(s_\zeta)$ [75].

B. Longitudinal Enlargement

We wish to enclose all longitudinal positions of centers ${}^{\mathtt{L}}c_k^{(i)}$, $i \in \{1,2,3\}$, for all ${}^{\mathtt{L}}c_k^{(1)} \in \mathcal{D}_k^{\mathtt{rprt}(q)}$ using $\mathcal{A}^\epsilon$. First, we derive the function $\epsilon_\zeta(s_{\eta,k}, \kappa_\Gamma^{\mathtt{C}})$ for computing the longitudinal enlargement based on a single position ${}^{\mathtt{L}}c_k^{(1)}$. Second, we determine the over-approximative longitudinal enlargement $\overline{\epsilon}_\zeta$ for a set of positions based on $\epsilon_\zeta(s_{\eta,k}, \kappa_\Gamma^{\mathtt{C}})$.

1) Single Position: Given ${}^{\mathtt{L}}c_k^{(1)}$, we aim to determine the longitudinal position coordinate $\hat{s}_{\zeta,k}$ of ${}^{\mathtt{L}}c_k^{(3)}$ relative to ${}^{\mathtt{L}}c_k^{(1)}$, such that $\hat{s}_{\zeta,k} = s_{\zeta,k} + \epsilon_\zeta$. Note that we are interested in the maximum longitudinal enlargement, therefore, it is sufficient to only consider ${}^{\mathtt{L}}c_k^{(3)}$ (${}^{\mathtt{L}}c_k^{(3)}$ is farther away from ${}^{\mathtt{L}}c_k^{(1)}$ than ${}^{\mathtt{L}}c_k^{(2)}$). As illustrated in Fig. 17a, ϵ_ζ is the arc length of a circle sector with radius $r_\Gamma^{\mathtt{C}} = 1/|\kappa_\Gamma^{\mathtt{C}}|$ and central angle φ:

$$\epsilon_\zeta(s_{\eta,k}, \kappa_\Gamma^{\mathtt{C}}) = \frac{\varphi(s_{\eta,k}, \kappa_\Gamma^{\mathtt{C}})}{|\kappa_\Gamma^{\mathtt{C}}|}, \tag{11}$$

where the central angle is computed as follows:

$$\varphi(s_{\eta,k}, \kappa_\Gamma^{\mathtt{C}}) = \begin{cases} \arctan\left(\frac{\ell}{1/|\kappa_\Gamma^{\mathtt{C}}| - s_{\eta,k}}\right) & \text{for } \kappa_\Gamma^{\mathtt{C}} \geq 0 \\ \arctan\left(\frac{\ell}{1/|\kappa_\Gamma^{\mathtt{C}}| + s_{\eta,k}}\right) & \text{for } \kappa_\Gamma^{\mathtt{C}} < 0. \end{cases} \tag{12}$$

As the lateral coordinates are positive on the inner side of the curve for $\kappa_\Gamma^{\mathtt{C}} > 0$ (see Fig. 16), we need to subtract $s_{\eta,k}$ from $1/|\kappa_\Gamma^{\mathtt{C}}|$ to compute $\varphi(s_{\eta,k}, \kappa_\Gamma^{\mathtt{C}})$ (see Fig. 17a). For $\kappa_\Gamma^{\mathtt{C}} < 0$, the lateral coordinates are negative on the inner side of the curve. Therefore, we add $s_{\eta,k}$ to $1/|\kappa_\Gamma^{\mathtt{C}}|$ for computing $\varphi(s_{\eta,k}, \kappa_\Gamma^{\mathtt{C}})$. Note that $\lim_{|\kappa_\Gamma^{\mathtt{C}}| \to 0} \epsilon_\zeta(s_{\eta,k}, \kappa_\Gamma^{\mathtt{C}}) \to \ell$.

2) Set of Positions: For computational efficiency, we over-approximate (11) for a set of positions $\mathcal{D}_k^{\mathtt{rprt}(q)}$. Thus, the question arises, for which $s_{\eta,k} \in \operatorname{proj}_{s_\eta}(\mathcal{D}_k^{\mathtt{rprt}(q)})$ and $\kappa_\Gamma^{\mathtt{C}} \in [\underline{\kappa}_\Gamma, \overline{\kappa}_\Gamma]$ does the longitudinal enlargement (11) reach its maximum.

Lemma 1 *For $\kappa_\Gamma^{\mathtt{C}} \geq 0$, $\epsilon_\zeta(s_{\eta,k}, \kappa_\Gamma^{\mathtt{C}})$ (11) is monotonically increasing in $s_{\eta,k}$. Furthermore, for $\kappa_\Gamma^{\mathtt{C}} < 0$, $\epsilon_\zeta(s_{\eta,k}, \kappa_\Gamma^{\mathtt{C}})$ (11) is monotonically decreasing in $s_{\eta,k}$.*

Proof 1 *Given that $s'_{\eta,k} > s^*_{\eta,k}$ and $\kappa_\Gamma^{\mathtt{C}} \geq 0$, we have:*

$$\begin{aligned} \epsilon_\zeta(s'_{\eta,k}, \kappa_\Gamma^{\mathtt{C}}) &> \epsilon_\zeta(s^*_{\eta,k}, \kappa_\Gamma^{\mathtt{C}}) \\ \arctan\left(\frac{\ell}{1/|\kappa_\Gamma^{\mathtt{C}}| - s'_{\eta,k}}\right) &> \arctan\left(\frac{\ell}{1/|\kappa_\Gamma^{\mathtt{C}}| - s^*_{\eta,k}}\right) \\ 1/|\kappa_\Gamma^{\mathtt{C}}| - s'_{\eta,k} &< 1/|\kappa_\Gamma^{\mathtt{C}}| - s^*_{\eta,k} \\ s'_{\eta,k} &> s^*_{\eta,k}. \end{aligned}$$

Using the same reasoning, we obtain for $\kappa_\Gamma^{\mathtt{C}} < 0$ that $\epsilon_\zeta(s'_{\eta,k}, \kappa_\Gamma^{\mathtt{C}}) < \epsilon_\zeta(s^*_{\eta,k}, \kappa_\Gamma^{\mathtt{C}})$. □

Lemma 2 *For $\kappa_\Gamma^{\mathtt{C}} \geq 0$ and $(s_{\zeta,k}, s_{\eta,k}) \in \mathcal{D}_k^{\mathtt{rprt}(q)}$, (11) reaches its maximum for $s_{\eta,k} = \overline{s}_{\eta,k}^{\mathtt{rprt}(q)}$. For $\kappa_\Gamma^{\mathtt{C}} < 0$, (11) reaches its maximum for $s_{\eta,k} = \underline{s}_{\eta,k}^{\mathtt{rprt}(q)}$.*

Proof 2 *From Lemma 1, it follows that (11) is monotonically increasing in $s_{\eta,k}$ for $\kappa_\Gamma^{\mathtt{C}} \geq 0$. Therefore, we select $\overline{s}_{\eta,k}^{\mathtt{rprt}(q)}$ which is the maximum lateral position in $\mathcal{D}_k^{\mathtt{rprt}(q)}$ to compute (11). Similarly, as (11) is monotonically decreasing in $s_{\eta,k}$ for $\kappa_\Gamma^{\mathtt{C}} < 0$, we select the minimum lateral coordinate $\underline{s}_{\eta,k}^{\mathtt{rprt}(q)}$ of $\mathcal{D}_k^{\mathtt{rprt}(q)}$ to compute (11).* □

Lemma 3 *For any $s_{\eta,k} \in \operatorname{proj}_{s_\eta}(\mathcal{D}_k^{\mathtt{rprt}(q)})$ and $\kappa_\Gamma^{\mathtt{C}} \in [\underline{\kappa}_\Gamma, \overline{\kappa}_\Gamma]$, (11) is bounded by*

$$\overline{\epsilon}_\zeta = \sup\left\{\epsilon_\zeta\left(\overline{s}_{\eta,k}^{\mathtt{rprt}(q)}, [0, \overline{\kappa}_\Gamma]\right) \cup \epsilon_\zeta\left(\underline{s}_{\eta,k}^{\mathtt{rprt}(q)}, [\underline{\kappa}_\Gamma, 0]\right)\right\}. \tag{13}$$

Proof 3 *According to Lemma 2, $\overline{s}_{\eta,k}^{\mathtt{rprt}(q)}$ and $\underline{s}_{\eta,k}^{\mathtt{rprt}(q)}$ maximize (11) for $\kappa_\Gamma^{\mathtt{C}} \geq 0$ and $\kappa_\Gamma^{\mathtt{C}} < 0$, respectively. In contrast to $s_{\eta,k}$, (11) is not monotonic in $\kappa_\Gamma^{\mathtt{C}}$. Using interval arithmetics [76], we obtain the upper bound on (11) over the intervals $[0, \overline{\kappa}_\Gamma]$ and $[\underline{\kappa}_\Gamma, 0]$.* □

As (13) becomes increasingly over-approximative for larger intervals of curvatures, we pre-compute (13) for smaller intervals and store the values in a look-up table generated offline.

C. Lateral Enlargement

Similarly to the longitudinal enlargement, we wish to enlarge $\mathcal{D}_k^{\mathtt{rprt}(q)}$ in the lateral direction such that all lateral positions of centers ${}^{\mathtt{L}}c_k^{(i)}$, $i \in \{1,2,3\}$, are enclosed for all

${}^{\mathtt{L}}c_k^{(1)} \in \mathcal{D}_k^{\mathtt{rprt}(q)}$. Similarly to the previous section, we first derive the function $\epsilon_\eta(s_{\eta,k}, \kappa_\Gamma^{\mathtt{C}})$ for determining the lateral enlargements based on a single position ${}^{\mathtt{L}}c_k^{(1)}$. Second, we determine the over-approximative lateral enlargements $\underline{\epsilon}_\eta$ and $\overline{\epsilon}_\eta$ based on $\epsilon_\eta(s_{\eta,k}, \kappa_\Gamma^{\mathtt{C}})$.

1) Single Position: Given ${}^{\mathtt{L}}c_k^{(1)}$, we aim to determine the lateral position coordinate $\hat{s}_{\eta,k}$ of ${}^{\mathtt{L}}c_k^{(3)}$ relative to ${}^{\mathtt{L}}c_k^{(1)}$, such that $\hat{s}_{\eta,k} = s_{\eta,k} - \epsilon_\eta$ (again, it is sufficient to only compute it for ${}^{\mathtt{L}}c_k^{(3)}$). As illustrated in Fig. 17b, we use the Pythagorean theorem to determine ϵ_η:

$$\epsilon_\eta(s_{\eta,k}, \kappa_\Gamma^{\mathtt{C}}) = \begin{cases} \sqrt{(1/|\kappa_\Gamma^{\mathtt{C}}| - s_{\eta,k})^2 + \ell^2} \\ -(1/|\kappa_\Gamma^{\mathtt{C}}| - s_{\eta,k}) , & \text{for } \kappa_\Gamma^{\mathtt{C}} \geq 0, \\ \sqrt{(1/|\kappa_\Gamma^{\mathtt{C}}| + s_{\eta,k})^2 + \ell^2} \\ -(1/|\kappa_\Gamma^{\mathtt{C}}| + s_{\eta,k}) , & \text{for } \kappa_\Gamma^{\mathtt{C}} < 0. \end{cases} \tag{14}$$

As the sign of the lateral coordinate s_η changes on the inner side of the curve depending on the sign of $\kappa_\Gamma^{\mathtt{C}}$ (see Fig. 16), we distinguish two cases for (14) similar to (12). Note that $\lim\limits_{|\kappa_\Gamma^{\mathtt{C}}| \to 0^+} \epsilon_\eta(s_{\eta,k}, \kappa_\Gamma^{\mathtt{C}}) \to 0$.

2) Set of Positions: For computational efficiency, we over-approximate (14) for a set of positions $\mathcal{D}_k^{\mathtt{rprt}(q)}$. Thus, the question arises for which $s_{\eta,k} \in \mathrm{proj}_{s_\eta}(\mathcal{D}_k^{\mathtt{rprt}(q)})$ and $\kappa_\Gamma^{\mathtt{C}} \in [\underline{\kappa}_\Gamma, \overline{\kappa}_\Gamma]$ the lateral enlargement (14) reaches its maximum.

Lemma 4 *For $\kappa_\Gamma^{\mathtt{C}} \geq 0$, $\epsilon_\eta(s_{\eta,k}, \kappa_\Gamma^{\mathtt{C}})$ (14) is monotonically increasing in $s_{\eta,k}$. Furthermore, for $\kappa_\Gamma^{\mathtt{C}} < 0$, $\epsilon_\eta(s_{\eta,k}, \kappa_\Gamma^{\mathtt{C}})$ (14) is monotonically decreasing in $s_{\eta,k}$.*

Proof 4 *The partial derivative of* (14) *for $\kappa_\Gamma^{\mathtt{C}} \geq 0$ with respect to $s_{\eta,k}$ is*

$$\frac{\partial \epsilon_\eta}{\partial s_{\eta,k}}(s_{\eta,k}, \kappa_\Gamma^{\mathtt{C}}) = \begin{cases} +1 - \frac{1/|\kappa_\Gamma^{\mathtt{C}}| - s_{\eta,k}}{\sqrt{(1/|\kappa_\Gamma^{\mathtt{C}}| - s_{\eta,k})^2 + \ell^2}}, & \text{for } \kappa_\Gamma^{\mathtt{C}} \geq 0, \\ -1 + \frac{1/|\kappa_\Gamma^{\mathtt{C}}| + s_{\eta,k}}{\sqrt{(1/|\kappa_\Gamma^{\mathtt{C}}| + s_{\eta,k})^2 + \ell^2}}, & \text{for } \kappa_\Gamma^{\mathtt{C}} < 0. \end{cases} \tag{15}$$

Since $1/|\kappa_\Gamma^{\mathtt{C}}| \mp s_{\eta,k} > 0$ must hold such that $s_{\eta,k}$ is within the unique projection domain of $\Gamma^{\mathtt{C}}$ and $1/|\kappa_\Gamma^{\mathtt{C}}| \mp s_{\eta,k} < \sqrt{(1/|\kappa_\Gamma^{\mathtt{C}}| \mp s_{\eta,k})^2 + \ell^2}$, we have:

$$0 < \frac{1/|\kappa_\Gamma^{\mathtt{C}}| \mp s_{\eta,k}}{\sqrt{(1/|\kappa_\Gamma^{\mathtt{C}}| \mp s_{\eta,k})^2 + \ell^2}} < 1.$$

Thus, it holds that

$$\forall \kappa_\Gamma^{\mathtt{C}} \geq 0 : \frac{\partial \epsilon_\eta}{\partial s_{\eta,k}}(s_{\eta,k}, \kappa_\Gamma^{\mathtt{C}}) > 0,$$
$$\forall \kappa_\Gamma^{\mathtt{C}} < 0 : \frac{\partial \epsilon_\eta}{\partial s_{\eta,k}}(s_{\eta,k}, \kappa_\Gamma^{\mathtt{C}}) < 0.$$

Therefore, (14) *is monotonically increasing in $s_{\eta,k}$ for $\kappa_\Gamma^{\mathtt{C}} \geq 0$ and monotonically decreasing in $s_{\eta,k}$ for $\kappa_\Gamma^{\mathtt{C}} < 0$.* □

Lemma 5 *For $\kappa_\Gamma^{\mathtt{C}} \geq 0$ and $(s_{\zeta,k}, s_{\eta,k}) \in \mathcal{D}_k^{\mathtt{rprt}(q)}$,* (14) *reaches its maximum for $s_{\eta,k} = \overline{s}_{\eta,k}^{\mathtt{rprt}(q)}$. For $\kappa_\Gamma^{\mathtt{C}} < 0$,* (14) *reaches its maximum for $s_{\eta,k} = \underline{s}_{\eta,k}^{\mathtt{rprt}(q)}$.*

Proof 5 *From Lemma 4, it follows that* (14) *is monotonically increasing in $s_{\eta,k}$ for $\kappa_\Gamma^{\mathtt{C}} \geq 0$. Therefore, we select $\overline{s}_{\eta,k}^{\mathtt{rprt}(q)}$, which is the maximum lateral position in $\mathcal{D}_k^{\mathtt{rprt}(q)}$, for computing* (14)*. Similarly, as* (14) *is monotonically decreasing in $s_{\eta,k}$ for $\kappa_\Gamma^{\mathtt{C}} < 0$, we select the minimum lateral coordinate $\underline{s}_{\eta,k}^{\mathtt{rprt}(q)}$ of $\mathcal{D}_k^{\mathtt{rprt}(q)}$ to compute* (14). □

The remaining task is to determine the curvature $\kappa_\Gamma^{\mathtt{C}}$, for which $\epsilon_\eta(s_{\eta,k}, \kappa_\Gamma^{\mathtt{C}})$ reaches its maximum for a given $\mathcal{D}_k^{\mathtt{rprt}(q)}$ and reference path $\Gamma(s_\zeta)$.

Lemma 6 (14) *is monotonically increasing in $|\kappa_\Gamma^{\mathtt{C}}|$.*

Proof 6 *We substitute $1/|\kappa_\Gamma^{\mathtt{C}}|$ with $r_\Gamma^{\mathtt{C}}$ in* (14) *and compute the partial derivative of* (14) *with respect to $r_\Gamma^{\mathtt{C}}$:*

$$\frac{\partial \epsilon_\eta}{\partial r_\Gamma^{\mathtt{C}}}(s_{\eta,k}, r_\Gamma^{\mathtt{C}}) = \begin{cases} -1 + \frac{r_\Gamma^{\mathtt{C}} - s_{\eta,k}}{\sqrt{(r_\Gamma^{\mathtt{C}} - s_{\eta,k})^2 + \ell^2}}, & \textit{for } \kappa_\Gamma^{\mathtt{C}} \geq 0, \\ -1 + \frac{r_\Gamma^{\mathtt{C}} + s_{\eta,k}}{\sqrt{(r_\Gamma^{\mathtt{C}} + s_{\eta,k})^2 + \ell^2}}, & \textit{for } \kappa_\Gamma^{\mathtt{C}} < 0. \end{cases} \tag{16}$$

Since $\frac{\partial \epsilon_\eta}{\partial r_\Gamma^{\mathtt{C}}}(s_{\eta,k}, r_\Gamma^{\mathtt{C}}) < 0$, (14) *is monotonically decreasing in $r_\Gamma^{\mathtt{C}}$. As a result,* (14) *is monotonically increasing in $|\kappa_\Gamma^{\mathtt{C}}|$, because $|\kappa_\Gamma^{\mathtt{C}}|$ is the reciprocal of $r_\Gamma^{\mathtt{C}}$.* □

Lemma 7 *For any $s_{\eta,k} \in \mathrm{proj}_{s_\eta}(\mathcal{D}_k^{\mathtt{rprt}(q)})$ and $\kappa_\Gamma^{\mathtt{C}} \in [\underline{\kappa}_\Gamma, \overline{\kappa}_\Gamma]$,* (14) *is bounded by*

$$\underline{\epsilon}_\eta = \begin{cases} \epsilon_\eta\left(\overline{s}_{\eta,k}^{\mathtt{rprt}(q)}, \overline{\kappa}_\Gamma\right), & \textit{for } \overline{\kappa}_\Gamma \geq 0, \\ 0 & \textit{for } \overline{\kappa}_\Gamma < 0, \end{cases} \tag{17}$$

$$\overline{\epsilon}_\eta = \begin{cases} \epsilon_\eta\left(\underline{s}_{\eta,k}^{\mathtt{rprt}(q)}, \underline{\kappa}_\Gamma\right), & \textit{for } \underline{\kappa}_\Gamma < 0, \\ 0 & \textit{for } \underline{\kappa}_\Gamma \geq 0. \end{cases} \tag{18}$$

Proof 7 *Since a straight line in the curvilinear coordinate system $F^{\mathtt{L}}$ is bent toward the inner side of the circle $\Gamma^{\mathtt{C}}$ when it is transformed to the Cartesian frame $F^{\mathtt{G}}$, we need to enlarge $\mathcal{D}_k^{\mathtt{rprt}(q)}$ only to the circle's outer side. For $\kappa_\Gamma^{\mathtt{C}} \geq 0$, we enlarge $\mathcal{D}_k^{\mathtt{rprt}(q)}$ in the negative η-direction with $\underline{\epsilon}_\eta = \epsilon_\eta\left(\overline{s}_{\eta,k}^{\mathtt{rprt}(q)}, \overline{\kappa}_\Gamma\right)$ (see Lemma 5 and 6). For $\kappa_\Gamma^{\mathtt{C}} < 0$, we enlarge $\mathcal{D}_k^{\mathtt{rprt}(q)}$ in the positive η-direction with $\overline{\epsilon}_\eta = \epsilon_\eta\left(\underline{s}_{\eta,k}^{\mathtt{rprt}(q)}, \underline{\kappa}_\Gamma\right)$ (see Lemma 5 and 6).* □

Theorem 1 *By computing $\mathcal{A}^{\epsilon}$ according to* (13)*,* (17)*, and* (18)*, Problem 1 is solved.*

Proof 8 *The proof follows directly from Lemma 3 and 7.* □

Acknowledgment

The authors thank BMW for supporting our driving experiments, particularly, Dr. Sebastian Gnatzig and Dr. Tobias Rehder for their assistance during our experiments. Moreover, the authors thank Vitaliy Rusinov for his contribution to fast collision checking in the reachability analysis.

References

[1] S. Bhattacharya and R. Ghrist, "Path homotopy invariants and their application to optimal trajectory planning," *Annals of Mathematics and Artificial Intelligence*, vol. 84, pp. 139–160, 2018.

[2] J. Park, S. Karumanchi, and K. Iagnemma, "Homotopy-based divide-and-conquer strategy for optimal trajectory planning via mixed-integer programming," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1101–1115, 2015.

[3] A. Varava, K. Hang, D. Kragic, and F. T. Pokorny, "Herding by caging: A topological approach towards guiding moving agents via mobile robots," in *Robotics: Science and Systems*, 2017, pp. 1–9.

[4] S. Bhattacharya, V. Kumar, and M. Likhachev, "Search-based path planning with homotopy class constraints," in *Proc. of the AAAI Conference on Artificial Intelligence*, 2010, pp. 1230–1237.

[5] P. Bender, Ö. S. Taş, J. Ziegler, and C. Stiller, "The combinatorial aspect of motion planning: Maneuver variants in structured environments." in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2015, pp. 1386–1392.

[6] C. Miller, C. Pek, and M. Althoff, "Efficient mixed-integer programming for longitudinal and lateral motion planning of autonomous vehicles," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 1954–1961.

[7] R. M. Karp, "On the computational complexity of combinatorial problems," *Networks*, vol. 5, no. 1, pp. 45–68, 1975.

[8] M. Althoff, "Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets," in *Hybrid Systems: Computation and Control*, 2013, pp. 173–182.

[9] S. Söntges and M. Althoff, "Computing the drivable area of autonomous road vehicles in dynamic road scenes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 6, pp. 1855–1866, 2018.

[10] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

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

[12] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[13] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," in *Proc. of the American Control Conference*, 2001, pp. 43–49.

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

[15] M. Werling, S. Kammel, J. Ziegler, and L. Gröll, "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *Int. Journal of Robotics Research*, vol. 31, no. 3, pp. 346–359, 2012.

[16] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2009, pp. 1879–1884.

[17] F. von Hundelshausen, M. Himmelsbach, F. Hecker, A. Mueller, and H.-J. Wuensche, "Driving with tentacles: Integral structures for sensing and motion," *Journal of Field Robotics*, vol. 25, no. 9, pp. 640–673, 2008.

[18] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 4889–4895.

[19] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2009, pp. 489–494.

[20] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *Proc. of the IEEE European Control Conference*, 2001, pp. 2603–2608.

[21] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for Bertha – a local, continuous method," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2014, pp. 450–457.

[22] F. Molinari, Nguyen Ngoc Anh, and L. Del Re, "Efficient mixed integer programming for autonomous overtaking," in *Proc. of the American Control Conference*, 2017, pp. 2303–2308.

[23] B. Yi, S. Gottschling, J. Ferdinand, N. Simm, F. Bonarens, and C. Stiller, "Real time integrated vehicle dynamics control and trajectory planning with MPC for critical maneuvers," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2016, pp. 584–589.

[24] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[25] D. P. Bertsekas, *Nonlinear Programming*, ser. Athena scientific optimization and computation series. Athena Scientific, 2016.

[26] A. Domahidi, E. Chu, and S. Boyd, "ECOS: An SOCP solver for embedded systems," in *Proc. of the IEEE European Control Conference*, 2013, pp. 3071–3076.

[27] J. Nilsson, M. Brännström, J. Fredriksson, and E. Coelingh, "Longitudinal and lateral control for automated yielding maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 5, pp. 1404–1414, 2016.

[28] W. Zhan, J. Chen, C.-Y. Chan, C. Liu, and M. Tomizuka, "Spatially-partitioned environmental representation and planning architecture for on-road autonomous driving," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 632–639.

[29] B. Gutjahr, L. Gröll, and M. Werling, "Lateral vehicle trajectory optimization using constrained linear time-varying MPC," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1586–1595, 2017.

[30] C. Pek and M. Althoff, "Computationally efficient fail-safe trajectory planning for self-driving vehicles using convex optimization," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2018, pp. 1447–1454.

[31] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2017, pp. 3530–3538.

[32] M. Riedmiller, M. Montemerlo, and H. Dahlkamp, "Learning to drive a real car in 20 minutes," in *Proc. of the IEEE Int. Conf. on Frontiers in the Convergence of Bioscience and Information Technologies*, 2007, pp. 645–650.

[33] P. Wolf, K. Kurzer, T. Wingert, F. Kuhnt, and J. M. Zöllner, "Adaptive behavior generation for autonomous driving using deep reinforcement learning with compact semantic states," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 993–1000.

[34] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *arXiv preprint arXiv:1610.03295*, pp. 1–13, 2016.

[35] S. A. Seshia, D. Sadigh, and S. S. Sastry, "Towards verified artificial intelligence," *arXiv preprint arXiv:1606.08514*, pp. 1–13, 2017.

[36] T. Gu, J. M. Dolan, and J. Lee, "Automated tactical maneuver discovery, reasoning and trajectory planning for autonomous driving," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2016, pp. 5474–5480.

[37] J. Schlechtriemen, K. P. Wabersich, and K.-D. Kuhnert, "Wiggling through complex traffic: Planning trajectories constrained by predictions," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2016, pp. 1293–1300.

[38] Z. Sun, D. Hsu, T. Jiang, H. Kurniawati, and J. H. Reif, "Narrow passage sampling for probabilistic roadmap planning," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1105–1115, 2005.

[39] H. Mohy-ud-Din and A. Muhammad, "Detecting narrow passages in configuration spaces via spectra of probabilistic roadmaps," in *Proc. of the ACM Symposium on Applied Computing*, 2010, pp. 1294–1298.

[40] K. Esterle, P. Hart, J. Bernhard, and A. Knoll, "Spatiotemporal motion planning with combinatorial reasoning for autonomous driving," in *Proc. of the IEEE Conference on Intelligent Transportation Systems*, 2018, pp. 1053–1060.

[41] Q. H. Do, S. Mita, and K. Yoneda, "Narrow passage path planning using fast marching method and support vector machine," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2014, pp. 630–635.

[42] H. Liu, F. Xiao, and C. Wang, "A predictive model for narrow passage path planner by using support vector machine in changing environments," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2015, pp. 2991–2996.

[43] M. Buechel, G. Hinz, F. Ruehl, H. Schroth, C. Gyoeri, and A. Knoll, "Ontology-based traffic scene modeling, traffic regulations dependent situational awareness and decision-making for automated vehicles," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 1471–1476.

[44] R. Kohlhaas, D. Hammann, T. Schamm, and J. M. Zöllner, "Planning of high-level maneuver sequences on semantic state spaces," in *Proc. of the IEEE Conference on Intelligent Transportation Systems*, 2015, pp. 2090–2096.

[45] L. Zhao, R. Ichise, T. Yoshikawa, T. Naito, T. Kakinami, and Y. Sasaki, "Ontology-based decision making on uncontrolled intersections and narrow roads," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2015, pp. 83–88.

[46] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, E. Kaus, R. G. Herrtwich, C. Rabe, D. Pfeiffer, F. Lindner, F. Stein, F. Erbs, M. Enzweiler, C. Knöppel, J. Hipp, M. Haueis, M. Trepte, C. Brenk, A. Tamke, M. Ghanaat, M. Braun, A. Joos, H. Fritz, H. Mock, M. Hein, and E. Zeeb, "Making Bertha drive – an autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014.

[47] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski,

B. Salesky, Y. Seo, S. Singh, J. Snider, A. Stentz, W. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, "Autonomous driving in urban environments: Boss and the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[48] S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagzent, J. Schröder, M. Thuy, M. Goebl, F. von Hundelshausen, O. Pink, C. Frese, and C. Stiller, "Team AnnieWAY's autonomous system for the 2007 DARPA Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 615–639, 2008.

[49] A. Furda and L. Vlacic, "Enabling safe autonomous driving in real-world city traffic using multiple criteria decision making," *IEEE Intelligent Transportation Systems Magazine*, vol. 3, no. 1, pp. 4–17, 2011.

[50] S. Ulbrich and M. Maurer, "Probabilistic online POMDP decision making for lane changes in fully automated driving," in *Proc. of the IEEE Conference on Intelligent Transportation Systems*, 2013, pp. 2063–2067.

[51] C. Hubmann, J. Schulz, G. Xu, D. Althoff, and C. Stiller, "A belief state planner for interactive merge maneuvers in congested traffic," in *Proc. of the IEEE Conference on Intelligent Transportation Systems*, 2018, pp. 1617–1624.

[52] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs," in *Proc. of the IEEE Conference on Intelligent Transportation Systems*, 2014, pp. 392–399.

[53] D. Lenz, T. Kessler, and A. Knoll, "Tactical cooperative planning for autonomous highway driving using Monte-Carlo Tree Search," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2016, pp. 447–453.

[54] S. Söntges and M. Althoff, "Computing possible driving corridors for automated vehicles," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 160–166.

[55] H. Ahn, K. Berntorp, and S. Di Cairano, "Reachability-based decision making for city driving," in *Proc. of the American Control Conference*, 2018, pp. 3203–3208.

[56] A. Shkolnik, M. Walter, and R. Tedrake, "Reachability-guided sampling for planning under differential constraints," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2009, pp. 2859–2865.

[57] S. Kousik, S. Vaskov, F. Bu, M. Johnson-Roberson, and R. Vasudevan, "Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots," *arXiv preprint arXiv:1809.06746*, pp. 1–58, 2018.

[58] M. Gerdts and I. Xausa, "Avoidance trajectories using reachable sets and parametric sensitivity analysis," in *IFIP Conf. on System Modeling and Optimization*, 2013, pp. 491–500.

[59] B. Schürmann, D. Heß, J. Eilbrecht, O. Stursberg, F. Köster, and M. Althoff, "Ensuring drivability of planned motions using formal methods," in *Proc. of the IEEE Conference on Intelligent Transportation Systems*, 2017, pp. 1–8.

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

[61] E. Héry, S. Masi, P. Xu, and P. Bonnifait, "Map-based curvilinear coordinates for autonomous vehicles," in *Proc. of the IEEE Conference on Intelligent Transportation Systems*, 2017, pp. 1–7.

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

[63] J. Ziegler and C. Stiller, "Fast collision checking for intelligent vehicle motion planning," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2010, pp. 518–522.

[64] H. T. Croft, K. Falconer, and R. K. Guy, *Unsolved problems in geometry: unsolved problems in intuitive mathematics*. Springer Science & Business Media New York, 1991, vol. 2.

[65] S. Magdici and M. Althoff, "Fail-safe motion planning of autonomous vehicles," in *Proc. of the IEEE Conference on Intelligent Transportation Systems*, 2016, pp. 452–458.

[66] W. Lipski and F. P. Preparata, "Finding the contour of a union of iso-oriented rectangles," *Journal of Algorithms*, vol. 1, no. 3, pp. 235 – 246, 1980.

[67] H. Edelsbrunner, J. Van Leeuwen, T. Ottmann, and D. Wood, "Computing the connected components of simple rectilinear geometrical objects in d-space," *RAIRO, Informatique théorique*, vol. 18, no. 2, pp. 171–183, 1984.

[68] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 719–726.

[69] M. Koschi and M. Althoff, "Set-based prediction of traffic participants considering occlusions and traffic rules," *IEEE Transactions on Intelligent Vehicles*, 2020, [in press].

[70] X. Qian, F. Altché, P. Bender, C. Stiller, and A. de La Fortelle, "Optimal trajectory planning for autonomous driving integrating logical constraints: An MIQP perspective," in *Proc. of the IEEE Conference on Intelligent Transportation Systems*, 2016, pp. 205–210.

[71] C. Burger and M. Lauer, "Cooperative multiple vehicle trajectory planning using MIQP," in *Proc. of the IEEE Conference on Intelligent Transportation Systems*, 2018, pp. 602–607.

[72] J. Eilbrecht and O. Stursberg, "Cooperative driving using a hierarchy of mixed-integer programming and tracking control," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 673–678.

[73] Gurobi Optimization, LLC, "Gurobi optimizer reference manual," 2020. [Online]. Available: http://www.gurobi.com

[74] J. Dahl, G. R. de Campos, C. Olsson, and J. Fredriksson, "Collision avoidance: A literature review on threat-assessment techniques," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 1, pp. 101–113, 2019.

[75] J. Pegna and F.-E. Wolter, "Surface curve design by orthogonal projection of space curves onto free-form surfaces," *Journal of Mechanical Design*, vol. 118, no. 1, pp. 45–52, 1996.

[76] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to interval analysis*. Society for Industrial and Applied Mathematics, 2009.

Stefanie Manzinger is currently a Ph.D. candidate and joined the Cyber-Physical Systems Group at the Technical University of Munich under Prof. Dr.-Ing. Matthias Althoff in 2016. She received her B.Sc. degree in Mechatronics and Information Technology in 2014 and her M.Sc. degree in Robotics, Cognition, Intelligence in 2016, both from the Technical University of Munich, Germany. Her research interests include automated vehicles, reachability analysis, and motion planning.

Christian Pek is a postdoctoral researcher in the Division of Robotics, Perception and Learning at KTH Royal Institute of Technology. Before joining KTH, he was a PhD student in the Cyber-Physical Systems Group at the Technical University of Munich under Prof. Dr.-Ing. Matthias Althoff. He was a research assistant in the motion planning group at BMW Group from 2015 until 2019. Christian graduated with the Master of Science degree in computer science and robotics from the Technical University of Braunschweig, Germany, and the University of Auckland, New Zealand, in 2015. His vision is a future of robots which robustly and safely accomplish tasks with and around humans.

Matthias Althoff is an associate professor in computer science at the Technical University of Munich, Germany. He received his diploma engineering degree in Mechanical Engineering in 2005, and his Ph.D. degree in Electrical Engineering in 2010, both from the Technical University of Munich, Germany. From 2010 to 2012 he was a postdoctoral researcher at Carnegie Mellon University, Pittsburgh, USA, and from 2012 to 2013 an assistant professor at Ilmenau University of Technology, Germany. His research interests include formal verification of continuous and hybrid systems, reachability analysis, planning algorithms, nonlinear control, automated vehicles, and power systems.

A.3 Computation of Solution Spaces for Optimization-based Trajectory Planning [3]

Summary This work represents a further development of our approach in Appendix A.2. We propose novel methods to determine driving corridors within reachable sets and to derive collision avoidance constraints from the corresponding driving corridors for optimization-based trajectory planning. In contrast to our previous approach in Appendix A.2, driving corridors identified in this work enable to plan the longitudinal and lateral motion of the autonomous vehicle simultaneously in a single optimization problem. To obtain collision avoidance constraints, we first over-approximate the complement of the driving corridor with convex keep-out zones. The number of convex keep-out zones can be defined a-priori, i.e., the number of collision avoidance constraints is known in advance, which is necessary, for example, when generating customized code for optimization-based trajectory planning. Based on the convex keep-out zones, we leverage state-of-the-art methods for determining non-convex collision avoidance constraints, but also provide methods to obtain convex collision avoidance constraints. Due to the generic formulation of collision avoidance constraints, our approach can be paired with various existing optimization-based trajectory planning methods. By using driving corridors, the initialization of nonlinear optimization problems is facilitated, local minima caused by obstacles in the environment are eliminated, and sets of goal states that the vehicle is supposed to reach can be considered.

We demonstrate the usefulness of our approach using scenarios from the CommonRoad benchmark suite. We combine our approach with two different state-of-the-art trajectory planning methods based on successive convexification. Our results show that in combination with our approach, feasible solutions can be found for traffic scenarios that previously could not be solved by the selected state-of-the-art trajectory planners. Our evaluations also show that our approach facilitates the initialization of trajectory planning problems, which leads to reduced computational effort and increased robustness against parameter variations.

Author Contributions **S. M.** initiated the idea of combined longitudinal and lateral trajectory planning by identifying driving corridors within reachable sets. L. S. developed the initial concept and algorithms. **S. M.** and L. S. revised and adjusted the concept and algorithms. **S. M.** and L. S. designed, conducted, and evaluated the experiments. **S. M.** and L. S. wrote the article. M. A. led the research project, provided feedback, and helped improving the manuscript.

Journal article The author's version of the article is reprinted in this thesis. The final version of record is available at `http://dx.doi.org/10.1109/TIV.2021.3077702`.

This article has been accepted for publication in IEEE Transactions on Intelligent Vehicles. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIV.2021.3077702

Computation of Solution Spaces for Optimization-Based Trajectory Planning

Lukas Schäfer*, Stefanie Manzinger*, and Matthias Althoff

***Abstract*—The nonlinear vehicle dynamics and the non-convexity of collision avoidance constraints pose major challenges for optimization-based trajectory planning of automated vehicles. Current solutions are either tailored to specific traffic scenarios, simplify the vehicle dynamics, are computationally demanding, or may get stuck in local minima. This work presents a novel approach to address the aforementioned shortcomings by identifying collision-free driving corridors that represent spatio-temporal constraints for motion planning using set-based reachability analysis. We derive a suitable formulation of collision avoidance constraints from driving corridors that can be integrated into arbitrary nonlinear programs as well as (successive) convexification procedures. When combining our approach with existing motion planning methods based on continuous optimization, trajectories can be planned in arbitrary traffic situations in a computationally efficient way. We demonstrate the efficacy of our approach using scenarios from the CommonRoad benchmark suite.**

I. Introduction

VARIATIONAL methods for trajectory planning of automated vehicles have gained increasing interest over the past years. While discrete motion planning methods [1]–[4] are specifically suited for exploration, they may struggle to find solutions in cluttered environments due to discretization effects. In contrast, optimization-based motion planning approaches do not suffer from discretization effects as trajectories are optimized in continuous space. However, the nonlinear vehicle dynamics and the non-convexity of the set of feasible positions generally lead to a high computational burden. Some approaches require additional guidance through the solution space [5], e.g., in the form of driving corridors that represent spatio-temporal constraints.

A. Related Work

To address the above-mentioned challenges, a variety of problem reformulations is proposed in the literature. We categorize them subsequently by the fidelity of their applied vehicle model, their techniques to identify driving corridors, and their formulation of collision avoidance constraints.

Manuscript received Month Date, Year; revised Month Date, Year. This work was supported by the Deutsche Forschungsgemeinschaft (German Research Foundation) within the Priority Programme SPP 1835 Cooperative Interacting Automobiles (grant number: AL 1185/4-2).

* These authors have contributed equally to this work.

The authors are with the Department of Informatics, Technical University of Munich, D-85748 Garching, Germany (e-mail: lukas.schaefer@tum.de, stefanie.manzinger@in.tum.de, althoff@in.tum.de)

1) Vehicle Dynamics: The fidelity of vehicle models for optimization-based trajectory planning ranges from very simple models [6]–[9], such as double-integrator dynamics, to rather complicated models [5], [10]–[13], such as dynamic single-track models. While the more complicated approaches demand excessive computational effort, less complicated approaches may fail in critical situations, because they neglect the non-holonomicity of the vehicle or decouple the longitudinal and lateral movement [6], [14]–[16].

2) Driving Corridors: Topological approaches for driving corridor identification have been studied for some time [12], [14], [17]–[25]. These approaches often exploit the concept of homotopy or homology to infer different maneuver variants [12], [22]–[25]. Basically, two trajectories are called homotopic if they can be continuously deformed into one another without intersecting any obstacle [26]. Topological approaches commonly decompose the collision-free regions in space-time into (convex) sub-regions: for each selected combination of sub-regions, an optimal trajectory can be planned. However, these approaches typically neglect the vehicle dynamics, and thus, cannot exclude non-drivable corridors prior to trajectory planning. Moreover, the number of distinct maneuver variants grows exponentially with the number of obstacles and some approaches are difficult to apply in dynamic environments [23], [24].

Another line of research finds driving corridors by inflating solutions from discrete motion planning methods, e.g., graph-based or sampling-based methods [27]–[33] or multi-agent partially observable Markov decision processes [34]. However, some approaches consider only static environments in their experiments [30]–[32] or only a discrete set of actions for computational tractability [34]. In general, these methods struggle in detecting narrow passageways in cluttered environments due to discretization effects.

In [35], a partitioning of the cluttered environment is obtained by means of convex lifting. The resulting partitioning of the space is utilized to obtain a reference path by graph search and to generate a driving corridor. The authors state that their algorithm can be extended to moving obstacles, but have not yet evaluated this.

Support vector machines (SVMs) have also been applied to identify driving corridors for path [36], [37] and trajectory planning [38], [39]. By assigning a passing side, i.e., either to the left or right, the SVM solver optimizes a separating surface to construct a collision-free driving corridor. However, deciding on the passing side is often the most crucial aspect.

3) Collision Avoidance Constraints: For optimization-based motion planning, obstacles are usually modeled as convex shapes, e.g., polytopes [40]–[42] or ellipsoids [43]–[46], where polytopes allow one to more freely specify shapes. More general non-convex obstacles can be handled by applying (semi-) convex decomposition techniques beforehand [47].

Given a proper representation of the obstacles, mixed-integer programming is often proposed to handle non-convex collision avoidance constraints [6], [7], [48], [49]. While mixed-integer programs guarantee global optimality of feasible solutions, their computational complexity is high [7]. Another line of research uses nonlinear programming [40], [42], [44]; however, these approaches strongly rely on a suitable initial guess and are also computationally expensive.

In contrast, convex optimization problems can be efficiently solved to global optimality [50] and real-time capable solvers are available, e.g., [51], [52]. To employ convex optimization techniques, the non-convex optimization problem is typically approximated by a single [15], [16], [53]–[56] or a sequence of convex optimization problems [5], [46], [57]–[62]. This usually requires extracting a convex subset from the non-convex set of feasible positions, which we discuss next.

In [5], [11], [15], [16], [53], [63], [64], the set of admissible (lateral) positions is described by an interval of admissible deviations from a reference path. However, collision avoidance can only be guaranteed for minor deviations of the longitudinal position from the initial guess [5], [11], [53] or the vehicle dynamics is decoupled [15], [16], [63]. This issue can be circumvented by restricting the feasible positions to lie within an ellipsoid [33], [35], [65], [66]. However, ellipsoidal constraints can result in rather conservative under-approximations of the feasible set of positions. More flexible polyhedral under-approximations can be obtained by linearizing signed-distance functions [58] or potential fields [55].

The works in [42], [67], [68] propose smooth reformulations of collision avoidance constraints for polyhedral obstacles. If the collision avoidance constraints are differentiable, a convex approximation can be obtained by directly linearizing the collision avoidance constraints [31], [57], [60], [61], [69]. However, an unsuitable initial guess might cause convergence to an infeasible local minimum [43].

To obtain a larger feasible set as compared to direct-linearization techniques, the works [46], [70], [71] propose projecting the current state of the system onto the boundary of a convex keep-out zone followed by linearizing the constraint at the projection point. The approaches in [41], [72] compute polyhedral inner-approximations by growing a collision-free ellipsoid and computing the tangents to the ellipsoid where it coincides with the boundary of an obstacle. A related idea is proposed in [54], where the edges of the polyhedron are sampled and a limiting obstacle is assigned to each edge. However, all these procedures have in common that they require a collision-free initial guess.

B. Contributions

We identify collision-free driving corridors within the reachable set of an automated vehicle. This work significantly differs from our previous work on motion planning with reachable sets [63] and proposes the following innovations:

- our novel method for obtaining collision-free driving corridors enables combined longitudinal and lateral trajectory planning;
- our collision avoidance constraints are created so that arbitrary gradient- and Hessian-based solvers as well as (successive) convexification procedures can be used for trajectory planning.

Our proposed approach offers the following benefits:

- generic formulation of collision avoidance, i.e., our approach can be embedded in a wide range of different optimization-based motion planning methods, independently of the fidelity of the vehicle model;
- simplified initialization, i.e., the driving corridors facilitate the search for a suitable initial guess for nonlinear optimization problems;
- elimination of local minima induced by obstacles, i.e., the driving corridors guide the optimization-based motion planner through the collision-free solution space;
- consideration of a set of goal states, i.e., driving corridors can be constrained to end in a set of terminal states, e.g., a specific goal region or standstill in safe areas;
- applicability in arbitrary traffic scenarios, i.e., our approach is capable of handling arbitrarily cluttered scenarios involving static and dynamic obstacles;
- the computational effort of our approach typically improves with the criticality of the scenario [63], i.e., with shrinking solution space for trajectory planning.

The remainder of this paper is structured as follows: Sec. II introduces the problem statement and the solution concept. The computation of reachable sets is briefly reviewed in Sec. III. Sec. IV elaborates on the identification of driving corridors, followed by the derivation of collision avoidance constraints in Secs. V and VI. Numerical results are provided in Sec. VII and our conclusions are drawn in Sec. VIII.

II. Problem Statement and Solution Concept

To precisely formulate our problem statement and for subsequent derivations, we introduce our notation. For a set $\mathcal{S}$, let $\mathcal{S}^{\mathrm{o}}$ denotes its interior, $\partial\mathcal{S}$ its boundary, $\mathcal{S}^{\complement}$ its complement, $\mathrm{conv}(\mathcal{S})$ its convex hull, and $\mathrm{cl}(\mathcal{S})$ its closure. For a hyperplane $H^{=}_{(a,b)} := \{x|a^Tx = b\}$, with $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$, let $H^{\leq}_{(a,b)}$ denote the corresponding halfspace $\{x|a^Tx \leq b\}$; analogously, $H^{\geq}_{(a,b)} := \{x|a^Tx \geq b\}$. If a set $\mathcal{S}$ is countable, its cardinality is denoted by $|\mathcal{S}|$. The set $\{r, r+1, \dots, q\} \subset \mathbb{N}_0, 0 \leq r \leq q$, is denoted by $\mathcal{I}_{[r:q]}$. A sequence $\mathcal{W}$ with components $\mathcal{W}_i,\ i \in \mathcal{I}_{[r:q]}$, is denoted by $\mathcal{W} = (\mathcal{W}_i)_{i=r}^{q}$. We introduce $\square$ as the placeholder for a variable where the minimum and maximum admissible values are denoted by $\underline{\square}$ and $\overline{\square}$, respectively. We further introduce a local curvilinear coordinate frame as F^{L} aligned with a given reference path $\Gamma : \mathbb{R} \to \mathbb{R}^2$. In F^{L}, a global position $(s_x, s_y)^T$ is expressed in terms of the arc length s_ζ and the orthogonal deviation s_η from $\Gamma(s_\zeta)$.

A. Problem Statement

Let us introduce the compact set of admissible states $\mathcal{X} \subset \mathbb{R}^{n_x}$ and the set of admissible control inputs $\mathcal{U} \subset \mathbb{R}^{n_u}$ of an automated vehicle, whose motions are governed by

$$x_{k+1} = f(x_k, u_k), \tag{1}$$

where $x_k \in \mathcal{X}$ is the state, $u_k \in \mathcal{U}$ the input, and $k \in \mathbb{N}_0$ is the discrete time step corresponding to the time $t_k = k\Delta t$, $\Delta t \in \mathbb{R}^+$. The dynamics in (1) are formulated in F^{L} and $f(x_k, u_k)$ is continuously differentiable on $\mathcal{X}$. Possible state and input trajectories of the system are denoted by $x(\cdot)$ and $u(\cdot)$, respectively. Next, we define a few important sets as well as the occupancy and projection operations:

Definition 1 (Occupancy)**.** *The operator* $\mathrm{occ}(x)$ *relates the state* $x \in \mathcal{X}$ *to the set of points in the position domain occupied by the automated vehicle as* $\mathrm{occ} : \mathcal{X} \to \mathbb{P}(\mathbb{R}^2)$*, where* $\mathbb{P}(\mathbb{R}^2)$ *denotes the power set of* $\mathbb{R}^2$*.*

Definition 2 (Set of Forbidden States)**.** *Given the set* $\mathcal{O}_k \subset \mathbb{R}^2$ *of occupied positions of all obstacles (e.g., other cars and pedestrians) including the space outside of the road, the set of forbidden states of the automated vehicle at time step k is*

$$\mathcal{F}_k := \{x \,|\, \mathrm{occ}(x) \cap \mathcal{O}_k \neq \emptyset\}\,.$$

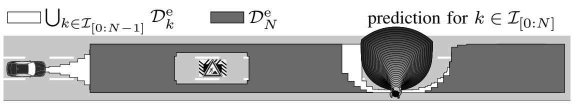
Definition 3 (One-Step Reachable Set)**.** *Let* $\mathcal{R}^{\mathrm{e}}_0 = \mathcal{X}_0$*, where* $\mathcal{X}_0$ *is the set of collision-free initial states of the automated vehicle including measurement uncertainties. The one-step reachable set* $\mathcal{R}^{\mathrm{e}}_{k+1}$ *is the set of all states that can be reached from the previous set of states* $\mathcal{R}^{\mathrm{e}}_k \subseteq \mathcal{X}$ *within one time step without intersecting* $\mathcal{F}_{k+1}$*:*

$$\mathcal{R}^{\mathrm{e}}_{k+1} := \Big\{x_{k+1} \in \mathcal{X} \,\Big|\, \exists x_k \in \mathcal{R}^{\mathrm{e}}_k,\, \exists u_k \in \mathcal{U}: \\ x_{k+1} = f(x_k, u_k) \,\wedge\, x_{k+1} \notin \mathcal{F}_{k+1}\Big\}.$$

Definition 4 (Projection)**.** *The operator* $\mathrm{proj}: \mathcal{X} \to \mathbb{R}^2$ *maps the state* $x \in \mathcal{X}$ *to the* (s_ζ, s_η) *plane:* $\mathrm{proj}(x) := (s_\zeta, s_\eta)^T$*. Using the same notation, we project a set of states* $\mathcal{X}$*:* $\mathrm{proj}(\mathcal{X}) := \{\mathrm{proj}(x) \,|\, x \in \mathcal{X}\}$*. Similarly, we use* $\mathrm{proj}_\zeta : \mathcal{X} \to \mathbb{R}$ *and* $\mathrm{proj}_\eta : \mathcal{X} \to \mathbb{R}$ *to map a state* $x \in \mathcal{X}$ *to the longitudinal or lateral position domain, respectively.*

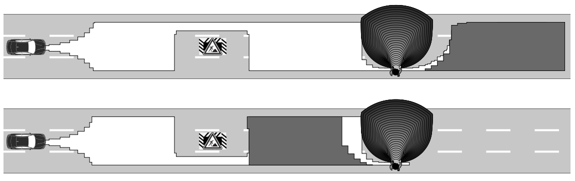
Definition 5 (Drivable Area)**.** *The drivable area* $\mathcal{D}^{\mathrm{e}}_k$ *at time step k is defined as* $\mathcal{D}^{\mathrm{e}}_k := \mathrm{proj}(\mathcal{R}^{\mathrm{e}}_k)$*.*

Our approach provides collision avoidance constraints for optimization-based trajectory planners. The input of our method is the current environment model that comprises the road network, the curvilinear coordinate system F^{L}, and all safety-relevant traffic participants, including their motion prediction. Our method is not tailored to any particular prediction method and only requires that $\mathcal{O}_k$ can be represented by the union of closed sets containing the future occupied positions of obstacles, including uncertainties. We aim to solve the following non-convex optimal control problem to plan the trajectory of the automated vehicle:

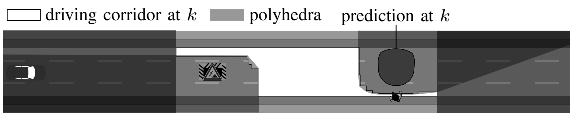


$\bigcup_{k\in\mathcal{I}_{[0:N-1]}} \mathcal{D}^{\mathrm{e}}_k$
$\mathcal{D}^{\mathrm{e}}_N$
prediction for $k \in \mathcal{I}_{[0:N]}$

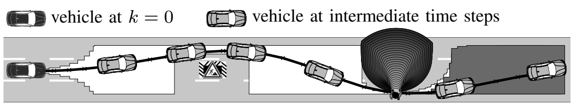
(a) Drivable area $\mathcal{D}^{\mathrm{e}}_k$ of the automated vehicle for consecutive time steps $k \in \mathcal{I}_{[0:N]}$. The drivable area at the final time step N is colored in gray.



(b) Two possible driving corridors are shown; for simplicity, we show the union of the driving corridors over time. The driving corridor at the final time step N is colored in gray.



(c) Approximation of the driving corridor complement with polyhedra depicted at a specific time step k.



(d) We extract collision avoidance constraints from the approximated driving corridor for the trajectory optimization.

Fig. 1: Illustration of the computation steps of our approach.

$$\min_{u(\cdot)} \sum_{k=0}^{N} J(x_k, u_k) \tag{2a}$$

such that

$$x_0 = \tilde{x}_0,\ x_N \in \mathcal{X}_{\mathrm{goal}}, \tag{2b}$$
$$\forall k \in \mathcal{I}_{[0:N-1]} : x_{k+1} = f(x_k, u_k), \tag{2c}$$
$$\forall k \in \mathcal{I}_{[0:N]} : \mathrm{proj}(x_k) \in \mathcal{D}^{\mathrm{e}}_k, \tag{2d}$$
$$g(x_k, u_k, k) \leq 0, \tag{2e}$$

where the cost function $J : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ is continuously differentiable. The (measured) initial state of the automated vehicle is denoted by $\tilde{x}_0$. Collision avoidance is encoded by the constraint (2d), i.e., the positions of the automated vehicle are limited to the drivable area. Additional constraints such as actuator constrains are summarized in the set of continuously differentiable, time-variant constraints $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{N}_0 \to \mathbb{R}^g$ in (2e).

B. Solution Concept

Our method determines a continuously differentiable approximation of the collision avoidance constraint (2d) in four steps (see Fig. 1):

1) We compute the drivable area (see Def. 5) of the automated vehicle in the current traffic scenario to explore

the dynamically reachable, collision-free solution space for trajectory planning (see Fig. 1a and Sec. III).
2) We extract dynamics-aware driving corridors from the drivable area (see Fig. 1b and Sec. IV).
3) We compute an approximation of the driving corridor complement using a fixed number of polyhedra (see Fig. 1c and Sec. V). This intermediate step allows us to use state-of-the-art methods for continuous trajectory optimization and to define the number of collision avoidance constraints in advance.
4) We derive collision avoidance constraints for the (non-) convex trajectory optimization from the approximation of the driving corridor complement (see Fig. 1d and Sec. VI).

These four steps are detailed in the subsequent sections.

III. Reachable Set Computation

Vehicle models used for motion planning usually possess nonlinear dynamics and a rather high-dimensional state space, which makes it difficult to calculate the reachable set [73]. We therefore aim to compute accurate approximations $\mathcal{R}$ of the exact reachable set $\mathcal{R}^{\mathrm{e}}$, i.e., $\mathcal{R} \approx \mathcal{R}^{\mathrm{e}}$, for computational efficiency. Among others, we realize this by approximating the vehicle dynamics by two second-order integrator models in the road-aligned coordinate system F^{L} [63]. The state $x = (s_\zeta, v_\zeta, s_\eta, v_\eta)^T$ and input $u = (a_\zeta, a_\eta)^T$ of the system are composed of the position s, velocity v, and acceleration a in the longitudinal ζ- and lateral η-directions, where both the velocity and acceleration are bounded:

$$\ddot{s}_\zeta(t) = a_\zeta(t), \quad \ddot{s}_\eta(t) = a_\eta(t), \tag{3a}$$
$$\underline{v}_\zeta \le v_\zeta(t) \le \overline{v}_\zeta, \quad \underline{v}_\eta \le v_\eta(t) \le \overline{v}_\eta, \tag{3b}$$
$$\underline{a}_\zeta \le a_\zeta(t) \le \overline{a}_\zeta, \quad \underline{a}_\eta \le a_\eta(t) \le \overline{a}_\eta. \tag{3c}$$

The approximation of the reachable set is represented as the union of base sets $\mathcal{R}_k^{(i)}$, $i \in \mathbb{N}_0$, i.e.,

$$\mathcal{R}_k^{\mathrm{e}} \approx \mathcal{R}_k := \cup_i \mathcal{R}_k^{(i)}, \tag{4}$$

where a base set $\mathcal{R}_k^{(i)}$ is the Cartesian product of two convex polytopes in the (s_ζ, v_ζ) and (s_η, v_η) plane [74]. The projection of a base set $\mathrm{proj}(\mathcal{R}_k^{(i)})$ yields an axis-aligned rectangle $\mathcal{D}_k^{(i)}$. The union of $\mathcal{D}_k^{(i)}$ approximates the drivable area: $\mathcal{D}_k^{\mathrm{e}} \approx \mathcal{D}_k := \cup_i \mathcal{D}_k^{(i)}$.

We use the algorithm presented in our previous works [63], [74] to compute the reachable set of the automated vehicle, which is briefly summarized below. To simplify the notation, we denote both the union and the collection of base sets $\mathcal{R}_k^{(i)}$ with $\mathcal{R}_k$; this is done analogously for the drivable area $\mathcal{D}_k$. The initial base set $\mathcal{R}_0^{(0)}$ results from the initial state $\tilde{x}_0$ to which the set of measurement uncertainties is added. The reachable set for consecutive time steps k is computed as follows:

1) Propagation: The polytopes of each base set $\mathcal{R}_k^{(i)}$ are propagated according to (3), which yields $\mathcal{R}_{k+1}^{\mathrm{prop}}$ and $\mathcal{D}_{k+1}^{\mathrm{prop}}$. At this stage, obstacles are not considered.

2) Removal of Forbidden States: We remove all colliding states from $\mathcal{R}_{k+1}^{\mathrm{prop}}$ to obtain $\mathcal{R}_{k+1}$. Since convex polytopes are not closed under set difference, we under-approximate $\mathcal{R}_{k+1}^{\mathrm{prop}} \setminus \mathcal{F}_{k+1}$ with several base sets $\mathcal{R}_{k+1}^{(j)}$.

To this end, we transform the set of obstacles $\mathcal{O}_k$ to the curvilinear coordinate frame F^{L} prior to the reachable set computation, and over-approximate the result with axis-aligned rectangles yielding the set $\tilde{\mathcal{O}}_k$. Efficient algorithms for axis-aligned rectangles benefit the remaining required steps:

a) Merging: Both sets $\mathcal{D}_{k+1}^{\mathrm{prop}}$ and $\tilde{\mathcal{O}}_{k+1}$ are merged using a sweep line algorithm [75], yielding rectilinear polygons.

b) Difference: By computing the difference between the rectilinear polygons of the propagated drivable area and the approximated obstacles, we obtain the collision-free reachable positions of the automated vehicle.

c) Partitioning: To cast the resulting collision-free reachable positions in the set representation defined in (4), we partition the set into rectangles $\mathcal{D}_{k+1}^{(j)}$ along the vertical η-direction, e.g., using a sweep line algorithm.

After we have obtained the collision-free drivable area $\mathcal{D}_{k+1}$, we determine the reachable velocities for $\mathcal{D}_{k+1}$ to obtain the reachable set $\mathcal{R}_{k+1}$ of the next time step. It is also possible to consider the shape of the automated vehicle for the removal of the forbidden states assuming that the automated vehicle is oriented along the reference path [63].

3) Update of Reachability Graph: As a consequence of the removal of forbidden states, multiple sets $\mathcal{R}_{k+1}^{(j)}$ can be reached from the same $\mathcal{R}_k^{(i)}$. For later use, we create a graph $\mathcal{G}_{\mathcal{R}}$, in which each node stores a base set $\mathcal{R}_k^{(i)}$ and its projection $\mathcal{D}_k^{(i)}$. An edge $(\mathcal{R}_k^{(i)}, \mathcal{R}_{k+1}^{(j)})$ is added if and only if $\mathcal{R}_k^{(i)}$ can reach $\mathcal{R}_{k+1}^{(j)}$ in one time step.

IV. Identification of Driving Corridors for Coupled Dynamics

The obtained reachable sets are generally non-convex and often disconnected due to the presence of obstacles. To obtain better manageable sets, we decompose the reachable set into driving corridors:

Definition 6 (Driving Corridor)**.** *A driving corridor $\mathcal{C} := (\mathcal{C}_k)_{k=0}^N$ is a sequence of sets $\mathcal{C}_k$ over time steps k that satisfy:*

C1) Reachability: $\mathcal{C}_k \subseteq \mathcal{R}_k$ and for each $\mathcal{R}_k^{(i)} \in \mathcal{C}_k$, there exists $\mathcal{R}_{k+1}^{(j)} \in \mathcal{C}_{k+1}$ such that $(\mathcal{R}_k^{(i)}, \mathcal{R}_{k+1}^{(j)}) \in \mathcal{G}_{\mathcal{R}}$;
C2) Goal states: $\mathcal{C}_N \subseteq \mathcal{X}_{\mathrm{goal}}$;
C3) Connectedness: $\mathrm{proj}(\mathcal{C}_k)$ is connected [76];
C4) Vertical convexity: any non-empty intersection of $\mathrm{proj}(\mathcal{C}_k)$ with a vertical line is connected [77].

A driving corridor may represent several maneuver options for each obstacle like yielding, passing, or following; nevertheless, the passing sides for obstacles are unique for all solutions in a driving corridor (see Fig. 1b). The different concepts of driving corridors and the related concept of homotopy are usually applied to obtain unique sequences of maneuvers [14], [22], [25], [34], [78]. In contrast, we have conceptualized driving corridors from an optimization point of view. By using driving corridors according to Def. 6 for trajectory planning, we are able to eliminate

1) the need for binary variables that encode collision avoidance constraints for multiple obstacles, since $\mathcal{C}_k$ is a single, connected set and already collision-free;
2) local minima due to obstacle constraints, since we assign a unique passing side for each obstacle prior to optimization (see Fig. 2).

Our approach to determine driving corridors $\mathcal{C}$ within the reachable set $\mathcal{R}$ is described next. For simplicity, we drop the projection operator when referring to $\mathcal{C}$ in the position domain.

1) Goal Region: We intersect the reachable set of the final time step with $\mathcal{X}_{\text{goal}}$, i.e., $\hat{\mathcal{R}}_N = \bigcup_i \mathcal{R}_N^{(i)} \cap \mathcal{X}_{\text{goal}}$. In case $\hat{\mathcal{R}}_N = \emptyset$, the desired goal states cannot be reached, which can be reported to a high-level planner. To remove states at earlier points in time that cannot reach $\mathcal{X}_{\text{goal}}$, we perform a graph search over $\mathcal{G}_{\mathcal{R}}$ backwards in time and remove base sets $\mathcal{R}_k^{(i)}$ that do not reach the set $\hat{\mathcal{R}}_N$.

2) Identification and Selection: We iterate over the drivable area backwards in time and decompose it into connected, vertically convex sets to extract driving corridors. As shown in Fig. 3b, the drivable area $\mathcal{D}_k$ is already vertically sliced (but not yet vertically convex nor connected) due to the removal of forbidden states. The decomposition into vertically convex sets is performed using the adjacency graph:

Definition 7 (Adjacency Graph—adapted from [23])**.** *The nodes of an adjacency graph $\mathcal{A}_k$ are sets $\mathcal{D}_k^{(i)} \in \mathcal{D}_k$ and an edge in $\mathcal{A}_k$ represents that $\mathcal{D}_k^{(i)}, \mathcal{D}_k^{(q)}$ share a border which is not just a vertex (e.g., $\mathcal{D}_k^{(0)}$ and $\mathcal{D}_k^{(1)}$ in Fig. 3b). The edges in $\mathcal{A}_k$ are directed in increasing longitudinal direction.*

By construction, $\mathcal{A}_k$ is a directed acyclic graph. Since there are no loops in $\mathcal{A}_k$, any path in $\mathcal{A}_k$ connecting a source with a sink node represents a connected, vertically convex set $\mathcal{V}_k^{(q)}$, as shown in Fig. 3b. For the identification and selection of driving corridors, we explore the driving corridor graph $\mathcal{G}_{\mathcal{C}}$ (see Fig. 3c):

Definition 8 (Driving Corridor Graph)**.** *The nodes of a driving corridor graph $\mathcal{G}_{\mathcal{C}}$ are vertically convex sets $\mathcal{V}_k^{(q)}$. An edge in $\mathcal{G}_{\mathcal{C}}$ represents that a set $\mathcal{D}_{k+1}^{(j)} \in \mathcal{V}_{k+1}^{(p)}$ is reachable from some $\mathcal{D}_k^{(i)} \in \mathcal{V}_k^{(q)}$ within one time step, i.e., $(\mathcal{D}_k^{(i)}, \mathcal{D}_{k+1}^{(j)}) \in \mathcal{G}_{\mathcal{R}}$.*

A path in $\mathcal{G}_{\mathcal{C}}$ represents a driving corridor $\mathcal{C}$. The exploration of $\mathcal{G}_{\mathcal{C}}$ can be performed using standard graph search algorithms like depth-first search or breadth-first search. The vertically convex sets $\mathcal{V}_N^{(q)}$ at the final time step N are sink nodes of $\mathcal{G}_{\mathcal{C}}$ and obtained through $\mathcal{A}_N$. The predecessors for a node $\mathcal{V}_k^{(q)}$ are obtained from the reachability graph $\mathcal{G}_{\mathcal{R}}$: we extract

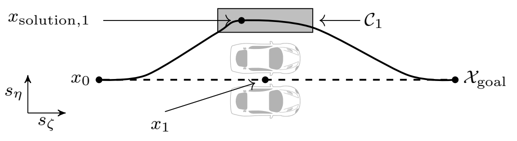
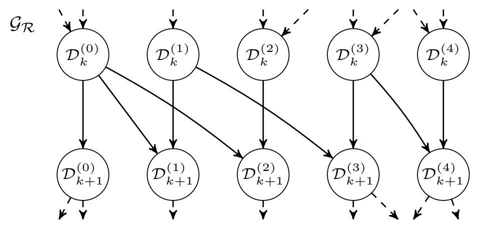
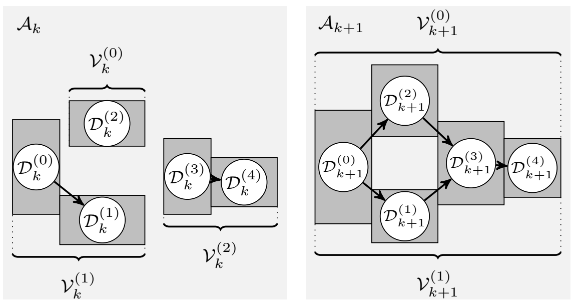


Fig. 2: Initialization (dashed line) in a (too) narrow gap close to an infeasible local minimum. Moving x_1 to either side will increase the violation of one of the constraints representing the obstacles. Using a driving corridor, a feasible solution (solid line) can be found since we rewrite the constraints as $x_1 \in \mathcal{C}_1$.

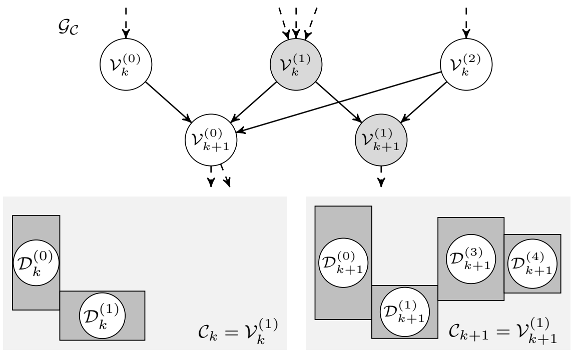


$\mathcal{G}_{\mathcal{R}}$
$\mathcal{D}_k^{(0)}$
$\mathcal{D}_k^{(1)}$
$\mathcal{D}_k^{(2)}$
$\mathcal{D}_k^{(3)}$
$\mathcal{D}_k^{(4)}$
$\mathcal{D}_{k+1}^{(0)}$
$\mathcal{D}_{k+1}^{(1)}$
$\mathcal{D}_{k+1}^{(2)}$
$\mathcal{D}_{k+1}^{(3)}$
$\mathcal{D}_{k+1}^{(4)}$

(a) Excerpt of the reachability graph $\mathcal{G}_{\mathcal{R}}$.

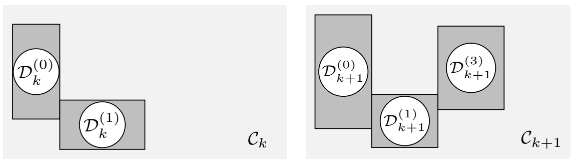


(b) A path in the adjacency graph $\mathcal{A}_k$ represents a connected, vertically convex set $\mathcal{V}_k^{(q)}$. $\mathcal{A}_k$ contains the sets $\mathcal{V}_k^{(0)} = \{\mathcal{D}_k^{(2)}\}$, $\mathcal{V}_k^{(1)} = \{\mathcal{D}_k^{(0)}, \mathcal{D}_k^{(1)}\}$, and $\mathcal{V}_k^{(2)} = \{\mathcal{D}_k^{(3)}, \mathcal{D}_k^{(4)}\}$. $\mathcal{A}_{k+1}$ contains the sets $\mathcal{V}_{k+1}^{(0)} = \{\mathcal{D}_{k+1}^{(0)}, \mathcal{D}_{k+1}^{(2)}, \mathcal{D}_{k+1}^{(3)}, \mathcal{D}_{k+1}^{(4)}\}$ and $\mathcal{V}_{k+1}^{(1)} = \{\mathcal{D}_{k+1}^{(0)}, \mathcal{D}_{k+1}^{(1)}, \mathcal{D}_{k+1}^{(3)}, \mathcal{D}_{k+1}^{(4)}\}$.



$\mathcal{G}_{\mathcal{C}}$
$\mathcal{V}_k^{(0)}$
$\mathcal{V}_k^{(1)}$
$\mathcal{V}_k^{(2)}$
$\mathcal{V}_{k+1}^{(0)}$
$\mathcal{V}_{k+1}^{(1)}$
$\mathcal{D}_k^{(0)}$
$\mathcal{D}_k^{(1)}$
$\mathcal{C}_k = \mathcal{V}_k^{(1)}$
$\mathcal{D}_{k+1}^{(0)}$
$\mathcal{D}_{k+1}^{(3)}$
$\mathcal{D}_{k+1}^{(4)}$
$\mathcal{D}_{k+1}^{(1)}$
$\mathcal{C}_{k+1} = \mathcal{V}_{k+1}^{(1)}$

(c) Excerpt of the driving corridor graph $\mathcal{G}_{\mathcal{C}}$. A path in $\mathcal{G}_{\mathcal{C}}$ represents a driving corridor $\mathcal{C}$. In our example, the selected driving corridor is $\mathcal{C}_k = \mathcal{V}_k^{(1)}$ and $\mathcal{C}_{k+1} = \mathcal{V}_{k+1}^{(1)}$ at time steps k and $k+1$, respectively.



(d) After selecting a driving corridor, the forward search removes the set $\mathcal{D}_{k+1}^{(4)}$ from $\mathcal{C}_{k+1}$, since it is not reachable from $\mathcal{C}_k$ according to $\mathcal{G}_{\mathcal{R}}$.

Fig. 3: Identification of driving corridors; we only depict nodes and edges that are relevant for our example. We explore the driving corridor graph $\mathcal{G}_{\mathcal{C}}$ backwards in time to determine a driving corridor. After selecting a driving corridor, a forward search from time step $k = 0$ to N is performed to remove sets $\mathcal{D}_k^{(i)} \in \mathcal{C}_k$ that are no longer reachable.

the union $\hat{\mathcal{D}}_{k-1}$ of parents for all $\mathcal{D}_k^{(i)} \in \mathcal{V}_k^{(q)}$ from the reachability graph $\mathcal{G}_\mathcal{R}$ and group them to vertically convex sets $\mathcal{V}_{k-1}^{(j)}$ by identifying paths in $\mathcal{A}_{k-1}$. The exploration of $\mathcal{G}_\mathcal{C}$ can be terminated as soon as a first driving corridor is found. If more time is available, the exploration of $\mathcal{G}_\mathcal{C}$ can be continued to determine alternative driving corridors. It is also possible to assign weights $w_{j,q}$ to the edges $(\mathcal{V}_{k-1}^{(j)}, \mathcal{V}_k^{(q)})$ to guide the selection of a driving corridor when using informed search techniques.

At each time step, creating $\mathcal{A}_k$ takes $\mathcal{O}(|\mathcal{R}_k|^2)$. A path representing a set $\mathcal{V}_k^{(q)}$ in $\mathcal{A}_k$ can be computed within $\mathcal{O}(|\mathcal{R}_k| + |E_k|)$ [79], where $|E_k|$ denotes the number of edges in $\mathcal{A}_k$ that are at most $\frac{1}{2}|\mathcal{R}_k|(|\mathcal{R}_k|-1)$. Let $r = \max_{k \in \mathcal{I}_{[0:N]}} |\mathcal{R}_k|$ and $e = \max_{k \in \mathcal{I}_{[0:N]}} |E_k|$. As we consider a time horizon of length N, the complexity of computing a first driving corridor is $\mathcal{O}(N(r+e))$ when using depth-first search.

3) Forward Search: Since we partition the predecessor sets $\hat{\mathcal{D}}_{k-1}$ of a set $\mathcal{V}_k^{(q)}$ again into vertically convex sets during the exploration of $\mathcal{G}_\mathcal{C}$, it may hold that certain sets $\mathcal{D}_k^{(i)} \in \mathcal{V}_k^{(q)}$ are not reachable when selecting a specific driving corridor in $\mathcal{G}_\mathcal{C}$. We therefore perform a forward search over the selected driving corridor: we iterate over all time steps starting from $k = 0$ and remove unreachable sets, see Fig. 3d. In the rare event that the driving corridor $\mathcal{C}_k$ becomes disconnected after the forward search due to approximation errors in the reachable set computation, we omit the corridor.

V. Polyhedral Approximation of the Driving Corridor Complement

Formulating collision avoidance as $\mathrm{proj}(x_k) \in \mathcal{C}_k$ is inherently difficult since the boundary of $\mathcal{C}_k$ is non-smooth and non-convex (see Fig. 4a). For example, $\mathrm{proj}(x_k) \in \mathcal{C}_k$ can be reformulated using disjunctive inequalities, where sets $\mathcal{D}_k^{(i)} \in \mathcal{C}_k$ are related by OR statements, i.e., $\mathrm{proj}(x_k) \in \mathcal{D}_k^{(0)} \vee \mathrm{proj}(x_k) \in \mathcal{D}_k^{(1)} \vee \ldots$ (see Fig. 4a), and thus, binary variables are required. As an alternative, an appropriate subset $\tilde{\mathcal{C}}_k \subset \mathcal{C}_k$ can be extracted, e.g., an ellipsoid within $\mathcal{C}_k$. However, most solutions for determining an appropriate set $\tilde{\mathcal{C}}_k$ require convex obstacles or convex decompositions of obstacles in the environment see, e.g., [41], [54], [70].

To exploit methods for smooth encodings of $\mathrm{proj}(x_k) \in \mathcal{C}_k$, see [42], [68], we approximate the complement $\mathcal{C}_k^\complement$ of the driving corridor $\mathcal{C}_k$ with polyhedral keep-out zones (see Fig. 4d). By limiting the number of polyhedral keep-out zones to $n_{\mathrm{max,k}}$, our approach facilitates real-time capability of optimization-based motion planning. The resulting polyhedral approximation $\tilde{\mathcal{C}}_k$ of $\mathcal{C}_k$ follows as:

Definition 9 (Polyhedral Approximation of $\mathcal{C}_k$)**.** *We define the polyhedral approximation $\tilde{\mathcal{C}}_k$ of $\mathcal{C}_k$ as the intersection of the closures of the complement of all polyhedral keep-out zones $\mathcal{P}_k^{\mathcal{C}(l)} \in \mathcal{P}_k^\mathcal{C}$, $|\mathcal{P}_k^\mathcal{C}| = n_{\mathrm{max,k}}$:*

$$\tilde{\mathcal{C}}_k := \bigcap_{l=1}^{n_{\mathrm{max,k}}} \mathrm{cl}(\mathbb{R}^2 \setminus \mathcal{P}_k^{\mathcal{C}(l)}).$$

We say $\mathrm{proj}(x_k) \in \tilde{\mathcal{C}}_k$ is a proper encoding of the collision avoidance constraint $\mathrm{proj}(x_k) \in \mathcal{C}_k$ if $\tilde{\mathcal{C}}_k \subseteq \mathcal{C}_k$, i.e., $\mathrm{proj}(x_k) \in \tilde{\mathcal{C}}_k \implies \mathrm{proj}(x_k) \in \mathcal{C}_k$. Below, we give an overview of the computation of $\mathcal{P}_k^\mathcal{C}$.

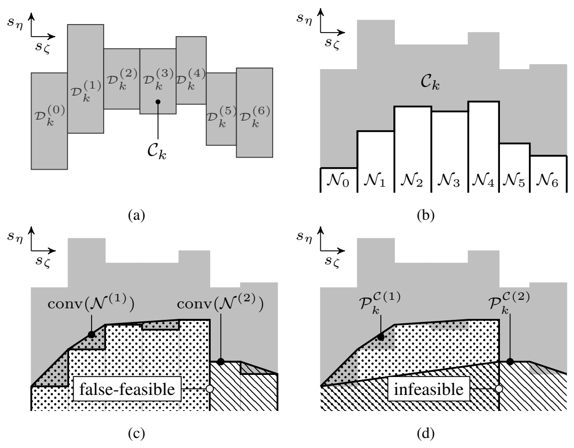

s_η
s_ζ
$\mathcal{D}_k^{(0)}$
$\mathcal{D}_k^{(1)}$
$\mathcal{D}_k^{(2)}$
$\mathcal{D}_k^{(3)}$
$\mathcal{D}_k^{(4)}$
$\mathcal{D}_k^{(5)}$
$\mathcal{D}_k^{(6)}$
$\mathcal{C}_k$
(a)
$\mathcal{C}_k$
$\mathcal{N}_0$ $\mathcal{N}_1$ $\mathcal{N}_2$ $\mathcal{N}_3$ $\mathcal{N}_4$ $\mathcal{N}_5$ $\mathcal{N}_6$
(b)
$\mathrm{conv}(\mathcal{N}^{(1)})$
$\mathrm{conv}(\mathcal{N}^{(2)})$
false-feasible
(c)
$\mathcal{P}_k^{\mathcal{C}(1)}$
$\mathcal{P}_k^{\mathcal{C}(2)}$
infeasible
(d)

Fig. 4: (a) Reformulating $\mathrm{proj}(x_k) \in \mathcal{C}_k$ using OR statements, i.e., $\mathrm{proj}(x_k) \in \mathcal{D}_k^{(0)} \vee \mathrm{proj}(x_k) \in \mathcal{D}_k^{(1)} \vee \ldots$, requires binary variables. (b) We reformulate $\mathrm{proj}(x_k) \in \mathcal{C}_k$ without binary variables by covering the complement of the driving corridor with polyhedral keep-out zones. An exact cover of $\mathcal{C}_k^\complement$ can be obtained with polyhedra $\mathcal{N}_j$. (c) To limit the number of polyhedral keep-out zones, we partition $\mathcal{N}$ into subsequences $\mathcal{N}^{(1)}$ and $\mathcal{N}^{(2)}$. By computing the convex hull of the subsequences, we obtain polyhedral keep-out zones. However, this encoding of $\mathrm{proj}(x_k) \in \mathcal{C}_k$ is incorrect, since all positions in the relative interior of the intersection of adjacent polyhedra are not excluded from the solution space. (d) To prevent false-feasibles, we modify $\mathrm{conv}(\mathcal{N}^{(1)})$ and $\mathrm{conv}(\mathcal{N}^{(2)})$ to obtain $\mathcal{P}_k^{\mathcal{C}(1)}$ and $\mathcal{P}_k^{\mathcal{C}(2)}$.

A. Overview

Since the drivable area is computed in the road-aligned coordinate frame F^L, we compute the cover of $\mathcal{C}_k^\complement$ in the longitudinal and lateral direction separately. The polyhedra bounding the driving corridor in the longitudinal direction are immediately obtained as

$$\mathcal{P}_k^{\mathcal{C}(n_{\mathrm{max,k}}-1)} := \{(s_{\zeta,k}, s_{\eta,k}) \in \mathbb{R}^2 \mid s_{\zeta,k} \geq \min\big(\mathrm{proj}_\zeta(\mathcal{C}_k)\big)\},$$
$$\mathcal{P}_k^{\mathcal{C}(n_{\mathrm{max,k}})} := \{(s_{\zeta,k}, s_{\eta,k}) \in \mathbb{R}^2 \mid s_{\zeta,k} \leq \max\big(\mathrm{proj}_\zeta(\mathcal{C}_k)\big\}.$$

An exact cover of $\mathcal{C}_k^\complement$ in the lateral direction can be obtained using a sequence $\mathcal{N} := (\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_M)$ of interiorly disjoint, orthogonal polyhedra $\mathcal{N}_j$ (see Fig. 4b); for brevity, we omit the time-dependency in the notation for $\mathcal{N}$. In most cases, this leads to the number of polyhedra being different from $n_{\mathrm{max,k}}$. We therefore partition $\mathcal{N}$ into $n_{\mathrm{max,k}}-2$ subsequences $\mathcal{N}^{(l)} \subseteq \mathcal{N}$, from which we obtain the remaining polyhedral keep-out zones in the lateral direction.

From $\mathcal{N}^{(l)}$, a polyhedron can be computed using the convex hull see, e.g., $\mathrm{conv}(\mathcal{N}^{(1)})$ and $\mathrm{conv}(\mathcal{N}^{(2)})$ in Fig. 4c. Since we consider the generic formulation of inequality constraints $g(x_k, u_k, k) \leq 0$ and adjacent polyhedra are interiorly disjoint, the entire boundary of every $\mathrm{conv}(\mathcal{N}^{(l)})$ is considered as collision-free. However, every point in the relative interior [50, Sec. 2.1.3] of the intersection of adjacent polyhedra is outside of $\mathcal{C}_k$ (see Fig. 4c). To properly encode $\mathrm{proj}(x_k) \in \mathcal{C}_k$, we modify adjacent $\mathrm{conv}(\mathcal{N}^{(l)})$ so that the resulting polyhedra

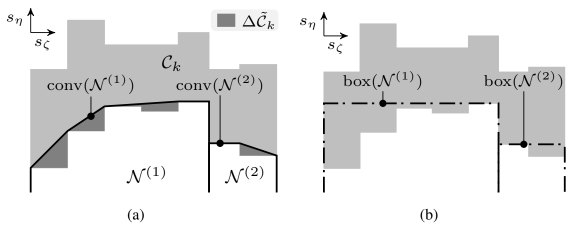

s_η
s_ζ
$\Delta\tilde{\mathcal{C}}_k$
$\mathcal{C}_k$
$\mathrm{conv}(\mathcal{N}^{(1)})$
$\mathrm{conv}(\mathcal{N}^{(2)})$
$\mathcal{N}^{(1)}$
$\mathcal{N}^{(2)}$
(a)
s_η
s_ζ
$\mathrm{box}(\mathcal{N}^{(1)})$
$\mathrm{box}(\mathcal{N}^{(2)})$
(b)

Fig. 5: (a) Approximation error $\Delta\tilde{\mathcal{C}}_k$. (b) We estimate the value of $\Delta\tilde{\mathcal{C}}_k$ by replacing the computation of $\mathrm{conv}(\mathcal{N}^{(l)})$ with $\mathrm{box}(\mathcal{N}^{(l)})$.

$\mathcal{P}_k^{\mathcal{C}(l)}$ overlap without further reducing the solution space compared to the union of $\mathrm{conv}(\mathcal{N}^{(l)})$ (see Fig. 4d). We have found empirically that the proposed approach facilitates convergence. Below, we discuss the partitioning of $\mathcal{N}$ into $n_{\mathrm{max,k}}-2$ subsequences, followed by explaining the algorithm to obtain the polyhedra $\mathcal{P}_k^{\mathcal{C}(l)}$, $l \in \mathcal{I}_{[1:n_{\mathrm{max,k}}-2]}$, for the lateral boundary.

B. Partitioning

We wish to partition the sequence $\mathcal{N}$ into $n_{\mathrm{max,k}}-2$ disjoint subsequences $\mathcal{N}^{(l)} \subseteq \mathcal{N}$ such that the approximation error $\Delta\tilde{\mathcal{C}}_k$, i.e, the loss of solution space of $\tilde{\mathcal{C}}_k$ compared to $\mathcal{C}_k$, is reduced (see Fig. 5a):

$$\Delta\tilde{\mathcal{C}}_k := \sum_{l=1}^{n_{\mathrm{max,k}}-2} \mathrm{area}\left(\mathrm{conv}(\mathcal{N}^{(l)}) \setminus \mathcal{N}^{(l)}\right). \tag{5}$$

Minimizing (5) leads to a combinatorial optimization problem that can become difficult to solve with increasing cardinality of $\mathcal{N}$. For computational efficiency, we propose several simplifications:

1) We use a greedy algorithm that recursively partitions $\mathcal{N}$ into subsequences $\mathcal{N}^{(l)}$.
2) We replace the costly computation of the convex hull in (5) with the computation of the minimum bounding box $\mathrm{box}(\mathcal{N}^{(l)})$ (see Fig. 5b).
3) At each recursion, we continue partitioning the sequence $\mathcal{N}^{(l)}$ that reduces $\Delta\tilde{\mathcal{C}}_k$ the most. If this partition is inadmissible because too many new subsequences would have to be created, i.e., the overall number of sequences exceeds $n_{\mathrm{max,k}}-2$, we always consider the next best one.
4) For each sequence $\mathcal{N}^{(l)}$, we consider only some of its possible partitions to evaluate the best partition. The heuristic used is described in Appendix A. This heuristic can be changed and is not the main focus of this paper.

Our algorithm can be extended to ensure the connectivity of $\tilde{\mathcal{C}}_k$ after computing the polyhedra from $\mathcal{N}^{(l)}$, $l \in \mathcal{I}_{[1:n_{\mathrm{max,k}}-2]}$: a partition is only admissible if $\mathrm{box}(\mathcal{N}^{(l)})^{\mathrm{o}}$ is pairwise disjoint for all $\mathcal{N}^{(j)}, j \neq l$, which can be efficiently validated.

C. Computation of the Polyhedra for the Lateral Direction

After determining the subsequences $\mathcal{N}^{(l)}$ as described above, we compute their corresponding polyhedral keep-out zones $\mathcal{P}_k^{\mathcal{C}(l)}$. Let us therefore introduce the polyhedron $\tilde{\mathcal{P}}_k^{(l)} \supseteq \mathrm{conv}(\mathcal{N}^{(l)})$ that is obtained by removing both halfspaces defining the vertical edges of $\mathrm{conv}(\mathcal{N}^{(l)})$, e.g., compare $\mathrm{conv}(\mathcal{N}^{(2)})$ in Fig. 4c with $\tilde{\mathcal{P}}_k^{(2)}$ in Fig. 6a. By intersecting $\tilde{\mathcal{P}}_k^{(l)}$ with halfspaces $H^{\leq,\leftarrow}_{(a_1,b_1)}$ and $H^{\leq,\rightarrow}_{(a_2,b_2)}$ that bound $\tilde{\mathcal{P}}_k^{(l)}$ in forward and backward driving direction, respectively, we obtain $\mathcal{P}_k^{\mathcal{C}(l)}$:

$$\mathcal{P}_k^{\mathcal{C}(l)} = \tilde{\mathcal{P}}_k^{(l)} \cap H^{\leq,\leftarrow}_{(a_1,b_1)} \cap H^{\leq,\rightarrow}_{(a_2,b_2)}. \tag{6}$$

To ensure a proper encoding of $\mathrm{proj}(x_k) \in \mathcal{C}_k$ as discussed in Sec. V-A, the halfspaces are created such that (a) $\mathcal{P}_k^{\mathcal{C}(l)} \supseteq \mathrm{conv}(\mathcal{N}^{(l)})$ and (b) the relative interior of the intersection of adjacent $\mathrm{conv}(\mathcal{N}^{(l)})$ is excluded from the solution space (see Fig. 4c and Fig. 4d). Additionally, we choose the halfspaces so that (c) the solution space is not further reduced compared to the union of $\mathrm{conv}(\mathcal{N}^{(l)})$ (see Fig. 4c and Fig. 4d).

1) Algorithm: To obtain the halfspace $H^{\leq,\leftarrow}_{(a_1,b_1)}$ limiting $\tilde{\mathcal{P}}_k^{(l)}$ in backward driving direction, we apply the following approach (the computation of $H^{\leq,\rightarrow}_{(a_2,b_2)}$ works analogously, see Fig. 6c and Fig. 6d): we initialize $H^{\leq,\leftarrow}_{(a_1,b_1)}$ as $\mathbb{R}^2$ and determine the set $\mathcal{E}^{\leftarrow}$ that contains the vertices of all $\mathrm{conv}(\mathcal{N}^{(j)})$, $j \neq l$, that are encountered when traversing along $\partial\mathcal{C}_k$ in backward driving direction starting from $\mathcal{N}^{(l)}$ (see Fig. 6a). To satisfy the conditions (a) and (c), we search for a halfspace $H^{\leq,\leftarrow}_{(a_1,b_1)}$, where the corresponding hyperplane $H^{=}_{(a_1,b_1)}$ separates $\mathrm{conv}(\mathcal{N}^{(l)})$ and $\mathcal{E}^{\leftarrow}$. The first support point of $H^{=}_{(a_1,b_1)}$ is chosen as the most backward vertex of $\mathrm{conv}(\mathcal{N}^{(l)})$. The second support point is selected from $\mathcal{E}^{\leftarrow}$. To this end, we iterate over the vertices $y \in \mathcal{E}^{\leftarrow}$: if y is contained in $\tilde{\mathcal{P}}_k^{(l)} \cap H^{\leq,\leftarrow}_{(a_1,b_1)}$, indicating an additional reduction of the solution space (see the left-most vertex in Fig. 6a), we update $H^{\leq,\leftarrow}_{(a_1,b_1)}$ with y as the second support point of $H^{=}_{(a_1,b_1)}$. As

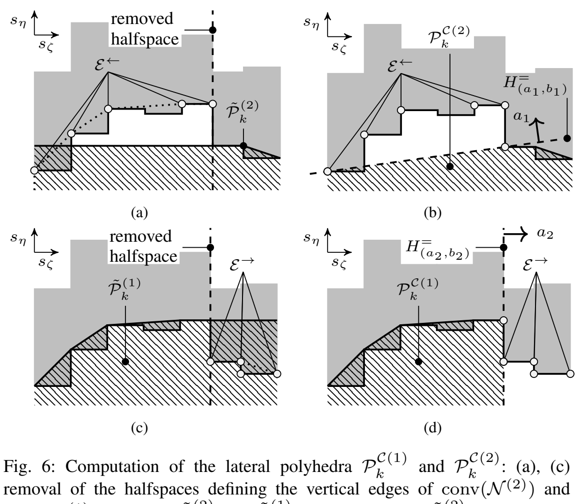

s_η
s_ζ
removed halfspace
$\mathcal{E}^{\leftarrow}$
$\tilde{\mathcal{P}}_k^{(2)}$
(a)
s_η
s_ζ
$\mathcal{P}_k^{\mathcal{C}(2)}$
$\mathcal{E}^{\leftarrow}$
$H^{=}_{(a_1,b_1)}$
a_1
(b)
s_η
s_ζ
removed halfspace
$\mathcal{E}^{\rightarrow}$
$\tilde{\mathcal{P}}_k^{(1)}$
(c)
s_η
s_ζ
$H^{=}_{(a_2,b_2)}$
a_2
$\mathcal{E}^{\rightarrow}$
$\mathcal{P}_k^{\mathcal{C}(1)}$
(d)

Fig. 6: Computation of the lateral polyhedra $\mathcal{P}_k^{\mathcal{C}(1)}$ and $\mathcal{P}_k^{\mathcal{C}(2)}$: (a), (c) removal of the halfspaces defining the vertical edges of $\mathrm{conv}(\mathcal{N}^{(2)})$ and $\mathrm{conv}(\mathcal{N}^{(1)})$ to obtain $\tilde{\mathcal{P}}_k^{(2)}$ and $\tilde{\mathcal{P}}_k^{(1)}$, respectively. (b) $\tilde{\mathcal{P}}_k^{(2)}$ is intersected with $H^{\leq,\leftarrow}_{(a_1,b_1)}$ yielding $\mathcal{P}_k^{\mathcal{C}(2)}$, where $H^{=}_{(a_1,b_1)}$ separates $\mathcal{P}_k^{\mathcal{C}(2)}$ from $\mathcal{E}^{\leftarrow}$ (backward direction). (d) $\mathcal{P}_k^{\mathcal{C}(1)}$ is obtained by intersecting $\tilde{\mathcal{P}}_k^{(1)}$ with $H^{\leq,\rightarrow}_{(a_2,b_2)}$, where $H^{=}_{(a_2,b_2)}$ separates $\mathcal{P}_k^{\mathcal{C}(1)}$ from $\mathcal{E}^{\rightarrow}$ (forward direction).

an example, compare $\tilde{\mathcal{P}}_k^{(2)}$ in Fig. 6a with the polyhedron $\mathcal{P}_k^{\mathcal{C}(2)}$ in Fig. 6b. To speed up the computations, it suffices to consider only the most backward and most forward vertex of each $\operatorname{conv}(\mathcal{N}^{(j)})$ due to convexity.

Let us now consider the adjacent polyhedra $\mathcal{P}_k^{\mathcal{C}(1)}$ and $\mathcal{P}_k^{\mathcal{C}(2)}$ in Fig. 4d. Computing $H^{\leq,\leftarrow}_{(a_1,b_1)}$ in the case of $\mathcal{P}_k^{\mathcal{C}(2)}$, yields a halfspace with non-vertical hyperplane $H^{=}_{(a_1,b_1)}$ (see Fig. 6b). When computing the halfspace $H^{\leq,\rightarrow}_{(a_2,b_2)}$ in the case of $\tilde{\mathcal{P}}_k^{(1)}$, the corresponding hyperplane $H^{=}_{(a_2,b_2)}$ is vertical (see Fig. 6c and Fig. 6d). Thus, all positions in the relative interior of the intersection of $\operatorname{conv}(\mathcal{N}^{(1)})$ and $\operatorname{conv}(\mathcal{N}^{(2)})$ are excluded from the solution space, i.e., condition (c) is satisfied. Since $\mathcal{C}_k$ is connected and vertically convex (see Def. 6), the same situation occurs for every pair of adjacent $\operatorname{conv}(\mathcal{N}^{(l)})$, and therefore, the polyhedral approximation $\tilde{\mathcal{C}}_k$ of $\mathcal{C}_k$ ensures a proper encoding of $\operatorname{proj}(x_k) \in \mathcal{C}_k$.

2) Complexity: The complexity of the computation of $\mathcal{P}_k^{\mathcal{C}(l)}$ is dominated by the computation of the convex hull of $\mathcal{N}^{(l)}$: Since any $\mathcal{N}_j$ has two vertices, the number of vertices of any $\mathcal{N}^{(l)}$ is bounded by $2|\mathcal{N}|$. Thus, the complexity of computing the convex hull with respect to $|\mathcal{N}|$ is $\mathcal{O}(|\mathcal{N}|\log(|\mathcal{N}|))$ [80]. If the vertices of $\mathcal{N}^{(l)}$ are stored in a list, removing the vertical edges takes $\mathcal{O}(|\mathcal{N}|)$. Computing $H^{\leq,\leftarrow}_{(a_1,b_1)}$ and $H^{\leq,\rightarrow}_{(a_2,b_2)}$ has complexity $\mathcal{O}(|\mathcal{N}|)$, since at most $2|\mathcal{N}|$ dot and vector products in $\mathbb{R}^2$ have to be evaluated. Computing the polyhedron $\mathcal{P}_k^{\mathcal{C}(l)}$ in halfspace representation can be done in $\mathcal{O}(|\mathcal{N}|)$ if the list of vertices of $\mathcal{P}_k^{\mathcal{C}(l)}$ is ordered. Thus, the overall complexity of the approach is $\mathcal{O}((n_{\max,k}-2)|\mathcal{N}|\log(|\mathcal{N}|))$.

VI. Integration Into Motion Planning Algorithms

Our approach can be integrated into a wide range of optimization-based motion planning algorithms, which is discussed next.

A. Duality-Based Reformulation

A state is collision-free if the signed distance with respect to every polyhedral keep-out zone is non-negative. However, non-convexity and non-differentiability of the signed-distance function prevent these constraints from being directly enforced [42]. To tackle this problem, Zhang et al. [42] propose a non-conservative and smooth reformulation of collision avoidance constraints for convex keep-out zones based on strong duality of convex optimization. Their results also enable one to find least-intrusive solutions in the case that a collision cannot be avoided. Below, we summarize the main results of [42], which allow us to integrate the collision avoidance constraints from driving corridors into arbitrary motion planning algorithms that rely on gradient- and Hessian-based solvers.

Consider a polyhedral keep-out zone $\mathcal{P}_k^{\mathcal{C}(l)} = \{y \,|\, y = \operatorname{proj}(x), A_k^{(l)} y - b_k^{(l)} \leq 0\}$ with matrix $A_k^{(l)}$ and vector $b_k^{(l)}$ of appropriate dimension. Using the results from [42], we encode the collision avoidance constraint $\operatorname{proj}(x_k) \notin \left(\mathcal{P}_k^{\mathcal{C}(l)}\right)^{\circ}$ as

$$(A_k^{(l)}\operatorname{proj}(x_k) - b_k^{(l)})^T \lambda_k^{(l)} \geq -\nu_k^{(l)} \tag{7a}$$

$$\|(A_k^{(l)})^T \lambda_k^{(l)}\|_2 = 1, \tag{7b}$$

$$\nu_k^{(l)} \geq 0, \lambda_k^{(l)} \geq 0, \tag{7c}$$

where the slack variable $\nu_k^{(l)} \in \mathbb{R}$ measures the penetration of $\mathcal{P}_k^{\mathcal{C}(l)}$, and $\lambda_k^{(l)}$ is the dual variable associated with the original constraint. The inequalities in (7c) apply element-wise. Since we aim to find least-intrusive solutions, the optimization problem (2) is rewritten as a soft-constrained problem by encoding the constraint (2d) using (7) and penalizing the penetration of all polyhedral keep-out zones, i.e.,

$$\min_{u(\cdot)} \sum_{k=0}^{N} J(x_k, u_k) + \kappa \sum_{k=1}^{N} \sum_{l=1}^{n_{\max,k}} \nu_k^{(l)}. \tag{8}$$

If the penalty weight $\kappa \in \mathbb{R}^+$ is chosen sufficiently large, a collision-free solution can be found if one exists [42], [81].

B. Convexification of Collision Avoidance Constraints

Since the collision-avoidance constraint $\operatorname{proj}(x_k) \in \tilde{\mathcal{C}}_k$ is usually non-convex due to the equality constraint (7b), we sketch a convexification procedure that is based on geometrical insight into (7). We project $\operatorname{proj}(x_k)$ onto each face of $\mathcal{P}_k^{\mathcal{C}(l)}$ separately and select the closest face. Subsequently, we determine a corresponding $\lambda_k^{(l)}$ that solves the constraints (7). Any such $\lambda_k^{(l)}$ provides a supporting hyperplane $H^{=}_{(n_k^{(l)},d_k^{(l)})}$, where $n_k^{(l)} = (A_k^{(l)})^T \lambda_k^{(l)}$ and $d_k^{(l)} = (\lambda_k^{(l)})^T b_k^{(l)}$, at the associated polyhedron $\mathcal{P}_k^{\mathcal{C}(l)}$ [42] [50, Sec. 8.1]. Thus, we obtain the convexified set of collision-free positions $\mathcal{P}_k^{\epsilon}$ as

$$\mathcal{P}_k^{\epsilon} := \bigcap_{\mathcal{P}_k^{\mathcal{C}(l)} \in \mathcal{P}_k^{\mathcal{C}}} H^{\geq}_{(n_k^{(l)},d_k^{(l)})}. \tag{9}$$

To obtain a $\lambda_k^{(l)}$ solving (7), we distinguish the following three cases:

C1) $\operatorname{proj}(x_k) \notin \mathcal{P}_k^{\mathcal{C}(l)}$: thus, we set $\nu_k^{(l)} = 0$ and solve the convex optimization problem $y^* = \arg\min_{y \in \mathcal{P}_k^{\mathcal{C}(l)}} \|y - \operatorname{proj}(x_k)\|_2^2$, see [50, Sec. 8.1]. The supporting hyperplane at $\mathcal{P}_k^{\mathcal{C}(l)}$ in y^* with normal vector in the direction of $\operatorname{proj}(x_k) - y^*$ implicitly defines a feasible $\lambda_k^{(l)}$.

C2) $\operatorname{proj}(x_k) \in \partial\mathcal{P}_k^{\mathcal{C}(l)}$: thus, we set $\nu_k^{(l)} = 0$. If $\operatorname{proj}(x_k)$ is not a vertex of $\mathcal{P}_k^{\mathcal{C}(l)}$, $\lambda_k^{(l)}$ follows as the canonical basis vector that extracts the corresponding hyperplane from $A_k^{(l)}$, $b_k^{(l)}$. If $\operatorname{proj}(x_k)$ is a vertex of $\mathcal{P}_k^{\mathcal{C}(l)}$, $\lambda_k^{(l)}$ is chosen so that $n_k^{(l)}$ is contained in the normal cone at $\mathcal{P}_k^{\mathcal{C}(l)}$ in $\operatorname{proj}(x_k)$. Moreover, the chosen $\lambda_k^{(l)}$ has to satisfy (7b) and (7c).

C3) $\operatorname{proj}(x_k) \in \left(\mathcal{P}_k^{\mathcal{C}(l)}\right)^{\circ}$: due to convexity of $\mathcal{P}_k^{\mathcal{C}(l)}$, $\operatorname{proj}(x_k)$ is not projected onto a vertex of $\mathcal{P}_k^{\mathcal{C}(l)}$. Thus, $\lambda_k^{(l)}$ is chosen as a canonical basis vector as in the first case of C2.

Note that the ambiguous cases in C2 and C3 do not admit a unique choice of $\lambda_k^{(l)}$ and, therefore, $H^{\geq}_{(n_k^{(l)},d_k^{(l)})}$. This might be resolved by comparing $n_k^{(l)}$ with the gradient of the cost function as for instance in [47, Sec. 3.3]. Since $\mathcal{P}_k^{\epsilon}$ is constructed from separating hyperplanes, see (9), it follows that $\operatorname{proj}(x_k) \in \mathcal{P}_k^{\epsilon} \subseteq \tilde{\mathcal{C}}_k$ if $\operatorname{proj}(x_k) \in \tilde{\mathcal{C}}_k$. While $\mathcal{P}_k^{\epsilon} \neq \emptyset$ holds if $\operatorname{proj}(x_k) \in \tilde{\mathcal{C}}_k$, $\mathcal{P}_k^{\epsilon}$ might be empty for $\operatorname{proj}(x_k) \notin \tilde{\mathcal{C}}_k$.

VII. Numerical Results

In Sec. VII-B, we demonstrate that our approach extracts driving corridors that represent semantically meaningful maneuver variants leading to a given goal region for the automated vehicle. In Sec. VII-C and VII-D, we embed our approach into two state-of-the-art motion planning algorithms. The results in Sec. VII-C show how our approach can enhance state-of-the-art planning algorithms so that local minima as in Fig. 2 are avoided. By applying our method to different scenarios from CommonRoad[1] [82] in Sec. VII-D, we demonstrate its applicability to arbitrary traffic scenarios and how the driving corridors can facilitate the search for a suitable initial guess. In Sec. VII-E we analyze the computation times of our approach, followed by a discussion on motion safety in Sec. VII-F. Since we have already shown that the computational effort of our approach typically improves with the criticality of the situation in our previous work [63], we focus on the new benefits of our approach in the evaluation.

A. Implementation Details

All computations were conducted on a laptop equipped with an Intel Core i7-10750H and 16 GB of memory. Our scenarios are modeled with the CommonRoad library. We consider a planning horizon of $5\,\mathrm{s}$ with $\Delta t = 0.1\,\mathrm{s}$ unless otherwise stated. Further implementation details are given below.

1) Driving Corridor Identification: Our first strategy for driving corridor identification referred to as DC1 demonstrates the real-time capability of our approach. Starting from the largest goal set $\mathcal{V}_N^{(*)}$, we explore the corridor graph $\mathcal{G}_\mathcal{C}$ using a depth-first search and terminate exploration once the node at time step $k = 0$ is reached. To obtain a driving corridor with nonrestrictive collision avoidance constraints, we select the largest $\mathcal{V}_k^{(*)} \in \mathcal{A}_k$ at each time step for exploration.

Our second strategy referred to as DC2 demonstrates the ability of our approach to explore complex scenarios in an anytime fashion and is employed if not otherwise stated. Strategy DC2 explores $\mathcal{G}_\mathcal{C}$ starting from every $\mathcal{V}_N^{(q)}$ and terminates as soon as at least $n_\mathcal{C}$ driving corridors for each $\mathcal{V}_N^{(q)}$ are identified. Throughout this section, we set $n_\mathcal{C} = 10$. We first explore the largest $\mathcal{V}_k^{(*)} \in \mathcal{A}_k$. All remaining $\mathcal{V}_k^{(q)} \in \mathcal{A}_k$ are sorted by their dissimilarity with respect to $\mathcal{V}_k^{(*)}$ (i.e., we compare their sets $\mathcal{D}_k^{(i)}$) and their size. This sorting strategy favors exploring different maneuver variants that reach the same goal region.

In case of DC2, we add weights to the edges of $\mathcal{G}_\mathcal{C}$ where the weight for an edge $(\mathcal{V}_{k-1}^{(j)}, \mathcal{V}_k^{(q)})$ is chosen as $w_{j,q} = \frac{1}{\mathrm{area}(\mathcal{V}_k^{(q)})}$. For each goal set $\mathcal{V}_N^{(q)}$, we determine the driving corridor $\mathcal{C}^*$ with the greatest cumulative area and an alternative driving corridor that should preferably represent a different tactical maneuver, e.g., overtaking an obstacle to the right side instead of overtaking to the left side. Therefore, we sufficiently increase the weights of the edges from $\mathcal{C}^*$ (thus, it is costly to select those edges again), and subsequently repeat the search for the driving corridor with the greatest cumulative area.

[1]https://commonroad.in.tum.de/

2) Polyhedral Approximation: For the experiments, we use the following number of polyhedra $n_{\max,\mathrm{k}}$: if $k \leq 10$, $n_{\max,\mathrm{k}} = 4$; if $11 \leq k \leq 15$, $n_{\max,\mathrm{k}} = 5$; if $16 \leq k \leq 20$, $n_{\max,\mathrm{k}} = 6$; if $21 \leq k \leq 32$, $n_{\max,\mathrm{k}} = 7$; else $n_{\max,\mathrm{k}} = 8$.

3) Trajectory Optimization: The trajectory optimization problems are modeled with CVXPY-codegen [83], which is based on the CVXPY modeling language [84], using ECOS [51] as the backend solver. For the evaluation, we approximate the occupancy of the automated vehicle as a disc. The vehicle dynamics are modeled similarly to the kinematic bicycle model in [85], but we use jerk instead of the longitudinal acceleration as input. The vehicle parameters are taken from the CommonRoad library (vehicle ID: 2).

We use a weighted combination of the cost functions proposed by CommonRoad [82] for trajectory planning:

$$J = w_{LC} J_{LC} + w_V J_V + w_O J_O + w_A J_A + w_u (J_J + J_{SR}),$$

where the (lateral) position tracking error in J_{LC} refers to the center of a given target lane. The desired velocity v_{des} for the partial cost J_V is extracted from the goal states of the CommonRoad scenarios and set to a constant value. The desired orientations for the partial cost function J_O are obtained by propagating the initial longitudinal position of the automated vehicle along the center of the given target lane using the desired velocity v_{des}.

For successive convexification procedures, we limit the maximum number of iterations by 50; if no collision-free solution can be found in time that satisfies the convergence criteria, it is considered to be infeasible.

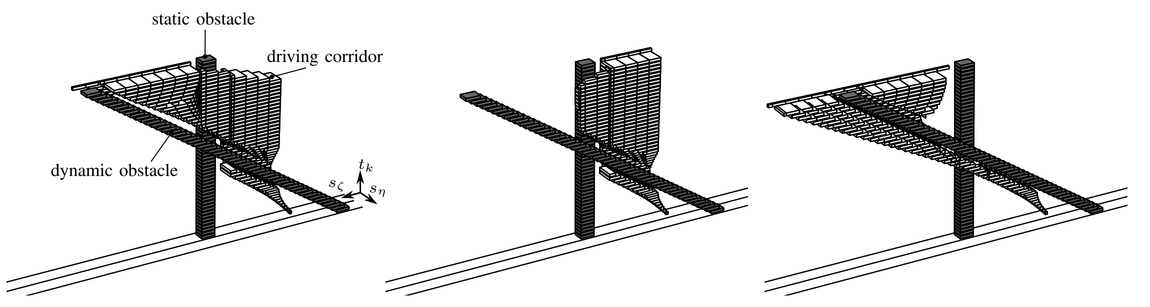
B. Driving Corridor Computation

We consider the scenario in Fig. 7a featuring a two-lane road with a static and a dynamic obstacle to determine driving corridors. The dynamic obstacle travels at a constant velocity of $17\,\mathrm{m/s}$. The goal for the automated vehicle is to return to its initial lane after $5\,\mathrm{s}$. We therefore intersect the drivable area of the automated vehicle at the final time step with the goal region.

Fig. 7 shows the results where the driving corridors are depicted as white boxes and the occupancies of the obstacles as gray boxes stacked over time. In this scenario, three semantically different maneuvers are found that lead to the desired goal region of the automated vehicle: (a) waiting until the dynamic obstacle has passed, and then, overtaking the static obstacle (see Fig. 7a); (b) staying behind the static obstacle (see Fig. 7b); and (c) passing the static obstacle before the dynamic obstacle (see Fig. 7c).

C. Augmenting Motion Planning Algorithms

In this section, we augment two state-of-the-art trajectory optimization algorithms with our proposed method. In the first experiment, we demonstrate that our method eliminates local minima occuring in optimization-based trajectory planning as illustrated in Fig. 2. The second experiment shows that our method facilitates the generalization of trajectory planners to on-road traffic situations.



(a) Waiting until the dynamic obstacle has passed, and then, overtaking the static obstacle.

(b) Staying behind the static obstacle.

(c) Passing the static obstacle before the dynamic obstacle.

Fig. 7: Driving corridors corresponding to three semantically different maneuvers. The slice of the drivable area at the top is intersected with the goal region.

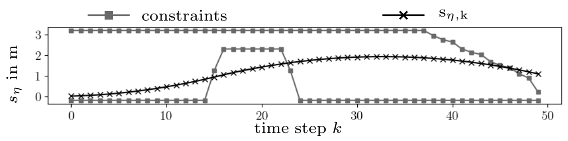
1) Elimination of Local Minima: We use the customized SQP algorithm proposed by [5] for our first experiment. At the beginning of each iteration of the successive convexification procedure in [5], the vehicle model is simulated using the input trajectory of the solution of the previous iteration (or the initial guess). The nonlinear trajectory optimization problem is then linearized around the forward-simulated trajectory to ensure dynamic feasibility of the solution.

The approach in [5] requires the passing sides of obstacles as input so that collision avoidance constraints can be represented as admissible intervals of lateral deviations $e_{\eta,k}$ from a reference path:

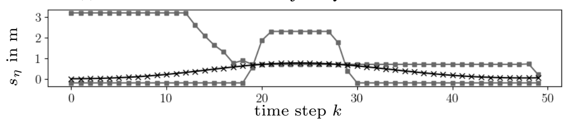
$$\underline{e}_{\eta,k}(\hat{s}_{\zeta,k}) \leq e_{\eta,k} \leq \overline{e}_{\eta,k}(\hat{s}_{\zeta,k}), \tag{10}$$

where $\hat{s}_{\zeta,k}$ denotes the longitudinal position of the forward-simulated trajectory. In simple scenarios one may argue that passing sides can be easily determined without driving corridors. However, the collision avoidance constraints in [5] are typically discontinuous with respect to the longitudinal position of the vehicle. For comparison, we therefore replace the original method for determining collision avoidance constraints with our proposed method that identifies driving corridors to obtain collision avoidance constraints, as shown in Sec. VI-B. We use the simple traffic scenario of the previous subsection and gradually raise the velocity of the dynamic obstacle starting from $10\,\mathrm{m/s}$ to $20\,\mathrm{m/s}$ in steps of $1\,\mathrm{m/s}$. For the original version of the algorithm in [5], we specify that the static obstacle must be passed on the left side and the dynamic obstacle on the right side. As an initial guess, we set all control inputs of the vehicle model to zero and simulate the dynamics forward in time.

For velocities of $10\,\mathrm{m/s}$ to $15\,\mathrm{m/s}$ of the dynamic obstacle, the customized SQP algorithm in [5] can find a collision-free trajectory. However, the method fails if the velocity of the dynamic obstacle is between $16\,\mathrm{m/s}$ to $20\,\mathrm{m/s}$ because the algorithm gets stuck in infeasible local minima. Fig. 8 shows the initial set of position constraints for the customized SQP algorithm as well as the set at convergence if the velocity of the dynamic obstacle is set to $17\,\mathrm{m/s}$. The empty set of



(a) Constraints and lateral trajectory at the first iteration



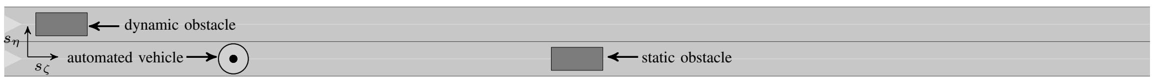
(b) Constraints and lateral trajectory at convergence

Fig. 8: Lateral position constraints used in the algorithm in [5]. Even though the initial guess in Fig. 8a seems to admit a collision-free trajectory, the algorithm converges to an infeasible local minimum shown in Fig. 8b.

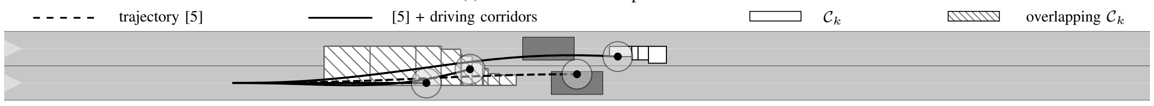
constraints for s_η in Fig. 8b indicates infeasibility.

In contrast to the original algorithm, the combination of the customized SQP algorithm with our computation of solution spaces provides a feasible trajectory for every identified driving corridor in each instance of the scenario. Fig. 9 provides a comparison of the results when the velocity of the dynamic obstacle is set to $17\,\mathrm{m/s}$. The trajectories that are obtained with our proposed method are depicted by the solid black lines in Fig. 9; the corresponding driving corridors are depicted in Fig. 7. Any of these found trajectories are feasible solutions of the nonlinear motion planning problem. The solution of the customized SQP algorithm in [5] is depicted by the dashed black line in Fig. 9; the infeasibility of the converged solution which was already indicated in Fig. 8b can be observed in Fig. 9b. In contrast, augmenting the algorithm from [5] with our approach yields collision-free trajectories as local minima induced by obstacles are eliminated.

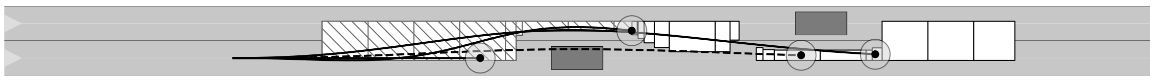
2) Generalization to Traffic Scenarios: To show that our method facilitates the generalization of trajectory planners to on-road traffic situations, we combine the zeroth order hold discretization scheme [86] with the successive convexification algorithm proposed in [62] that was initially developed



(a) Scenario at time step $k = 0$



(b) Trajectories at time step $k = 25$



(c) Trajectories at time step $k = 39$

Fig. 9: Comparison of the results using the customized SQP algorithm from [5] and the combination of the algorithm with our dynamics-aware solution spaces. The circles depict the occupancy of the automated vehicle at the current time step. In Fig. 9b, it can be observed that the algorithm from [5] does not find a collision-free solution, see the dashed line. In contrast, augmenting the algorithm with our approach yields a feasible solution for the three distinct maneuvers shown in Fig. 7. The regions where the driving corridors corresponding to Fig. 7a and Fig. 7b overlap are indicated by the striped white boxes.

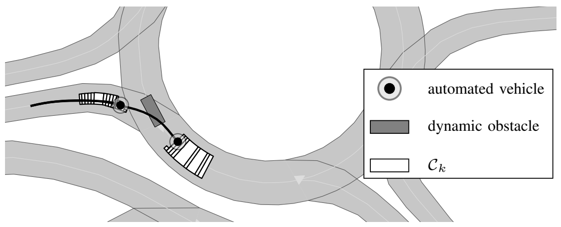
for powered descent guidance for extraterrestrial spaceflight. Instead of simulating the vehicle dynamics forward for linearization as in our previous experiment, [62] linearizes the nonlinear trajectory planning problem directly around the solution computed in the previous iteration of the successive convexification procedure. The resulting error in the vehicle dynamics constraint (2c) can be handled using slack variables which are referred to as virtual control inputs in [62].

Using a traffic scenario featuring a roundabout (see Fig. 10), we demonstrate that our method enables the application of the algorithm in [62] to traffic scenarios. The automated vehicle aims to enter the roundabout and take the second exit. There is an oncoming vehicle in the roundabout that must be taken into account. Since this maneuver requires longer time horizons so that the automated vehicle is able to reach the second exit, we use a planning horizon of 7 s. Our approach finds a trajectory for the automated vehicle both for the entry into the roundabout before and after the oncoming vehicle (see Fig. 10). In Sec. VII-D, we increase the difficulty of the planning problems and apply the planner to complex traffic scenarios with multiple obstacles and lanes.

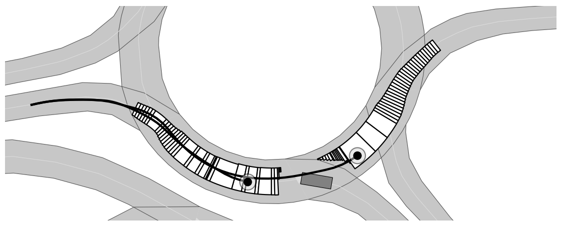
D. Applicability in Arbitrary Traffic Scenarios

Planning schemes for autonomous driving that iteratively linearize the vehicle model, such as [5], [12], usually rely on the previously computed solution as an initial guess, similar to model predictive control. In general, this initial guess is only close to the solution if the same maneuver is followed. However, if new maneuvers are initiated, implying a change to another driving corridor, the previously computed solution may be insufficient as an initial guess.

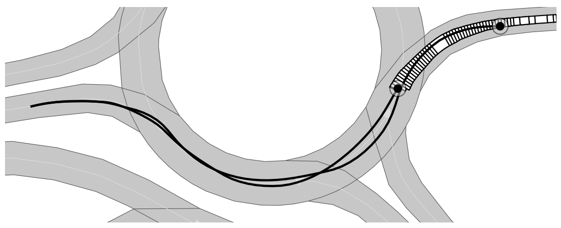
In this section, we demonstrate that driving corridors facilitate the initialization of trajectory planning methods and that our approach can be applied in arbitrary traffic scenarios with multiple obstacles. We therefore introduce different strategies for generating an initial guess for the optimization problems and compare their performance on 30 highway scenarios based on the NGSIM dataset from the CommonRoad library [82] that have five or six lanes and up to 57 obstacles (see Appendix B for the identifiers of the scenarios). After introducing different initialization strategies and the problem setup, we present the results of our experiments in Sec. VII-D3.



(a) Driving corridors and trajectories at time step $k = 21$



(b) Driving corridors and trajectories at time step $k = 47$



(c) Driving corridors and trajectories at time step $k = N = 70$

Fig. 10: Roundabout scenario: (a), (b) our approach identifies a driving corridor for the entry before and after the obstacle in the roundabout. (c) the automated vehicle reaches the second exit using either driving corridor.

1) Initialization Strategies: We introduce the *solution-space-guided initialization* that works as follows: first, the goal state is projected onto the driving corridor $\mathcal{C}_N$ at the terminal time step. Afterwards, we linearly interpolate between the projected goal state and the initial state of the automated vehicle. The collision avoidance constraints are convexified at each time step $k \in \mathcal{I}_{[1:N]}$ using the procedure in Sec. VI-B and we solve N convex optimization problems of the form $y_k^* = \arg\min_{y_k} \|y_k - \mathrm{proj}(x_k)\|_2^2$, s.t. $y_k \in \mathcal{P}_k^\epsilon$. The solutions y_k^* serve as our initial guess.

For comparison, we consider the following initialization strategies:

- *Simulate*: the vehicle dynamics are simulated forward in time assuming that all control inputs are zero.
- *Initial Lane*: we translate the initial state along the centerline of the initial lane according to the desired velocity v_{des} provided by the planning problem.
- *Interpolate*: we linearly interpolate between the initial state and the goal state, e.g., as in [60], [62].

2) Problem Setup: Since the considered initialization schemes except *simulate* generally do not provide a dynamically feasible initial guess, the motion planning algorithm of [5] introduced in Sec. VII-C1 cannot be applied. We therefore use the combination of [62], [86] with our approach as in Sec. VII-C2 and exploit the virtual control inputs that handle infeasible initializations. The goal state for the initialization schemes *solution-space-guided initialization* and *interpolate* are obtained by propagating the initial longitudinal position of the automated vehicle along the center of the target lane using the desired velocity v_{des}. To assess the robustness of the initialization schemes with respect to parameter variations, we choose the mean, lower, and upper bound from the uncertain goal states of the CommonRoad scenarios as desired velocities.

3) Evaluation: Using our method, we obtain 138 driving corridors for all traffic scenarios, in which we plan trajectories for each desired velocity v_{des}, i.e., we obtain 414 combinations of driving corridors and desired velocities. Tab. I summarizes the results, which we discuss below.

In total, we are able to find a feasible trajectory in each scenario for every initialization strategy. We further analyze the success rates of the initialization strategies with regard to all combinations of driving corridors and desired velocities (see the third column in Tab. I), i.e., the success rate is 100% if for each combination of driving corridors and desired velocities a solution is found. It can be seen that the *solution-space-guided initialization* solves more instances of the motion planning problem than any other scheme, as shown in Tab. I, and the success rate is increased by 6% compared to the *simulate* initialization strategy. Furthermore, in cases where the initial guess leads to a feasible solution, the number of convex programming iterations until convergence can be reduced by an average of almost 20% with the *solution-space-guided initialization* compared to *simulate*. Some combinations of driving corridors and desired velocities were infeasible due to (a) non-connectivity of the approximated driving corridor, which might yield an initial guess that switches between connected components, and (b) non-drivability of the driving corridor with regard to the high-fidelity vehicle model used for motion planning. Therefore, a possible direction for future research is the assessment of the drivability of the driving corridors prior to trajectory planning.

The desired velocity v_{des} affects the cost function J for the trajectory optimization and the initialization schemes *solution-space-guided initialization*, *interpolate*, and *initial lane*. We therefore analyze for each driving corridor the standard deviation in the required number of convex programming iterations resulting from variations in the desired velocity. Tab. I shows the standard deviations averaged over all scenarios for each initialization scheme. The average standard deviation using the *solution-space-guided initialization* is close to the case of *simulate* where the reference velocity only affects the cost function. In contrast, *interpolate* and *initial lane* exhibit much larger variations in the number of iterations when changing the desired velocity. Moreover, if *interpolate* or *initial lane* is used for initialization, it is more likely that a variation of the desired velocity will render the optimization task infeasible, as shown in the last column of Tab. I, which lists the number of driving corridors where a feasible solution could not be found for all desired velocities.

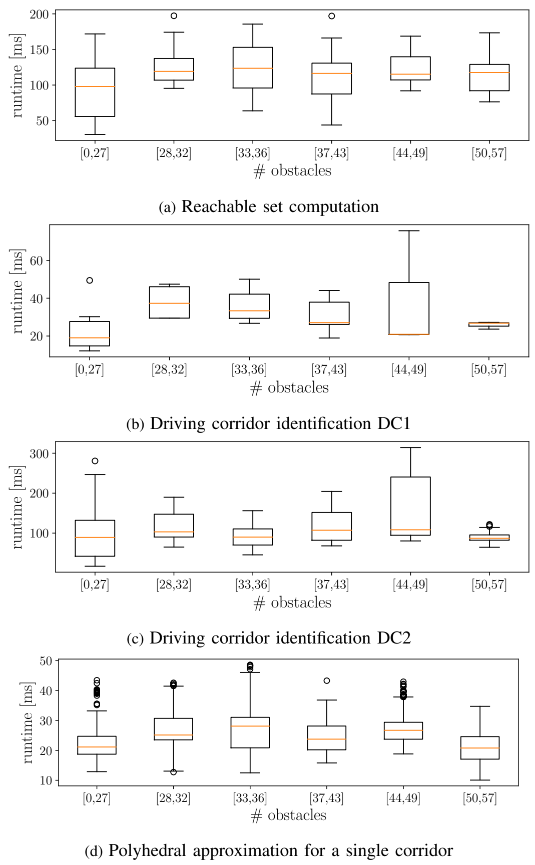
E. Computation Times

We analyze the runtime behavior of our approach in dependence of the number of obstacles in the scenarios. Fig. 11 shows boxplots of the resulting computation times for the reachable set computation, the driving corridor identification, and the computation of the polyhedral approximation, which we repeated 100 times to obtain a statistically profound example. The results shown in Fig. 11 indicate that our approach scales favorably with the number of obstacles as the median computation times vary only slightly; therefore, our method is suited to be employed in cluttered environments.

One iteration of the trajectory planning problem, i.e., convexifying the non-convex trajectory optimization problem (2) and solving the convexified optimization problem, requires 56 ms on average. Considering the average number of iterations given in Tab. I, the *solution-space-guided initialization* reduces the average computation time for motion planning to 347 ms compared to 426 ms for *simulate*. In conclusion, the solution-space-guided initialization provides an improvement in terms of the overall computational effort and increases robustness with respect to parameter variations only at a slightly increased effort to provide the initial guess.

F. Discussion on Motion Safety

To ensure motion safety, the automated vehicle must reason over an infinite time horizon (or at least until a set of safe goal states is reached) while considering its own dynamics and the future motion of other traffic participants according to [87]. Driving corridors can be restricted to end in a set of safe goal states to ensure persistent feasibility. The missing ingredients to ensure safety are (a) that the full-dimensional vehicle shape is considered for trajectory planning, (b) that the provided predictions must be conservative (i.e., guaranteed to include the real future behavior of other road users), and (c) robustness



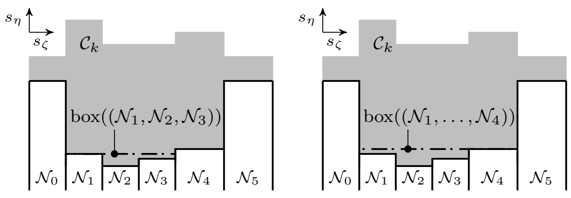
(a) Reachable set computation

(b) Driving corridor identification DC1

(c) Driving corridor identification DC2

(d) Polyhedral approximation for a single corridor

Fig. 11: Boxplots showing the runtime of our approach with respect to the number of obstacles in the scenarios listed in Appendix B. We clustered the scenarios with respect to the number of obstacles and repeated the computation of reachable sets, the identification of driving corridors, and the polyhedral approximation 100 times in each scenario.



(a) Largest sequence $(\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3)$ for $\mathcal{N}_1$.

(b) Largest sequence $(\mathcal{N}_1, \dots, \mathcal{N}_4)$ for $\mathcal{N}_4$.

Fig. 12: Visualization of the largest sequence $\mathcal{N}^{(i)}$ for a node $\mathcal{N}_j$ such that $\mathcal{N}_j$ limits the lateral extent of box$(\mathcal{N}^{(i)})$ towards the interior of the driving corridor $\mathcal{C}_k$.

with respect to disturbances and model uncertainties. Below, we will further discuss points (a) to (c).

In our earlier work in [63], a mathematically rigorous model for collision avoidance of the full-size vehicle in the special setup of decoupled longitudinal and lateral motion planning is given. Difficulties in generalizing the consideration of full-size vehicles arise from the fact that our approach is formulated in a road-aligned coordinate system and the assumption that the vehicle is oriented along the reference path when computing the reachable set (see Sec. III). The extension of our approach to formally correct collision-avoidance constraints for full-size vehicles in the road-aligned coordinate system is subject of future research.

Regarding (b) and (c), we suggest to integrate our approach in the online verification framework proposed in our previous works [88]–[90] to guarantee legal safety of automated vehicles and drivability of motions despite disturbances and model uncertainties. Moreover, as shown in [91], [92], our method can be extended to cooperative driving with explicit communication between groups of automated vehicles, which offers the possibility of further enhancing road safety due to reduced uncertainties regarding the intentions of others.

VIII. Conclusions

This paper provides a generalization of our previous results on combining set-based reachability analysis with optimal control by enabling the usage of vehicle models that jointly consider the longitudinal and lateral dynamics. Moreover, our approach can be combined with arbitrary existing optimization-based algorithms that rely on gradient- or Hessian-based solvers. Our results demonstrate that the proposed approach can identify different driving maneuvers in arbitrary traffic scenarios. Thereby, feasible solutions can be found in traffic scenarios that are not solvable using state-of-the-art planning algorithms. Apart from avoiding infeasible local minima, our method facilitates the generation of initial guesses for the trajectory planning problem such that the computational effort is reduced while increasing the robustness of the algorithms.

Appendix A
Heuristic for Partitioning

Considering all possible partitions of a sequence $\mathcal{N}^{(l)} \subseteq \mathcal{N}$ into subsequences can lead to a large computational overhead. Even if we only consider partitions of $\mathcal{N}^{(l)}$ into two

TABLE I: Comparison of Initialization Schemes

Initialization strategy	**Solutions**	**Success rate**	**Average # iterations**	**Avg. STD # iterations due to change of v_{des}**	**# Corridors that are not feasible for all v_{des}**
solution-space-guided	345	83%	6.2	0.55	2
interpolate	335	81%	8.6	1.44	11
simulate	317	77%	7.6	0.34	1
initial lane	326	79%	8.7	1.39	14

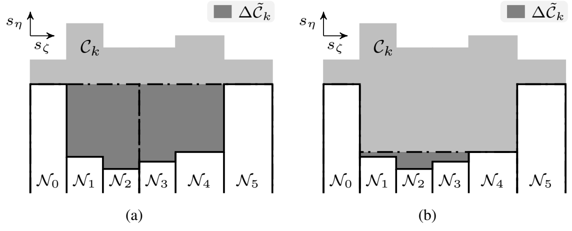


Fig. 13: (a) Any partition of $\mathcal{N}$ into two subsequences $\mathcal{N}^{(0)}$ and $\mathcal{N}^{(1)}$ cannot reduce $\Delta\tilde{\mathcal{C}}_k$ when using $\mathrm{box}(\mathcal{N}^{(l)})$ instead of $\mathrm{conv}(\mathcal{N}^{(l)})$ in (5). (b) Partitioning $\mathcal{N}$ into three subsequences, e.g., $(\mathcal{N}_0), (\mathcal{N}_1, \dots, \mathcal{N}_4)$, and $(\mathcal{N}_5)$, resolves this issue.

or three subsequences, we must already evaluate $(|\mathcal{N}^{(l)}| - 1) + \frac{(|\mathcal{N}^{(l)}|-2)(|\mathcal{N}^{(l)}|-1)}{2}$ possible partitions. Under this restriction, computing the partitions in the scenarios listed in Appendix B and using the number of polyhedra $n_{\max,\mathrm{k}}$ given in Sec. VII-A2 required $63\,\mathrm{ms}$ on average and at most $248\,\mathrm{ms}$.

We subsequently propose a heuristic that selects only $|\mathcal{N}^{(l)}|$ possible partitions of $\mathcal{N}^{(l)}$ for evaluation. These can be precomputed in $\mathcal{O}(|\mathcal{N}^{(l)}|^2)$, whereas storing and updating after each partitioning takes $\mathcal{O}(|\mathcal{N}^{(l)}|)$. The mean and maximum computation times decrease to less than 10 ms and 15 ms, respectively, in the scenarios listed in Appendix B. Compared to considering all possible combinations for partitions into two or three subsequences, our proposed heuristic yielded the same or even better results in terms of $\Delta\tilde{\mathcal{C}}_k$ in $84\,\%$ of the cases.

Let us introduce the approximation error of a set $\mathcal{N}'$ of sequences $\mathcal{N}^{(j)}$ as

$$\Delta(\mathcal{N}') := \sum_{\mathcal{N}^{(j)} \in \mathcal{N}'} \mathrm{area}\left(\mathrm{box}(\mathcal{N}^{(j)}) \setminus \mathcal{N}^{(j)}\right). \tag{11}$$

Without loss of generality, we assume that $\mathcal{N}^{(l)} = (\mathcal{N}_0, \dots, \mathcal{N}_n)$. For each $\mathcal{N}_j \in \mathcal{N}^{(l)}$, we determine the largest subsequence $\mathcal{N}^{(i)} = (\mathcal{N}_s, \dots, \mathcal{N}_j, \dots, \mathcal{N}_e)$, $0 \leq s \leq j \leq e \leq n$, such that $\mathcal{N}_j$ limits the lateral extent of $\mathrm{box}(\mathcal{N}^{(i)})$ towards the interior of the driving corridor $\mathcal{C}_k$, as shown in Fig. 12. The resulting partition is $\mathcal{N}' = \{(\mathcal{N}_0, \dots, \mathcal{N}_{s-1}), (\mathcal{N}_s, \dots, \mathcal{N}_e), (\mathcal{N}_{e+1}, \dots, \mathcal{N}_n)\}$. Following the reasoning in [93], we consider $\mathcal{N}'$ if the relative improvement of the approximation error is above a threshold ϵ:

$$\frac{\Delta(\{\mathcal{N}^{(l)}\}) - \Delta(\mathcal{N}')}{\mathrm{area}(\mathrm{box}(\mathrm{clip}(\mathcal{N}^{(l)})))} \geq \epsilon, \tag{12}$$

where $\mathrm{clip}(\cdot)$ clips all $\mathcal{N}_j \in \mathcal{N}^{(l)}$ so that $\mathrm{area}(\mathrm{clip}(\mathrm{box}(\mathcal{N}^{(l)}))) \in \mathbb{R}^+$ has a finite value. Using this heuristic, we can consider partitions of $\mathcal{N}^{(l)}$ into two or three subsequences. Note that we consider partitions into three subsequences, since it is possible that no partition into two subsequences can reduce the approximation error due to the replacement of the convex hull in (5) with the minimum bounding box (see Fig. 13).

Appendix B
Selected Scenarios for Evaluation in Sec. VII-D

The scenarios are USA_US101-1_1_T-1, USA_US101-*_3_T-1 for $* \in \{5, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26\}$, USA_US101-*_4_T-1 for $* \in \{4, 6, 15, 17, 26, 28\}$, USA_US101-*_5_T-1 for $* \in \{7, 9, 10, 11, 12, 27\}$, and USA_US101-*_6_T-1 for $* \in \{2, 3, 8, 13, 14\}$.

References

[1] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.
[2] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.
[3] M. Werling, S. Kammel, J. Ziegler, and L. Gröll, "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *Int. Journal of Robotics Research*, vol. 31, no. 3, pp. 346–359, 2012.
[4] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2009, pp. 1879–1884.
[5] A. Carvalho, Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "Predictive control of an autonomous ground vehicle using an iterative linearization approach," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2013, pp. 2335–2340.
[6] J. Nilsson, M. Brännström, J. Fredriksson, and E. Coelingh, "Longitudinal and lateral control for automated yielding maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 5, pp. 1404–1414, 2016.
[7] X. Qian, F. Altché, P. Bender, C. Stiller, and A. de La Fortelle, "Optimal trajectory planning for autonomous driving integrating logical constraints: An MIQP perspective," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2016, pp. 205–210.
[8] J. Tomas-Gabarron, E. Egea-Lopez, and J. Garcia-Haro, "Vehicular trajectory optimization for cooperative collision avoidance at high speeds," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1930–1941, 2013.
[9] D. N. Godbole, V. Hagenmeyer, R. Sengupta, and D. Swaroop, "Design of emergency manoeuvres for automated highway system: obstacle avoidance problem," in *Proc. of the IEEE Conference on Decision and Control*, vol. 5, 1997, pp. 4774–4779.
[10] C. Liu, S. Lee, S. Varnhagen, and H. E. Tseng, "Path planning for autonomous vehicles using model predictive control," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 174–179.
[11] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.
[12] B. Yi, P. Bender, F. Bonarens, and C. Stiller, "Model predictive trajectory planning for automated driving," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 1, pp. 24–38, 2019.
[13] P. Liu, B. Paden, and U. Ozguner, "Model predictive trajectory optimization and tracking for on-road autonomous vehicles," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2018, pp. 3692–3697.
[14] M. Schmidt, C. Wissing, J. Braun, T. Nattermann, and T. Bertram, "Maneuver identification for interaction-aware highway lane change behavior planning based on polygon clipping and convex optimization," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2019, pp. 3948–3953.
[15] B. Gutjahr, L. Gröll, and M. Werling, "Lateral vehicle trajectory optimization using constrained linear time-varying MPC," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1586–1595, 2017.
[16] C. Pek and M. Althoff, "Computationally efficient fail-safe trajectory planning for self-driving vehicles using convex optimization," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2018, pp. 1447–1454.
[17] F. Altché and A. de La Fortelle, "Partitioning of the free space-time for on-road navigation of autonomous ground vehicles," in *Proc. of the IEEE Conference on Decision and Control*, 2017, pp. 2126–2133.
[18] K. Esterle, P. Hart, J. Bernhard, and A. Knoll, "Spatiotemporal motion planning with combinatorial reasoning for autonomous driving," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2018, pp. 1053–1060.

[19] K. Esterle, V. Aravantinos, and A. Knoll, "From specifications to behavior: Maneuver verification in a semantic state space," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2019, pp. 2140–2147.

[20] J. Schlechtriemen, K. P. Wabersich, and K. Kuhnert, "Wiggling through complex traffic: Planning trajectories constrained by predictions," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2016, pp. 1293–1300.

[21] M. Schmidt, C. Manna, J. H. Braun, C. Wissing, M. Mohamed, and T. Bertram, "An interaction-aware lane change behavior planner for automated vehicles on highways based on polygon clipping," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1876–1883, 2019.

[22] P. Bender, Ö. Ş. Taş, J. Ziegler, and C. Stiller, "The combinatorial aspect of motion planning: Maneuver variants in structured environments," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2015, pp. 1386–1392.

[23] J. Park, S. Karumanchi, and K. Iagnemma, "Homotopy-based divide-and-conquer strategy for optimal trajectory planning via mixed-integer programming," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1101–1115, 2015.

[24] S. Kim, K. Sreenath, S. Bhattacharya, and V. Kumar, "Trajectory planning for systems with homotopy class constraints," in *Latest Advances in Robot Kinematics*, J. Lenarcic and M. Husty, Eds. Springer Netherlands, 2012, pp. 83–90.

[25] T. Gu, J. M. Dolan, and J. Lee, "Automated tactical maneuver discovery, reasoning and trajectory planning for autonomous driving," in *Proc. of the IEEE/RSJ Conference on Intelligent Robots and Systems*, 2016, pp. 5474–5480.

[26] S. Bhattacharya and R. Ghrist, "Path homotopy invariants and their application to optimal trajectory planning," *Annals of Mathematics and Artificial Intelligence*, vol. 84, pp. 139–160, 2018.

[27] W. Lim, S. Lee, M. Sunwoo, and K. Jo, "Hierarchical trajectory planning of an autonomous car based on the integration of a sampling and an optimization method," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 2, pp. 613–626, 2018.

[28] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *Int. Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[29] C. Rösmann, F. Hoffmann, and T. Bertram, "Planning of multiple robot trajectories in distinctive topologies," in *Proc. of the European Conference on Mobile Robots*, 2015, pp. 1–6.

[30] B. Li and Y. Zhang, "Fast trajectory planning for off-road autonomous driving with a spatiotemporal tunnel and numerical optimal control approach," in *Proc. of the IEEE Conference on Advanced Robotics and Mechatronics*, 2019, pp. 924–929.

[31] X. Liu, P. Lu, and B. Pan, "Survey of convex optimization for aerospace applications," *Astrodynamics*, vol. 1, no. 1, pp. 23–40, 2017.

[32] L. Campos-Macías, D. Gómez-Gutiérrez, R. Aldana-López, R. de la Guardia, and J. I. Parra-Vilchis, "A hybrid method for online trajectory planning of mobile robots in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 935–942, 2017.

[33] Z. Zhu, E. Schmerling, and M. Pavone, "A convex optimization approach to smooth trajectories for motion planning with car-like robots," in *Proc. of the IEEE Conference on Decision and Control*, 2015, pp. 835–842.

[34] W. Ding, L. Zhang, J. Chen, and S. Shen, "Safe trajectory generation for complex urban environments using spatio-temporal semantic corridor," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2997–3004, 2019.

[35] D. Ioan, S. Olaru, I. Prodan, F. Stoican, and S. Niculescu, "From obstacle-based space partitioning to corridors and path planning. A convex lifting approach," *IEEE Control Systems Letters*, vol. 4, no. 1, pp. 79–84, 2020.

[36] C. Qingyang, S. Zhenping, L. Daxue, F. Yuqiang, and L. Xiaohui, "Local path planning for an unmanned ground vehicle based on SVM," *Int. Journal of Advanced Robotic Systems*, vol. 9, no. 6, 2012.

[37] Q. H. Do, S. Mita, and K. Yoneda, "Narrow passage path planning using fast marching method and support vector machine," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2014, pp. 630–635.

[38] M. Morsali, J. Åslund, and E. Frisk, "Trajectory planning for autonomous vehicles in time varying environments using support vector machines," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 1–6.

[39] M. Morsali, J. Åslund, and E. Frisk, "Trajectory planning in traffic scenarios using support vector machines," *Proc. of the IFAC Symposium on Advances in Automotive Control*, vol. 52, no. 5, pp. 91–96, 2019.

[40] R. Soloperto, J. Köhler, F. Allgöwer, and M. A. Müller, "Collision avoidance for uncertain nonlinear systems with moving obstacles using robust model predictive control," in *Proc. of the European Control Conference*, 2019, pp. 811–817.

[41] R. Deits and R. Tedrake, *Computing Large Convex Regions of Obstacle-Free Space Through Semidefinite Programming*. Springer International Publishing, 2015, pp. 109–124.

[42] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, 2021.

[43] C. Zagaris, H. Park, J. Virgili-Llop, R. Zappulla, M. Romano, and I. Kolmanovsky, "Model predictive control of spacecraft relative motion with convexified keep-out-zone constraints," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 9, pp. 2054–2062, 2018.

[44] U. Rosolia, S. De Bruyne, and A. G. Alleyne, "Autonomous vehicle control: A nonconvex approach for obstacle avoidance," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 2, pp. 469–484, 2017.

[45] M. Szmuk, C. A. Pascucci, and B. Açıkmeşe, "Real-time quad-rotor path planning for mobile obstacle avoidance using convex optimization," in *Proc. of the IEEE/RSJ Conference on Intelligent Robots and Systems*, 2018, pp. 1–9.

[46] A. P. Vinod, S. Rice, Y. Mao, M. M. K. Oishi, and B. Açıkmeşe, "Stochastic motion planning using successive convexification and probabilistic occupancy functions," in *Proc. of the IEEE Conference on Decision and Control*, 2018, pp. 4425–4432.

[47] C. Liu, C.-Y. Lin, and M. Tomizuka, "The convex feasible set algorithm for real time optimization in motion planning," *SIAM Journal on Control and Optimization*, vol. 56, no. 4, pp. 2712–2733, 2018.

[48] C. Miller, C. Pek, and M. Althoff, "Efficient mixed-integer programming for longitudinal and lateral motion planning of autonomous vehicles," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 1954–1961.

[49] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *Proc. of the European Control Conference*, 2001, pp. 2603–2608.

[50] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[51] A. Domahidi, E. Chu, and S. Boyd, "ECOS: An SOCP solver for embedded systems," in *Proc. of the European Control Conference*, 2013, pp. 3071–3076.

[52] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.

[53] B. Yi, S. Gottschling, J. Ferdinand, N. Simm, F. Bonarens, and C. Stiller, "Real time integrated vehicle dynamics control and trajectory planning with MPC for critical maneuvers," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2016, pp. 584–589.

[54] C. Bali and A. Richards, "Robot navigation using convex model predictive control and approximate operating region optimization," in *Proc. of the IEEE/RSJ Conference on Intelligent Robots and Systems*, 2017, pp. 2171–2176.

[55] S. Dixit, U. Montanaro, M. Dianati, D. Oxtoby, T. Mizutani, A. Mouzakitis, and S. Fallah, "Trajectory planning for autonomous high-speed overtaking in structured environments using robust MPC," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 6, pp. 2310–2323, 2020.

[56] A. Britzelmeier and M. Gerdts, "A nonsmooth newton method for linear model-predictive control in tracking tasks for a mobile robot with obstacle avoidance," *IEEE Control Systems Letters*, vol. 4, no. 4, pp. 886–891, 2020.

[57] X. Liu and P. Lu, "Solving nonconvex optimal control problems by convex optimization," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 3, pp. 750–765, 2014.

[58] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *Int. Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

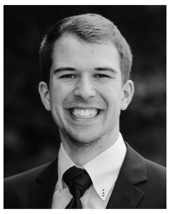
[59] Y. Mao, M. Szmuk, and B. Açıkmeşe, "Successive convexification of non-convex optimal control problems and its convergence properties," in *Proc. of the IEEE Conference on Decision and Control*, 2016, pp. 3636–3641.

[60] ——, "Successive convexification: A superlinearly convergent algorithm for non-convex optimal control problems," *arXiv preprint arXiv:1804.06539*, 2018.

[61] R. Bonalli, A. Cauligi, A. Bylard, and M. Pavone, "GuSTO: Guaranteed sequential trajectory optimization via sequential convex programming," in *Proc. of the International Conference on Robotics and Automation*, 2019, pp. 6741–6747.

[62] T. Reynolds, D. Malyuta, M. Mesbahi, B. Açıkmeşe, and J. M. Carson, "A real-time algorithm for non-convex powered descent guidance," in *AIAA Scitech 2020 Forum*, 2020.

[63] S. Manzinger, C. Pek, and M. Althoff, "Using reachable sets for trajectory planning of automated vehicles," *IEEE Transactions on Intelligent Vehicles*, 2020.
[64] S. M. Erlien, S. Fujita, and J. C. Gerdes, "Safe driving envelopes for shared control of ground vehicles," in *Proc. of the IFAC Symposium on Advances in Automotive Control*, vol. 46, no. 21, 2013, pp. 831–836.
[65] T. Schoels, L. Palmieri, K. O. Arras, and M. Diehl, "An NMPC approach using convex inner approximations for online motion planning with guaranteed collision avoidance," in *Proc. of the International Conference on Robotics and Automation*, 2020, pp. 3574–3580.
[66] T. Schoels, P. Rutquist, L. Palmieri, A. Zanelli, K. O. Arras, and M. Diehl, "CIAO*: MPC-based safe motion planning in predictable dynamic environments," vol. 53, no. 2, 2020, pp. 6555–6562.
[67] R. B. Patel and P. J. Goulart, "Trajectory generation for aircraft avoidance maneuvers using online optimization," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 1, pp. 218–230, 2011.
[68] M. Gerdts, R. Henrion, D. Hömberg, and C. Landry, "Path planning and collision avoidance for robots," *Numerical Algebra, Control and Optimization*, vol. 2, no. 3, pp. 437–463, 2012.
[69] D. Dueri, Y. Mao, Z. Mian, J. Ding, and B. Açıkmeşe, "Trajectory optimization with inter-sample obstacle avoidance via successive convexification," in *Proc. of the IEEE Conference on Decision and Control*, 2017, pp. 1150–1156.
[70] Y. Mao, D. Dueri, M. Szmuk, and B. Açıkmeşe, "Successive convexification of non-convex optimal control problems with state constraints," in *IFAC-PapersOnLine*, 2017, vol. 50, no. 1, pp. 4063–4069.
[71] C. Y. Son, D. Jang, H. Seo, T. Kim, H. Lee, and H. J. Kim, "Real-time optimal planning and model predictive control of a multi-rotor with a suspended load," in *Proc. of the International Conference on Robotics and Automation*, 2019, pp. 5665–5671.
[72] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.
[73] A. Platzer and E. M. Clarke, "Formal verification of curved flight collision avoidance maneuvers: A case study," in *Formal Methods*, A. Cavalcanti and D. R. Dams, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 547–562.
[74] S. Söntges and M. Althoff, "Computing the drivable area of autonomous road vehicles in dynamic road scenes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 6, pp. 1855–1866, 2018.
[75] W. Lipski and F. P. Preparata, "Finding the contour of a union of iso-oriented rectangles," *Journal of Algorithms*, vol. 1, no. 3, pp. 235–246, 1980.
[76] H. T. Croft, K. Falconer, and R. K. Guy, *Unsolved Problems in Geometry: Unsolved Problems in Intuitive Mathematics*, 1st ed., ser. Unsolved Problems in Intuitive Mathematics. Springer-Verlag New York, 1991, vol. 2.
[77] E. Fink and D. Wood, "Generalized halfspaces in restricted-orientation convexity," *Journal of Geometry*, vol. 62, pp. 99–120, 1998.
[78] S. Söntges and M. Althoff, "Computing possible driving corridors for automated vehicles," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 160–166.
[79] R. Sedgewick, *Algorithms in C, Part 5: Graph Algorithms, Third Edition*, 3rd ed. Addison-Wesley Professional, 2001.
[80] A. M. Andrew, "Another efficient algorithm for convex hulls in two dimensions," *Information Processing Letters*, vol. 9, no. 5, pp. 216–219, 1979.
[81] S.-P. Han and O. Mangasarian, "Exact penalty functions in nonlinear programming," *Mathematical Programming*, vol. 17, no. 1, pp. 251–269, 1979.
[82] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 719–726.
[83] N. Moehle, J. Mattingley, and S. Boyd, "Embedded convex optimization with CVXPY," Talk, 2017. [Online]. Available: https://www.nicholasmoehle.com/talks/codegen.pdf
[84] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.
[85] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2015, pp. 1094–1099.
[86] D. Malyuta, T. Reynolds, M. Szmuk, M. Mesbahi, B. Açıkmeşe, and J. M. Carson, "Discretization performance and accuracy analysis for the rocket powered descent guidance problem," in *AIAA Scitech 2019 Forum*, 2019.
[87] T. Fraichard, "A short paper about motion safety," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2007, pp. 1140–1145.
[88] C. Pek, S. Manzinger, M. Koschi, and M. Althoff, "Using online verification to prevent autonomous vehicles from causing accidents," *Nature Machine Intelligence*, vol. 2, no. 9, pp. 518–528, 2020.
[89] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 903–918, 2014.
[90] B. Schürmann, D. Heß, J. Eilbrecht, O. Stursberg, F. Köster, and M. Althoff, "Ensuring drivability of planned motions using formal methods," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2017, pp. 1–8.
[91] S. Manzinger and M. Althoff, "Negotiation of drivable areas of cooperative vehicles for conflict resolution," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2017, pp. 1–8.
[92] S. Manzinger and M. Althoff, "Tactical decision making for cooperative vehicles using reachable sets," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2018, pp. 444–451.
[93] M. Ghosh, N. M. Amato, Y. Lu, and J.-M. Lien, "Fast approximate convex decomposition using relative concavity," *Computer-Aided Design*, vol. 45, no. 2, pp. 494–504, 2013.



Lukas Schäfer is currently a M.Sc. student in Robotics, Cognition, Intelligence at the Technical University of Munich and was a student assistant in the Cyber-Physical Systems Group under Prof. Dr.-Ing. Matthias Althoff. He received his B.Eng. degree in Business Administration and Engineering from the Baden-Wuerttemberg Cooperative State University Stuttgart and his B.Sc. degree in Mechanical Engineering from the Technical University of Munich.

Stefanie Manzinger is a development engineer at MAN Truck & Bus SE, Germany. Before joining MAN Truck & Bus SE, she was a Ph.D. student in the Cyber-Physical Systems Group at the Technical University of Munich under Prof. Dr.-Ing. Matthias Althoff. She received her B.Sc. degree in Mechatronics and Information Technology in 2014 and her M.Sc. degree in Robotics, Cognition, Intelligence in 2016, both from the Technical University of Munich, Germany. Her research interests include automated vehicles, reachability analysis, and motion planning.

Matthias Althoff is an associate professor in computer science at the Technical University of Munich, Germany. He received his diploma engineering degree in Mechanical Engineering in 2005, and his Ph.D. degree in Electrical Engineering in 2010, both from the Technical University of Munich, Germany. From 2010 to 2012 he was a postdoctoral researcher at Carnegie Mellon University, Pittsburgh, USA, and from 2012 to 2013 an assistant professor at Ilmenau University of Technology, Germany. His research interests include formal verification of continuous and hybrid systems, reachability analysis, planning algorithms, nonlinear control, automated vehicles, and power systems.

A.4 Negotiation of Drivable Areas of Cooperative Vehicles for Conflict Resolution [5]

Summary We present a novel method for maneuver-based cooperation between communicating vehicles that is based on reachability analysis. Our approach is inspired by reservation-based conflict resolution algorithms, where vehicles reserve space on the road for exclusive access for a limited period of time. With our method, reservation requests for communicating vehicles participating in joint motion planning are determined by drivable areas, which are reachable sets projected onto the position domain. Areas which can be reached by at least two vehicles are in conflict. Conflicting areas are unambiguously reallocated to individual vehicles such that each vehicle has its unique area for motion planning.

Our method is applicable to mixed traffic scenarios where human drivers and autonomous vehicles co-exist. With our method, the search space for trajectory planning for each cooperative vehicle can be considerably reduced, since each trajectory must not leave the redistributed driving areas. In contrast to trajectory-based conflict resolution, our method offers a certain degree of flexibility in individual trajectory planning, since drivable areas are negotiated, i.e., we negotiate position constraints for trajectory planning.

The usefulness of our approach is demonstrated in traffic scenarios from the CommonRoad benchmark suite. The experiments demonstrate that high-level cooperative driving strategies are determined prior to trajectory planning by distributing shares of the drivable area. This study was our first step towards cooperative motion planning using reachable sets. The current redistribution of conflicting reachable sets only relies on geometric reasoning, so we have redesigned our approach in the publication in Appendix A.5 to take into account more general vehicle objectives.

Author Contributions **S. M.** initiated the idea of negotiating conflicting reachable sets among a group of cooperative vehicles so that each vehicle receives exclusive access to areas on the road for a limited time. **S. M.** developed the concept and algorithms. **S. M.** wrote the conference paper. **S. M.** designed, conducted, and evaluated the experiments. M. A. led the research project, provided feedback, and helped improving the manuscript.

Conference paper The author's version of the conference paper is reprinted in this thesis. The final version of record is available at `http://dx.doi.org/10.1109/ITSC.2017.8317765`.

Negotiation of Drivable Areas of Cooperative Vehicles for Conflict Resolution

Stefanie Manzinger and Matthias Althoff
Department of Informatics
Technische Universität München
Boltzmannstraße 3, 85748 Garching bei München, Germany
Email: {stefanie.manzinger, althoff}@in.tum.de

***Abstract*—We address the problem of cooperative conflict resolution for multi-vehicle motion planning in mixed-traffic scenarios, where automated and manually-driven vehicles co-exist. We propose a novel solution based on reachability analysis, which provides the drivable area of each collaborative traffic participant. Overlapping drivable areas are redistributed so that each traffic participant receives an individual area for motion planning. We do not stipulate a specific method for predicting the future motion of non-communicating traffic participants. Furthermore, uncertainties in the initial states of the cooperative vehicles, e.g. due to sensor noise, can be easily integrated. A byproduct of our approach is that collaborative groups can be automatically found by identifying conflicting drivable areas; if no conflict exists, collaboration becomes unnecessary. We demonstrate the redistribution of drivable areas with two numerical examples.**

I. Introduction

Collaborative motion planning of various automated road vehicles is clearly superior in terms of achievable safety and comfort compared to computing individual motion plans. This is because individual motion planning is a special case of collaborative planning when vehicles are not communicating. Many promising approaches for multi-vehicle motion planning have been developed; however, dealing with mixed-traffic situations and uncertainty is still an open research topic. We propose a unified approach for cooperative conflict resolution based on the computation of drivable areas where automated and manually-driven vehicles share the road. We first review literature concerning specific applications like intersection management and merging; after, we discuss priority-based, market-based, and reservation-based approaches.

Much work on cooperative motion planning has been devoted to road intersections, since these are hotspots for traffic accidents. Collision avoidance at intersections using V2V-communication for cooperation is investigated in [1] under the consideration of model uncertainty and communication delays. Colombo et al. [2] solve scheduling problems to ensure safety during intersection passages.

Another line of research is the design of cooperative lane-changing and merging strategies. In [3], it is discussed how V2V-communication can be utilized for cooperative decision making: a distributed receding horizon control framework is set up to solve tasks of platooning and cooperative merging. Further lane-changing and merging control algorithms for platoons of vehicles are developed in [4], [5].

Frese et al. [6] exploit priority-based motion planning to decouple the multi-vehicle motion planning problem, such that trajectory planning can only be conducted for single vehicles. This decreases the computational complexity; however, the solution space is reduced. Bekris et al. [7] combine a sampling-based motion planner with a priority-based coordination scheme, which is compared with a message-passing protocol for distributed constraint optimization. Moreover, priority-based algorithms for intersection management and traffic flow control are elaborated in [8]–[10].

Recently, market-based approaches have received major interest for multi-vehicle coordination, since they allow the incorporation of individual as well as global objectives, making it useful to balance self-interested and collective goals. In [11], maneuver plans are negotiated and refined via model predictive control. Auction-based coordination strategies for intersections can be found in [12]–[14].

Finally, we review literature concerning reservation-based algorithms [15]–[18], where communicating vehicles reserve some sort of space-time slots by requesting them via a supervisor. It must be guaranteed that the space-time slots are not occupied by more than one vehicle in order to ensure safety. In [15]–[17], reservation-based algorithms are applied to intersections; [15] in particular divides the intersection into tiles which can be allocated to communicating vehicles. Marinescu et al. [18] implement a slot-based approach for the merging of on-ramp traffic. They propose combining the hierarchical approach of exploiting vehicle-to-infrastructure communication with the decentralized approach of utilizing inter-vehicle communication for vehicle coordination.

We propose an algorithm related to the idea of reservation-based algorithms. In contrast to previous work, we distribute drivable areas of cooperative vehicles such that overlapping drivable areas are unambiguously reallocated to single cooperative vehicles. The computation of the drivable areas is based on reachability analysis, which allows us to not only compute a set of drivable positions but also to determine the maximum velocity range to reach a distinct set of positions. Thus, we are not restricted to distributing complete road segments among the collaborative vehicles, but we can precisely resolve conflicts. Our approach reduces the search space when planning coordinated maneuvers for multiple vehicles, since the trajectory of a single vehicle is restricted to its associated drivable area. Thus, the computational complexity of multi-vehicle motion planning can be decreased. Our method is suitable for identifying collaborative groups of vehicles for which motion planning can be conducted jointly. Moreover, it is possible to detect if collaboration with a cooperative vehicle becomes unnecessary. Since our method is set-based, the consideration of uncertainty in the initial position or velocity of the cooperative vehicles, e.g. due to sensor noise, is automatically supported. Our approach is applicable to mixed-traffic scenarios, where automated vehicles and human drivers co-exist.

The remainder of this paper is organized as follows: Sec. II introduces the problem statement and Sec. III presents the basic idea. A comprehensive description of our applied methods and our proposed algorithm is provided in Sec. IV. Sec. V demonstrates our approach on two numerical examples, followed by the conclusion in Sec. VI.

II. Problem Statement

Let us introduce $\square$ as the placeholder for a variable and $\square_n$ to denote the value of the corresponding variable of the n-th cooperative vehicle, $n \in \mathcal{N} := \{1, 2, \dots, N\}$. The system dynamics of the n-th vehicle is described by the differential equation

$$\dot{\mathbf{x}}_n(t) = f_n(\mathbf{x}_n(t), \mathbf{u}_n(t)),$$

where $\mathbf{x}_n$ is the state, $\mathbf{u}_n$ is the input, and t is the time. All possible initial states and the admissible inputs are bounded by sets: $\mathbf{x}_n(0) \in \mathcal{X}_{n,0}$, $\forall t : \mathbf{u}_n(t) \in \mathcal{U}_n$. We further introduce the solution of the differential equation $\dot{\mathbf{x}}_n(t) = f_n(\mathbf{x}_n(t), \mathbf{u}_n(t))$ as $\chi_n(t; \mathbf{x}_n(0), \mathbf{u}_n(\cdot))$, where $\mathbf{u}_n(\cdot)$ refers to the input trajectory.

The reachable set of the system $\dot{\mathbf{x}}_n(t) = f_n(\mathbf{x}_n(t), \mathbf{u}_n(t))$ is usually defined as the set of all states which can be reached from an initial set $\mathcal{X}_{n,0}$ at time t. However, the vehicles generally move in a structured environment cluttered with (time-dependent) obstacles represented by the set $\mathcal{O}(t) \subseteq \mathbb{R}^2$. Since we require the absence of collisions, each vehicle must not enter the set of forbidden states

$$\mathcal{F}_n(t) := \{\mathbf{x}_n(t) \in \mathcal{X}_n \,|\mathcal{Q}_n(\mathbf{x}_n(t)) \cap \mathcal{O}(t) \neq \emptyset\},$$

where $\mathcal{Q}_n(\mathbf{x}_n(t)) \subseteq \mathbb{R}^2$ denotes the occupancy of the n-th cooperative vehicle. Therefore, we restrict the reachable set $\mathcal{R}_n(\mathcal{X}_{n,0}, t)$ of the n-th vehicle to the set of states that can be reached without any collision with the obstacle set $\mathcal{O}(t)$:

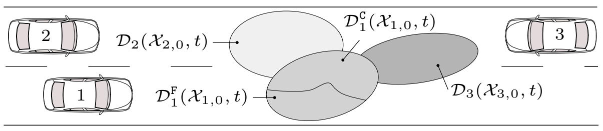
$$\mathcal{R}_n(\mathcal{X}_{n,0}, t) := \Big\{\chi_n(t; \mathbf{x}_n(0),\ \mathbf{u}_n(\cdot)) \Big| \mathbf{x}_n(0) \in \mathcal{X}_{n,0},$$
$$\forall \tau \in [0, t] : \mathbf{u}_n(\tau) \in \mathcal{U}_n, \chi_n(\tau; \mathbf{x}_n(0),\ \mathbf{u}_n(\cdot)) \notin \mathcal{F}_n(\tau)\Big\}.$$

The drivable area $\mathcal{D}_n(\mathcal{X}_{n,0}, t)$ of the n-th vehicle is given by the reachable positions at time t. We introduce the projection operator $\mathrm{proj}()$ to project a set of states to the position domain.

Definition 1 (Projection): Given that $\mathbf{x}(t) \in \mathcal{X}'$ contains the position $s_x(t)$ and $s_y(t)$ in x- and y-direction, we define the mapping from a set of states $\mathcal{X}' \subseteq \mathcal{X}$ to the set of positions as

$$\mathrm{proj}(\mathcal{X}') := \left\{[s_x(t), s_y(t)]^T \in \mathbb{R}^2 | \mathbf{x}(t) \in \mathcal{X}'\right\}.$$

Definition 2 (Drivable Area): The drivable area $\mathcal{D}_n(\mathcal{X}_{n,0}, t)$ of the n-th vehicle is defined as the projection of its reachable set $\mathcal{R}_n(\mathcal{X}_{n,0}, t)$: $\mathcal{D}_n(\mathcal{X}_{n,0}, t) := \mathrm{proj}(\mathcal{R}_n(\mathcal{X}_{n,0}, t))$.


2
$\mathcal{D}_2(\mathcal{X}_{2,0}, t)$
$\mathcal{D}_1^{\mathtt{C}}(\mathcal{X}_{1,0}, t)$
3
1
$\mathcal{D}_1^{\mathtt{F}}(\mathcal{X}_{1,0}, t)$
$\mathcal{D}_3(\mathcal{X}_{3,0}, t)$
Fig. 1: Conflicting and conflict-free drivable area $\mathcal{D}_1^{\mathtt{C}}(\mathcal{X}_{1,0}, t)$ and $\mathcal{D}_1^{\mathtt{F}}(\mathcal{X}_{1,0}, t)$ of vehicle 1.

The collaborative vehicles do not only have to avoid the set of forbidden states $\mathcal{F}_n(t)$, but they also have to prevent accidents among each other. Given the drivable areas $\mathcal{D}_n(\mathcal{X}_{n,0}, t)$ of N collaborative vehicles at time t, the occupancies of the vehicles may overlap (see Fig. 1). The overlapping region defines the area where conflicts potentially arise and a collision may occur. Thus, the drivable area $\mathcal{D}_n(\mathcal{X}_{n,0}, t)$ can be partitioned into a conflicting and a non-conflicting region.

Definition 3 (Conflicting Drivable Area): We introduce the conflicting reachable set $\mathcal{R}_n^{\mathtt{C}}(\mathcal{X}_{n,0}, t) \subseteq \mathcal{R}_n(\mathcal{X}_{n,0}, t)$ as the set of states $\mathbf{x}_n(t) \in \mathcal{R}_n(\mathcal{X}_{n,0}, t)$, where the occupancy $\mathcal{Q}_n(\mathbf{x}_n(t))$ of the n-th vehicle potentially intersects with the occupancy $\mathcal{Q}_k(\mathbf{x}_k(t))$ of another cooperative vehicle $k \in \mathcal{N} \setminus \{n\}$:

$$\mathcal{R}_n^{\mathtt{C}}(\mathcal{X}_{n,0}, t) := \Big\{\mathbf{x}_n(t) \in \mathcal{R}_n(\mathcal{X}_{n,0}, t) \Big| \mathbf{x}_k(t) \in \mathcal{R}_k(\mathcal{X}_{k,0}, t),$$
$$\exists k \in \mathcal{N} \setminus \{n\} : \mathcal{Q}_n(\mathbf{x}_n(t)) \cap \mathcal{Q}_k(\mathbf{x}_k(t)) \neq \emptyset\Big\}.$$

The conflicting drivable area $\mathcal{D}_n^{\mathtt{C}}(\mathcal{X}_{n,0}, t)$ of the n-th vehicle is then the projection of its conflicting reachable set $\mathcal{R}_n^{\mathtt{C}}(\mathcal{X}_{n,0}, t)$: $\mathcal{D}_n^{\mathtt{C}}(\mathcal{X}_{n,0}, t) := \mathrm{proj}(\mathcal{R}_n^{\mathtt{C}}(\mathcal{X}_{n,0}, t))$.

Definition 4 (Conflict-Free Drivable Area): The conflict-free drivable area of the n-th vehicle is

$$\mathcal{D}_n^{\mathtt{F}}(\mathcal{X}_{n,0}, t) := \mathcal{D}_n(\mathcal{X}_{n,0}, t) \setminus \mathcal{D}_n^{\mathtt{C}}(\mathcal{X}_{n,0}, t).$$

The goal of our approach is the reallocation of $\mathcal{D}_n^{\mathtt{C}}(\mathcal{X}_{n,0}, t)$ so that each cooperative traffic participant receives an individual area for motion planning. Henceforth, we will refer to the redistributed drivable area of the n-th vehicle as $\mathcal{D}_n^{\mathtt{R}}(\mathcal{X}_{n,0}, t)$.

Definition 5 (Redistributed Drivable Area): We define the redistributed drivable area as

$$\mathcal{D}_n^{\mathtt{R}}(\mathcal{X}_{n,0}, t) := \mathcal{D}_n^{\mathtt{F}}(\mathcal{X}_{n,0}, t) \cup \mathcal{E}_n(\mathcal{X}_{n,0}, t),$$

where $\mathcal{E}_n(\mathcal{X}_{n,0}, t) \subseteq \mathcal{D}_n^{\mathtt{C}}(\mathcal{X}_{n,0}, t)$. It holds that the interior of the redistributed drivable areas of all cooperative vehicles is pairwise disjoint.

III. Basic Idea

The starting point of our approach is the computation of the drivable areas $\mathcal{D}_n(\mathcal{X}_{n,0}, t)$ of each cooperative vehicle (see Fig. 2a). Since a conflict requires at least two different vehicles, the set of all conflicting subsets of vehicles is $\mathcal{P}_{\geq 2}(\mathcal{N})$, where $\mathcal{P}_{\geq 2}(\mathcal{N})$ denotes all subsets of the power set $\mathcal{P}(\mathcal{N})$ with cardinality greater than two. We demand that the vehicles staying in a conflict form a coalition ψ_r to solve it.

Definition 6 (Coalition): Let us introduce the relation $g : \{\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_e\} \to (\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_e)$ to convert the powerset $\mathcal{P}_{\geq 2}(\mathcal{N}) := \{\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_e\}$ into a tuple. The tuple $\Psi := g(\mathcal{P}_{\geq 2}(\mathcal{N}))$ is the ordered list of all unique subsets of vehicles which may have conflicting drivable areas. Henceforth, we refer to the r-th element $\psi_r \in \Psi$ as a coalition.

For instance, if we have $\mathcal{N} = \{1, 2, 3\}$, the tuple Ψ is:

$$\begin{aligned} \Psi &= g(\mathcal{P}_{\geq 2}(\mathcal{N})) \\ &= (\psi_1, \psi_2, \psi_3, \psi_4) \\ &= (\{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}). \end{aligned} \tag{1}$$

We assign a negotiable drivable area $\mathcal{D}^{\mathtt{N}}(\psi_r, t)$ to each coalition ψ_r. To this end, we simplify the determination of $\mathcal{D}^{\mathtt{N}}(\psi_r, t)$ and neglect the shape of the vehicles such that $\mathcal{Q}_n(\mathbf{x}_n(t)) = \mathbf{x}_n(t)$. This reduces the problem to the detection of overlapping regions (see Fig. 2b).

Definition 7 (Negotiable Drivable Area): We define the negotiable drivable area for each coalition $\psi_r \in \Psi$ as

$$\mathcal{D}^{\mathtt{N}}(\psi_r, t) := \bigcap_{n \in \psi_r} \mathcal{D}^{\mathtt{C}}_n(\mathcal{X}_{n,0}, t).$$

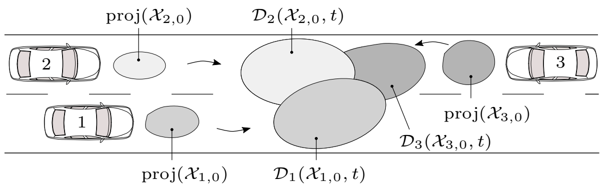
The members of the coalition ψ_r can distribute the negotiable drivable areas $\mathcal{D}^{\mathtt{N}}(\psi_r, t)$ among each other. The redistribution of the sets $\mathcal{D}^{\mathtt{N}}(\psi_r, t)$ can thereby be subject to a specific redistribution strategy, which e.g. minimizes a cost function J (see Fig. 2c).

The search for a feasible trajectory for each cooperative vehicle can be limited to its redistributed drivable area $\mathcal{D}^{\mathtt{R}}_n(\mathcal{X}_{n,0}, t)$. This can reduce the computational complexity of multi-vehicle motion planning, since the n-th vehicle may only cause a collision with obstacles $\mathcal{O}(t)$ and other cooperative vehicles $\mathcal{N} \setminus \{n\}$ close to the border of $\mathcal{D}^{\mathtt{R}}_n(\mathcal{X}_{n,0}, t)$. Especially, when one considers that a huge variety of combined trajectories of the collaborative vehicles can be excluded, since these combinations would lead to a collision. Our method reduces the set of trajectories which can be excluded, before motion planning. This can speed up the search for cooperative maneuvers.

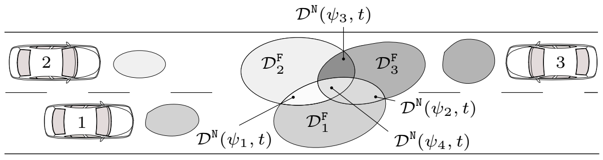
Moreover, cooperative groups can be identified through $\mathcal{D}^{\mathtt{N}}(\psi_r, \tau)$: if there exists a $\tau \in [0, t]$ such that $\mathcal{D}^{\mathtt{N}}(\psi_r, \tau) \neq \emptyset$ holds, the vehicles belonging to the coalition ψ_r should plan their motion jointly.

IV. Methodology and Algorithm

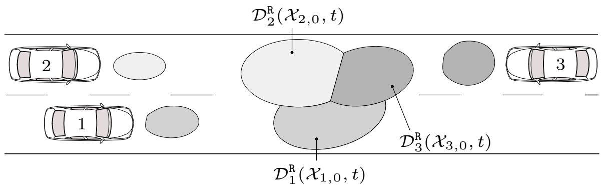
We apply an iterative approach which redistributes the drivable areas of the cooperative vehicles at discrete points in time t_i. During each iteration i, three steps have to be executed:


$\mathtt{proj}(\mathcal{X}_{2,0})$
$\mathcal{D}_2(\mathcal{X}_{2,0}, t)$
2
3
1
$\mathtt{proj}(\mathcal{X}_{3,0})$
$\mathcal{D}_3(\mathcal{X}_{3,0}, t)$
$\mathtt{proj}(\mathcal{X}_{1,0})$
$\mathcal{D}_1(\mathcal{X}_{1,0}, t)$

(a) Drivable areas $\mathcal{D}_n(\mathcal{X}_{n,0}, t)$.


$\mathcal{D}^{\mathtt{N}}(\psi_3, t)$
2
$\mathcal{D}^{\mathtt{F}}_2$
$\mathcal{D}^{\mathtt{F}}_3$
3
1
$\mathcal{D}^{\mathtt{F}}_1$
$\mathcal{D}^{\mathtt{N}}(\psi_2, t)$
$\mathcal{D}^{\mathtt{N}}(\psi_1, t)$
$\mathcal{D}^{\mathtt{N}}(\psi_4, t)$

(b) Non-conflicting drivable areas $\mathcal{D}^{\mathtt{F}}_n(\mathcal{X}_{n,0}, t)$ and negotiable drivable areas $\mathcal{D}^{\mathtt{N}}(\psi_r, t)$.


$\mathcal{D}^{\mathtt{R}}_2(\mathcal{X}_{2,0}, t)$
2
3
1
$\mathcal{D}^{\mathtt{R}}_3(\mathcal{X}_{3,0}, t)$
$\mathcal{D}^{\mathtt{R}}_1(\mathcal{X}_{1,0}, t)$

(c) Negotiation of $\mathcal{D}^{\mathtt{N}}(\psi_r, t)$ to determine the redistributed drivable areas $\mathcal{D}^{\mathtt{R}}_n(\mathcal{X}_{n,0}, t)$.

Fig. 2: Overview of the negotiation of drivable areas.

1) the computation of drivable areas $\mathcal{D}_n(\mathcal{X}_{n,i-1}, t_i)$,
2) the determination of negotiable areas $\mathcal{D}^{\mathtt{N}}(\psi_r, t_i)$,
3) the redistribution of all negotiable areas $\mathcal{D}^{\mathtt{N}}(\psi_r, t_i)$ to obtain $\mathcal{D}^{\mathtt{R}}_n(\mathcal{X}_{n,i-1}, t_i)$. The new set of states $\mathcal{X}_{n,i}$ is computed based on $\mathcal{D}^{\mathtt{R}}_n(\mathcal{X}_{n,i-1}, t_i)$.

There are three main challenges to be addressed: the efficient computation of the reachable set $\mathcal{R}_n(\mathcal{X}_{n,i-1}, t_i)$, the efficient assignment of the conflicting drivable areas $\mathcal{D}^{\mathtt{C}}_n(\mathcal{X}_{n,i-1}, t_i)$ to a coalition ψ_r, and the choice of the redistribution strategy.

The reachable set $\mathcal{R}_n(\mathcal{X}_{n,i-1}, t_i)$, more specifically $\mathcal{D}_n(\mathcal{X}_{n,i-1}, t_i)$, cannot be computed efficiently for general system models $f_n(\mathbf{x}_n(t), \mathbf{u}_n(t))$ in the presence of arbitrary obstacles $\mathcal{O}(t)$. The overapproximation of the reachable set $\mathcal{R}_n(\mathcal{X}_{n,i-1}, t_i)$ constitutes a compromise between computational efficiency and accuracy; we use the approach presented in [19] (see Sec. IV-B).

Since we neglect the shape of the cooperative vehicles and model them as moving point masses, we are able to apply fast algorithms from the field of computational geometry to assign the conflicting drivable areas $\mathcal{D}^{\mathtt{C}}_n(\mathcal{X}_{n,i-1}, t_i)$ to the coalitions ψ_r (see Sec. IV-C).

The choice of redistribution strategy highly influences the overall performance of the approach. In this paper, we select a strategy which should ensure that (see Sec. IV-D):

- the negotiable drivable areas $\mathcal{D}^{\mathtt{N}}(\psi_r, t_i)$ are fairly allocated, such that all vehicles have equally sized drivable areas
- the redistributed drivable areas $\mathcal{D}^{\mathtt{R}}_n(\mathcal{X}_{n,i-1}, t_i)$ are con-

nected.

The calculation of the drivable areas is based on models that are subsequently introduced and can be performed in a centralized or decentralized fashion. However, we assume that all computations are executed in a common coordinate system.

A. Vehicle and Obstacle Models

We model the dynamics of the cooperative vehicles as two double integrators with bounded velocity v and acceleration a. Let us introduce $\square_x$ and $\square_y$ to denote the value of the corresponding variable in x- and y-direction, respectively. After further introducing the notation $\underline{\square}$ and $\overline{\square}$ to specify the minimum and the maximum possible value of a variable, the dynamics is

$$\frac{\mathrm{d}}{\mathrm{d}t}\underbrace{\begin{bmatrix} s_{n,x} \\ v_{n,x} \\ s_{n,y} \\ v_{n,y} \end{bmatrix}}_{\mathbf{x}_n(t)} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} s_{n,x} \\ v_{n,x} \\ s_{n,y} \\ v_{n,y} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} a_{n,x} \\ a_{n,y} \end{bmatrix}}_{\mathbf{u}_n(t)}, \tag{2a}$$

$$\underline{v}_{n,x} \leq v_{n,x} \leq \overline{v}_{n,x}, \quad \underline{v}_{n,y} \leq v_{n,y} \leq \overline{v}_{n,y}, \tag{2b}$$

$$|a_{n,x}| \leq \overline{a}_{n,x}, \quad |a_{n,y}| \leq \overline{a}_{n,y}. \tag{2c}$$

The system dynamics of the obstacles is not restricted to any specific model. Moreover, we do not stipulate a particular method for the prediction of the occupancies of other traffic participants. Thus, any prediction supporting collision checks can be used, e.g. [20].

B. Computation of Drivable Areas

We compute the reachable set $\mathcal{R}_n(\mathcal{X}_{n,i-1}, t_i)$ of each cooperative vehicle at time t_i based on the set of states $\mathcal{X}_{n,i-1}$ of the previous time step t_{i-1} as presented in [19] and recall the main results of [19] in this subsection.

Pontryagin's principle is applied to calculate the boundaries of the reachable set of the n-th vehicle, ignoring the forbidden region $\mathcal{F}_n(t_i)$. Since the states of system model (2) are decoupled in longitudinal and lateral direction, their evolution over time can be computed separately. Two optimization problems are formulated in the x- and y-direction: given a specific terminal position, the objective is to maximize/minimize the speed at this position at time t. The Pontryagin principle yields a bang-bang input candidate solution with a single switching time. The velocity constraints (2b) are only considered at terminal time t. This results in an overapproximation of the reachable set, since the velocity constraints may be violated during $]t_{i-1}, t_i[$.

The lateral and longitudinal dynamics have to be considered jointly for the computation of $\mathcal{R}_n(\mathcal{X}_{n,i-1}, t_i)$, since it must be determined which states $\mathbf{x}_n(t_i)$ are within the set of forbidden states $\mathcal{F}_n(t_i)$. Söntges et. al [19] use the union of base sets $\mathcal{B}^{(q)}_{n,i}$, which are the Cartesian product of two convex polytopes in the (s_x, v_x)- and (s_y, v_y)-plane, to overapproximate $\mathcal{R}_n(\mathcal{X}_{n,i-1}, t_i)$ in a computationally efficient way:

$$\mathcal{R}_n(\mathcal{X}_{n,i-1}, t_i) \subseteq \bigcup_q \mathcal{B}^{(q)}_{n,i}.$$

The projection of the base sets $\mathcal{B}^{(q)}_{n,i}$ yields axis-aligned rectangles $\mathcal{A}^{(q)}_{n,i} := \mathrm{proj}(\mathcal{B}^{(q)}_{n,i})$ in the position domain. It holds that the interior of the axis-aligned rectangles $\mathcal{A}^{(q)}_{n,i}$ is pairwise disjoint [19] and the union of rectangles $\mathcal{A}^{(q)}_{n,i}$ is an overapproximation of the drivable area $\mathcal{D}_n(\mathcal{X}_{n,i-1}, t_i)$:

$$\mathcal{D}_n(\mathcal{X}_{n,i-1}, t_i) \subseteq \bigcup_q \mathcal{A}^{(q)}_{n,i}.$$

C. Assignment of Drivable Areas to Coalitions

Next, the negotiable drivable areas are determined. We exploit the fact that the drivable areas $\mathcal{A}_{n,i} := \{\mathcal{A}^{(1)}_{n,i}, \mathcal{A}^{(2)}_{n,i} \ldots\}$ are represented by axis-aligned rectangles (see Fig. 3a) and use a sweep line algorithm [21], [22], known from the field of computational geometry, to conduct the following tasks:

1) the detection of overlapping axis-aligned rectangles (see Fig. 3b),
2) the division of the drivable areas $\cup_{n\in\mathcal{N}} \cup_q \mathcal{A}^{(q)}_{n,i}$ into the conflict-free and the negotiable drivable areas $\mathcal{A}^{\mathtt{F}}_i := \{\mathcal{A}^{\mathtt{F}}_{1,i}, \ldots, \mathcal{A}^{\mathtt{F}}_{N,i}\}$ and $\mathcal{A}^{\mathtt{N}}_i := \{\mathcal{A}^{\mathtt{N}}_{\psi_1,i}, \ldots, \mathcal{A}^{\mathtt{N}}_{\psi_{|\Psi|},i}\}$ of a single vehicle $n \in \mathcal{N}$ and a coalition $\psi_r \in \Psi$, respectively. All subsets $\mathcal{A}^{\mathtt{F}}_{n,i}$ and $\mathcal{A}^{\mathtt{N}}_{\psi_r,i}$ are a collection of novel axis-aligned rectangles $\mathcal{A}^{\mathtt{F}(l)}_{n,i} \in \mathcal{A}^{\mathtt{F}}_{n,i}$ and $\mathcal{A}^{\mathtt{N}(p)}_{\psi_r,i} \in \mathcal{A}^{\mathtt{N}}_{\psi_r,i}$, whose interior is disjunct (see Fig. 3c).

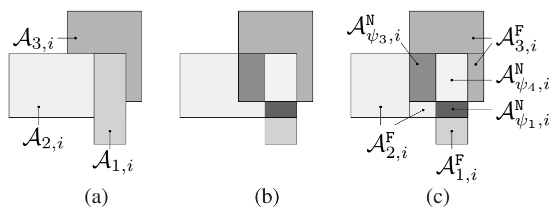

$\mathcal{A}_{3,i}$
$\mathcal{A}_{2,i}$
$\mathcal{A}_{1,i}$
$\mathcal{A}^{\mathtt{N}}_{\psi_3,i}$
$\mathcal{A}^{\mathtt{F}}_{3,i}$
$\mathcal{A}^{\mathtt{N}}_{\psi_4,i}$
$\mathcal{A}^{\mathtt{N}}_{\psi_1,i}$
$\mathcal{A}^{\mathtt{F}}_{2,i}$
$\mathcal{A}^{\mathtt{F}}_{1,i}$
(a)
(b)
(c)

Fig. 3: Assignment of drivable areas at time step i: $\mathcal{A}^{\mathtt{F}}_{n,i}$ and $\mathcal{A}^{\mathtt{N}}_{\psi_r,i}$ represent the conflict-free drivable area of the n-th vehicle and the negotiable area of the coalition $\psi_r \in \Psi$ (1), respectively.

D. Negotiation of Drivable Areas

Finally, the negotiable drivable areas $\mathcal{A}^{\mathtt{N}}_i$ are redistributed. Our applied method is related to the *Nearest Centroid Classifier* [23]: given a set of distinct clusters with associated class labels, a new observation is classified according to the class label of the nearest cluster centroid.

Here, each cooperative vehicle is considered a single class; therefore, the set of class labels is $\mathcal{N}$. Furthermore, each connected component of the conflict-free drivable area $\mathcal{A}^{\mathtt{F}}_{n,i}$ represents a cluster of the n-th vehicle. Thus, a single vehicle may have several clusters identified through the cluster centroid $\mathbf{c}^{(k)}_{n,i} := [x^{(k)}_{n,i}, y^{(k)}_{n,i}]^T$ with longitudinal and lateral position coordinate $x^{(k)}_{n,i}$ and $y^{(k)}_{n,i}$, respectively (see Fig. 4).

The connected components of the drivable area of a single vehicle are determined through a sweep line algorithm, which detects pairs of rectangles $\mathcal{A}^{\mathtt{F}(l)}_{n,i}, \mathcal{A}^{\mathtt{F}(o)}_{n,i} \in \mathcal{A}^{\mathtt{F}}_{n,i}$, for $l \neq o$,

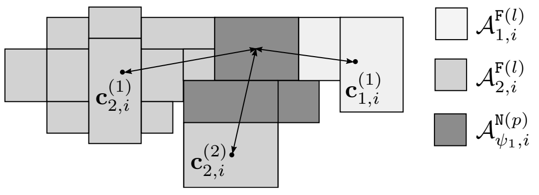

$\mathbf{c}_{2,i}^{(1)}$
$\mathbf{c}_{1,i}^{(1)}$
$\mathbf{c}_{2,i}^{(2)}$
$\mathcal{A}_{1,i}^{\mathtt{F}(l)}$
$\mathcal{A}_{2,i}^{\mathtt{F}(l)}$
$\mathcal{A}_{\psi_1,i}^{\mathtt{N}(p)}$

Fig. 4: The negotiable drivable areas are assigned to the vehicle with the nearest cluster centroid $\mathbf{c}_{n,i}^{(k)}$.

Algorithm 1

Input: Initial sets $\{\mathcal{B}_{1,0}, \ldots, \mathcal{B}_{N,0}\}$, collision detection for axis-aligned rectangles with $\{\mathcal{F}_1(t), \ldots, \mathcal{F}_N(t)\}$.
Output: $\{\mathcal{B}_{1,i}, \ldots, \mathcal{B}_{N,i}\}$ for $i = 1, \ldots, T$ time steps.

```
 1: function DRIVINGAREAS($\{\mathcal{B}_{1,0}, \ldots, \mathcal{B}_{N,0}\}$)
 2:     for $i = 1$ to $T$ do
 3:         $\{\mathcal{A}_{1,i}, \ldots, \mathcal{A}_{N,i}\} \leftarrow$ NEXTDRIVABLEAREAS(
                                        $\{\mathcal{B}_{1,i-1}, \ldots, \mathcal{B}_{N,i-1}\}$
                                        $\{\mathcal{F}_1(t_i), \ldots, \mathcal{F}_N(t_i)\}$)
 4:         $\mathcal{A}_i^{\mathtt{F}}, \mathcal{A}_i^{\mathtt{N}} \leftarrow$ COALITIONS($\{\mathcal{A}_{1,i}, \ldots, \mathcal{A}_{N,i}\}$)
 5:         $\{\mathcal{B}_{1,i}, \ldots, \mathcal{B}_{N,i}\} \leftarrow$ NEGOTIATE($\mathcal{A}_i^{\mathtt{F}}, \mathcal{A}_i^{\mathtt{N}}$)
 6:     end for
 7: end function

 8: function NEGOTIATE($\mathcal{A}_i^{\mathtt{F}}, \mathcal{A}_i^{\mathtt{N}}$)
 9:     $\mathcal{A}_i^{\mathtt{R}}$.initialize($\mathcal{A}_i^{\mathtt{F}}$)
10:     $\mathcal{C}_i \leftarrow$ COMPUTECLUSTERCENTROIDS($\mathcal{A}_i^{\mathtt{F}}$)
11:     for $\mathcal{A}_{\psi_r,i}^{\mathtt{N}} \in \mathcal{A}_i^{\mathtt{N}}$ do
12:         if $\mathcal{A}_{\psi_r,i}^{\mathtt{N}} := \emptyset$ then
13:             continue
14:         end if
15:         for $\mathcal{A}_{\psi_r,i}^{\mathtt{N}(p)} \in \mathcal{A}_{\psi_r,i}^{\mathtt{N}}$ do
16:             $n \leftarrow$ASSIGNMENT($\mathcal{A}_{\psi_r,i}^{\mathtt{N}(p)}, \mathcal{C}_i$)
17:             $\mathcal{A}_{n,i}^{\mathtt{R}} := \mathcal{A}_{n,i}^{\mathtt{R}} \cup \mathcal{A}_{\psi_r,i}^{\mathtt{N}(p)}$
18:         end for
19:     end for
20:     $\{\mathcal{B}_{1,i}, \ldots, \mathcal{B}_{N,i}\} \leftarrow$ NEXTBASESETS($\mathcal{A}_i^{\mathtt{R}}$)
21:     return $\{\mathcal{B}_{1,i}, \ldots, \mathcal{B}_{N,i}\}$
22: end function
```

whose boundaries intersect [22], [24]. The cluster centroids are computed by geometric decomposition:

$$
\begin{aligned}
x_{n,i}^{(k)} &= \frac{\sum_l \mathrm{centerX}\Big(\mathcal{A}_{n,i}^{\mathtt{F}(l)}\Big)\mathrm{area}\Big(\mathcal{A}_{n,i}^{\mathtt{F}(l)}\Big)}{\sum_l \mathrm{area}\Big(\mathcal{A}_{n,i}^{\mathtt{F}(l)}\Big)}, \\
y_{n,i}^{(k)} &= \frac{\sum_l \mathrm{centerY}\Big(\mathcal{A}_{n,i}^{\mathtt{F}(l)}\Big)\mathrm{area}\Big(\mathcal{A}_{n,i}^{\mathtt{F}(l)}\Big)}{\sum_l \mathrm{area}\Big(\mathcal{A}_{n,i}^{\mathtt{F}(l)}\Big)},
\end{aligned}
\tag{3}
$$

where $\mathcal{A}_{n,i}^{\mathtt{F}(l)}$ belongs to the k-th cluster of the n-th vehicle. The operators $\mathrm{centerX}(\square)$, $\mathrm{centerY}(\square)$, and $\mathrm{area}(\square)$ return the x- and y-coordinates of the center and the area of an axis-aligned rectangle $\square$, respectively.

After agreeing that the set $\mathcal{C}_{n,i} := \{\mathbf{c}_{n,i}^{(1)}, \mathbf{c}_{n,i}^{(2)}, \ldots\}$ contains all cluster centers $\mathbf{c}_{n,i}^{(k)}$ of the n-th vehicle at time step i, we denote the collection of all subsets $\mathcal{C}_{n,i}$ as $\mathcal{C}_i := \{\mathcal{C}_{1,i}, \ldots, \mathcal{C}_{N,i}\}$. Subsequently, each negotiable rectangle $\mathcal{A}_{\psi_r,i}^{\mathtt{N}(p)} \in \mathcal{A}_{\psi_r,i}^{\mathtt{N}}$ is assigned to the vehicle $n \in \psi_r$ with the nearest centroid $\mathbf{c}_{n,i}^{(k)}$ (see Fig. 4):

$$
\underset{n \in \psi_r}{\arg\min} \left(\min_{\mathbf{c}_{n,i}^{(k)} \in \mathcal{C}_{n,i}} d(\mathcal{A}_{\psi_r,i}^{\mathtt{N}(p)}, \mathbf{c}_{n,i}^{(k)}) \right), \tag{4}
$$

where $d(\mathcal{A}_{\psi_r,i}^{\mathtt{N}(p)}, \mathbf{c}_{n,i}^{(k)})$ denotes the Euclidean distance between the center of a rectangle $\mathcal{A}_{\psi_r,i}^{\mathtt{N}(p)}$ and a cluster center $\mathbf{c}_{n,i}^{(k)}$.

E. Algorithm

Alg. 1 shows the overall approach. First, the overapproximated drivable area $\mathcal{A}_{n,i} := \{\mathcal{A}_{n,i}^{(1)}, \mathcal{A}_{n,i}^{(2)}, \ldots\}$ of each cooperative vehicle is computed for each time step $i = 1, \ldots, T$ by propagating the base sets $\mathcal{B}_{n,i-1} := \{\mathcal{B}_{n,i-1}^{(1)}, \mathcal{B}_{n,i-1}^{(2)}, \ldots\}$ of the previous time step $i-1$ under consideration of $\mathcal{F}_n(t_i)$ using the approach of [19] (Alg. 1, line 3).

Then, the coalitions $\psi_r \in \Psi$ and their corresponding negotiable areas $\mathcal{A}_i^{\mathtt{N}}$, as well as the conflict-free areas $\mathcal{A}_i^{\mathtt{F}}$, are determined in the function COALITIONS as explained in Sec. IV-C (Alg. 1, line 4).

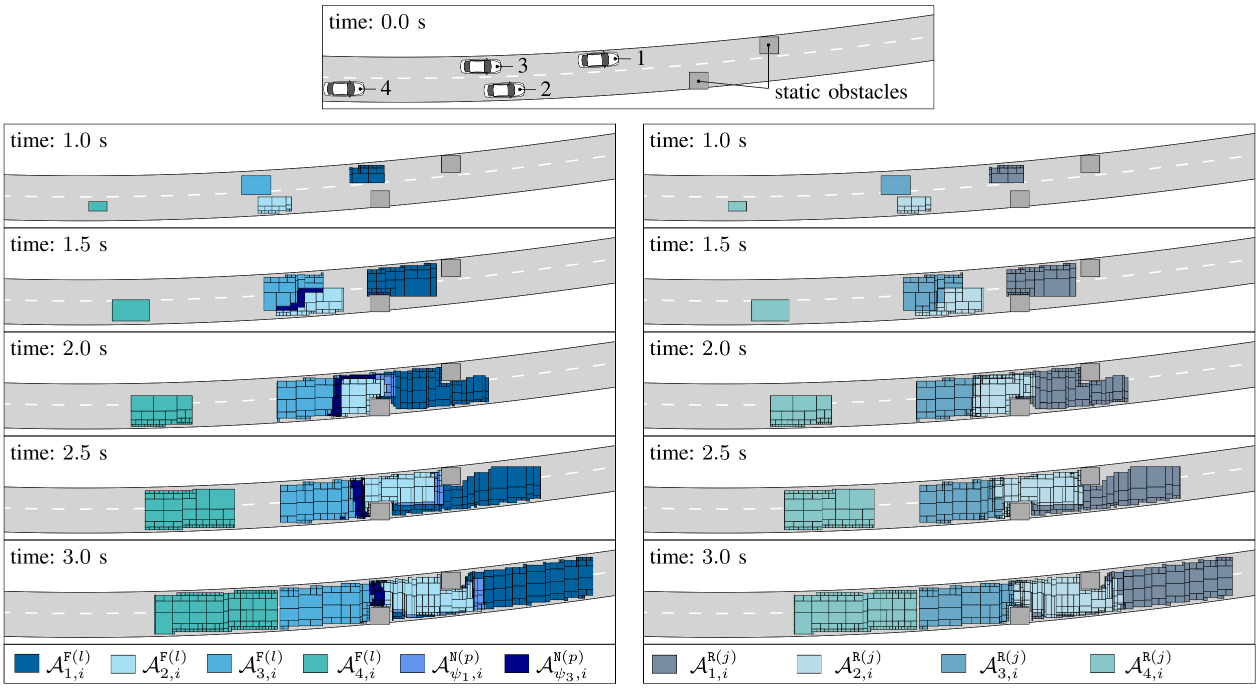
The final step of our approach is the negotiation of the areas $\mathcal{A}_i^{\mathtt{N}}$ in the function NEGOTIATE (Alg. 1, line 8). The redistributed drivable areas $\mathcal{A}_i^{\mathtt{R}} := \{\mathcal{A}_{1,i}^{\mathtt{R}}, \ldots, \mathcal{A}_{N,i}^{\mathtt{R}}\}$ are initialized with the conflict-free drivable areas (see Alg. 1, line 9). Afterwards, we obtain the connected components as elaborated in Sec. IV-D and compute their corresponding cluster centroids with (3) (Alg. 1, line 10). Then, each coalition ψ_r negotiates its associated area $\mathcal{A}_{\psi_r,i}^{\mathtt{N}}$, provided that the set $\mathcal{A}_{\psi_r,i}^{\mathtt{N}}$ is non-empty (Alg. 1, lines 12-14). Each rectangle $\mathcal{A}_{\psi_r,i}^{\mathtt{N}(p)} \in \mathcal{A}_{\psi_r,i}^{\mathtt{N}}$ is thereby assigned to the n-th vehicle through (4) in the function ASSIGNMENT (Alg. 1, line 16), where the Euclidean distance between the center of $\mathcal{A}_{\psi_r,i}^{\mathtt{N}(p)}$ and all cluster centroids $\mathbf{c}_{n,i}^{(k)}$ of the vehicles $n \in \psi_r$ is computed. The n-th vehicle with the nearest cluster centroid receives the area, and its redistributed drivable area $\mathcal{A}_{n,i}^{\mathtt{R}}$ is updated accordingly (Alg. 1, line 17). After the negotiation of the areas $\mathcal{A}_i^{\mathtt{N}}$, the new base sets $\mathcal{B}_{n,i}$ are computed by adding the velocity information to the redistributed drivable areas $\mathcal{A}_{n,i}^{\mathtt{R}}$, as presented in [19] (Alg. 1, line 20).

V. Numerical Examples

We show the application of our algorithm on two highway scenarios: in the first example, we only consider collaborative vehicles, whereas in the second example, we take mixed traffic based on recorded traffic data into account.

A. Scenario I: Traffic with Self-Driving Vehicles

In our first example, we consider four collaborative vehicles as depicted in Fig. 5, which have to pass a narrow passage caused by two static obstacles. The parameters for the drivable


time: 0.0 s
1
2
3
4
static obstacles
time: 1.0 s
time: 1.5 s
time: 2.0 s
time: 2.5 s
time: 3.0 s
$\mathcal{A}^{\mathrm{F}(l)}_{1,i}$ $\mathcal{A}^{\mathrm{F}(l)}_{2,i}$ $\mathcal{A}^{\mathrm{F}(l)}_{3,i}$ $\mathcal{A}^{\mathrm{F}(l)}_{4,i}$ $\mathcal{A}^{\mathrm{N}(p)}_{\psi_1,i}$ $\mathcal{A}^{\mathrm{N}(p)}_{\psi_3,i}$
$\mathcal{A}^{\mathrm{R}(j)}_{1,i}$ $\mathcal{A}^{\mathrm{R}(j)}_{2,i}$ $\mathcal{A}^{\mathrm{R}(j)}_{3,i}$ $\mathcal{A}^{\mathrm{R}(j)}_{4,i}$

(a) Conflict-free drivable areas $\mathcal{A}^{\mathrm{F}}_{n,i}$ and negotiable drivable areas $\mathcal{A}^{\mathrm{N}}_{\psi_r,i}$ using (1).

(b) Redistributed drivable areas $\mathcal{A}^{\mathrm{R}}_{n,i}$ after negotiation.

Fig. 5: Application of our proposed algorithm on a highway scenario with four collaborative vehicles.

area computation of vehicles 1 to 3 are selected equally, whereas vehicle 4 has less acceleration potential and the extreme values of the velocity are chosen to be different (see Tab. I).

Fig. 5a shows the conflict-free drivable areas $\mathcal{A}^{\mathrm{F}}_{n,i}$ and the negotiable drivable areas $\mathcal{A}^{\mathrm{N}}_{\psi_r,i}$ at different time steps i. Along the same lines, Fig. 5b depicts the redistributed drivable areas $\mathcal{A}^{\mathrm{R}}_{n,i}$ of each collaborative vehicle at the same time steps i. It can be determined that the movement of vehicle 4 is temporally conflict-free, since its drivable area does not intersect with any drivable area of the other vehicles. Thus, cooperation is unnecessary, and the trajectory planning for vehicle 4 can be conducted exclusively.

Furthermore, it can be identified that the drivable areas of vehicles 2 and 3 are in conflict first (see Fig. 5a at time instance 1.5 s). The conflict is resolved by giving vehicle 2 precedence over vehicle 3 to pass the obstacle in its current lane, which is automatically initiated by our approach. At a later stage, the drivable areas of vehicle 1 and 2 overlap; however, there is no change in the driving strategy, meaning that vehicle 1 stays in front of vehicle 2.

B. Scenario II: Mixed Traffic

We further demonstrate the applicability of our algorithm on the mixed-traffic scenario C-NGSIM_US101_1 based on the NGSIM US 101 Highway Dataset[1] (7:50 a.m. to 8:05 a.m.) from the CommonRoad[2] benchmark collection [25]. As shown in Fig. 7, there are four collaborative vehicles for which we compute the drivable areas based on the parameters given in Tab. II. The drivable areas of the vehicles are thereby restricted to five lanes, since we do not consider the highway on-ramp. We use the recorded trajectory data for the prediction of the obstacle movement and enlarge their rectangular shape to simulate uncertainty. However, any prediction which supports collision checks with axis-aligned rectangles can be applied.

TABLE I

Scenario I: Parameters for drivable area computation.

Parameter	**Value**
time discretization	0.1 s
time steps T	30
maximum speed $\overline{v}_{1/2/3,x}$	22.0 m/s
minimum speed $\underline{v}_{1/2/3,x}$	0.0 m/s
absolute maximum speed ($\overline{v}_{1/2/3,y} = -\underline{v}_{1/2/3,y}$)	15.0 m/s
absolute maximum acceleration $\overline{a}_{1/2/3,x}$	8.0 m/s^2
absolute maximum acceleration $\overline{a}_{1/2/3,y}$	4.0 m/s^2
maximum speed $\overline{v}_{4,x}$	18.0 m/s
minimum speed $\underline{v}_{4,x}$	0.0 m/s
absolute maximum speed ($\overline{v}_{4,y} = -\underline{v}_{4,y}$)	7.0 m/s
absolute maximum acceleration $\overline{a}_{4,x}$	4.0 m/s^2
absolute maximum acceleration $\overline{a}_{4,y}$	2.0 m/s^2

[1]http://www.fhwa.dot.gov/publications/research/operations/07030/

[2]http://commonroad.in.tum.de

TABLE II

Scenario II: Parameters for the drivable area computation.

Parameter	Value
time discretization	0.1 s
time steps T	30
maximum speed $\overline{v}_{n,x}$	36.0 m/s
minimum speed $\underline{v}_{n,x}$	0.0 m/s
absolute maximum speed ($\overline{v}_{n,y} = -\underline{v}_{n,y}$)	7.0 m/s
absolute maximum acceleration $\overline{a}_{n,x}$	5.5 m/s^2
absolute maximum acceleration $\overline{a}_{n,y}$	2.5 m/s^2

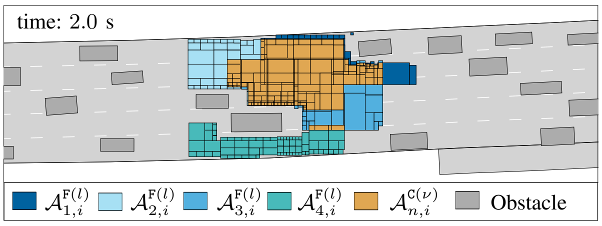

time: 2.0 s
$\mathcal{A}^{\mathtt{F}(l)}_{1,i}$
$\mathcal{A}^{\mathtt{F}(l)}_{2,i}$
$\mathcal{A}^{\mathtt{F}(l)}_{3,i}$
$\mathcal{A}^{\mathtt{F}(l)}_{4,i}$
$\mathcal{A}^{\mathtt{C}(\nu)}_{n,i}$
Obstacle

Fig. 6: Huge parts of the drivable areas of the collaborative vehicles overlap if the conflicting areas $\mathcal{A}^{\mathtt{C}(\nu)}_{n,i}$ are not negotiated.

We first analyze if there is the need for cooperation and compute the drivable areas of each collaborative vehicle using the approach of [19]. As shown in Fig. 6, the overapproximated drivable areas of vehicles 1, 2, and 3 overlap almost entirely at time instance 2.0 s (see $\mathcal{A}^{\mathtt{C}(\nu)}_{n,i}$ in Fig. 6). This indicates that a vast majority of combined trajectories of the collaborative vehicles lead to a collision among each other.

In contrast, when applying our proposed method, the propagated drivable areas are negotiated at each time step i such that they do not overlap. Moreover, high-level driving strategies are determined before motion planning (see Fig. 7): Vehicle 1 merges in front of vehicle 3. Vehicle 4 follows vehicle 3, whereas vehicle 2 may stay behind vehicle 3 or may drive behind vehicle 4 (see Fig. 7 at time instance 3.0 s). As can be seen, our approach works for complex traffic scenarios and efficiently distributes the drivable areas.

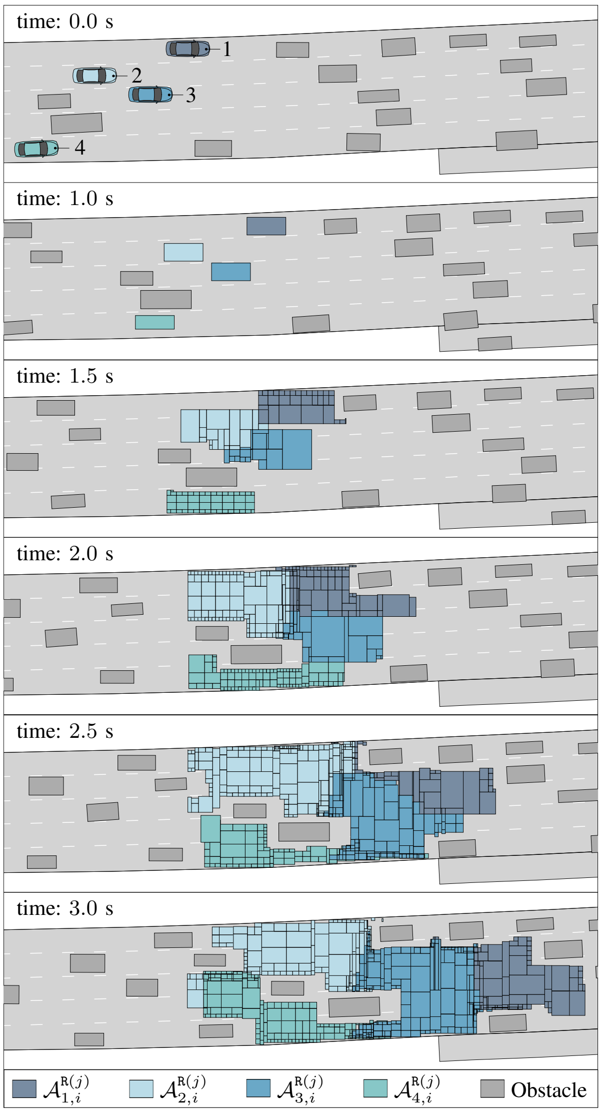

time: 0.0 s
1
2
3
4
time: 1.0 s
time: 1.5 s
time: 2.0 s
time: 2.5 s
time: 3.0 s
$\mathcal{A}^{\mathtt{R}(j)}_{1,i}$
$\mathcal{A}^{\mathtt{R}(j)}_{2,i}$
$\mathcal{A}^{\mathtt{R}(j)}_{3,i}$
$\mathcal{A}^{\mathtt{R}(j)}_{4,i}$
Obstacle

Fig. 7: Redistributed drivable areas $\mathcal{A}^{\mathtt{R}}_{n,i}$ at different points in time t_i.

VI. Conclusions

We present a new approach for cooperative conflict resolution based on the negotiation of drivable areas, wherein the computation of drivable areas is based on reachability analysis. Our method is suitable for reducing the search space for motion planning given the assumption that the admissible trajectories of a cooperative vehicle must be contained in its redistributed drivable area. An important aspect of our method is the independence of specific obstacle representations. In the future, we plan to investigate different negotiation strategies to extend our approach to arbitrary road networks. Furthermore, we want to integrate individual objectives during the negotiation phase, e.g. by using auction-based approaches [11], [26].

Acknowledgment

This work was supported by the Deutsche Forschungsgemeinschaft (German Research Foundation) within the Priority Program SPP 1835 Cooperative Interacting Automobiles (grant number: AL 1185/4-1).

References

[1] M. R. Hafner, D. Cunningham, L. Caminiti, and D. Del Vecchio, “Cooperative collision avoidance at intersections: Algorithms and experiments,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1162–1175, 2013.

[2] A. Colombo and D. Del Vecchio, “Least restrictive supervisors for intersection collision avoidance: A scheduling approach,” *IEEE Transactions on Automatic Control*, vol. 60, no. 6, pp. 1515–1527, 2015.

[3] D. Caveney and W. B. Dunbar, “Cooperative driving: Beyond V2V as an ADAS sensor,” in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2012, pp. 529–534.

[4] M. Goli and A. Eskandarian, “Evaluation of lateral trajectories with different controllers for multi-vehicle merging in platoon,” in *International Conference on Connected Vehicles and Expo*, 2014, pp. 673–678.

[5] H. C.-H. Hsu and A. Liu, “Kinematic design for platoon-lane-change maneuvers,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 1, pp. 185–190, 2008.

[6] C. Frese and J. Beyerer, “A comparison of motion planning algorithms for cooperative collision avoidance of multiple cognitive automobiles,” in *IEEE Intelligent Vehicles Symposium*, 2011, pp. 1156–1162.

[7] K. E. Bekris, K. I. Tsianos, and L. E. Kavraki, “Safe and distributed kinodynamic replanning for vehicular networks,” *Mobile Networks and Applications*, vol. 14, no. 3, pp. 292–308, 2009.

[8] J. Alonso, V. Milanés, J. Pérez, E. Onieva, C. González, and T. de Pedro, “Autonomous vehicle control systems for safe crossroads,” *Transportation research part C: emerging technologies*, vol. 19, no. 6, pp. 1095–1110, 2011.

[9] J. Grégoire, S. Bonnabel, and A. de La Fortelle, “Priority-based intersection management with kinodynamic constraints,” in *European Control Conference*, 2014, pp. 2902–2907.

[10] M. G. Plessen, D. Bernardini, H. Esen, and A. Bemporad, “Multi-automated vehicle coordination using decoupled prioritized path planning for multi-lane one- and bi-directional traffic flow control,” in *IEEE 55th Conference on Decision and Control*, 2016, pp. 1582–1588.

[11] H. Rewald and O. Stursberg, “Cooperation of autonomous vehicles using a hierarchy of auction-based and model-predictive control,” in *IEEE Intelligent Vehicles Symposium*, 2016, pp. 1078–1084.

[12] M. Vasirani and S. Ossowski, “A market-inspired approach for intersection management in urban road traffic networks,” *Journal of Artificial Intelligence Research*, vol. 43, pp. 621–659, 2012.

[13] H. Schepperle and K. Böhm, “Auction-based traffic management: Towards effective concurrent utilization of road intersections,” in *10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services*, 2008, pp. 105–112.

[14] D. Carlino, S. D. Boyles, and P. Stone, “Auction-based autonomous intersection management,” in *16th International IEEE Conference on Intelligent Transportation Systems*, 2013, pp. 529–534.

[15] K. Dresner and P. Stone, “Turning the corner: improved intersection control for autonomous vehicles,” in *IEEE Intelligent Vehicles Symposium*, 2005, pp. 423–428.

[16] O. Mehani and A. de La Fortelle, “Trajectory planning in a crossroads for a fleet of driverless vehicles,” in *Proc. of the 11th International Conference on Computer Aided Systems Theory*. Springer Berlin Heidelberg, 2007, pp. 1159–1166.

[17] A. de La Fortelle, “Analysis of reservation algorithms for cooperative planning at intersections,” in *13th International IEEE Conference on Intelligent Transportation Systems*, 2010, pp. 445–449.

[18] D. Marinescu, J. Čurn, M. Bouroche, and V. Cahill, “On-ramp traffic merging using cooperative intelligent vehicles: A slot-based approach,” in *15th International IEEE Conference on Intelligent Transportation Systems*, 2012, pp. 900–906.

[19] S. Söntges and M. Althoff, “Computing the drivable area of autonomous road vehicles in dynamic road scenes,” *IEEE Transactions on Intelligent Transportation Systems*, [to appear].

[20] M. Althoff and S. Magdici, “Set-based prediction of traffic participants on arbitrary road networks,” *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 2, pp. 187–202, 2016.

[21] J. L. Bentley and T. A. Ottmann, “Algorithms for reporting and counting geometric intersections,” *IEEE Transactions on Computers*, vol. C-28, no. 9, pp. 643–647, 1979.

[22] R. G. Karlsson and M. H. Overmars, “Scanline algorithms on a grid,” *BIT Numerical Mathematics*, vol. 28, no. 2, pp. 227–241, 1988.

[23] E.-H. S. Han and G. Karypis, *Centroid-Based Document Classification: Analysis and Experimental Results*. Springer Berlin Heidelberg, 2000, pp. 424–431.

[24] H. Edelsbrunner, J. Van Leeuwen, T. Ottmann, and D. Wood, “Computing the connected components of simple rectilinear geometrical objects in d-space,” *RAIRO, Informatique théorique*, vol. 18, no. 2, pp. 171–183, 1984.

[25] M. Althoff, M. Koschi, and S. Manzinger, “CommonRoad: Composable benchmarks for motion planning on roads,” in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 719–726.

[26] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, “Market-based multirobot coordination: A survey and analysis,” *Proc. of the IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006.

A.5 Tactical Decision Making for Cooperative Vehicles Using Reachable Sets [4]

Summary This paper presents our second work on cooperative motion planning using reachable sets and extensively redesigns the approach realized in Appendix A.4. Instead of redistributing conflicting reachable sets with the incentive to assign equally sized driving areas to cooperative vehicles, this work solves the negotiation process as a combinatorial optimization problem inspired by auctions. In general, the combinatorial optimization problem has a high computational complexity. However, by hierarchically structuring the conflicting road areas which have to be negotiated, we achieve computational tractability.

The benefits of our method are multi-fold: (1) the runtime complexity of the negotiation process is polynomial in the number of conflicting road areas that are represented by grid cells, but independent of the number of cooperative vehicles. (2) Individual goals of participating vehicles can be considered during the negotiation via utility functions. (3) Our approach is applicable to arbitrary traffic scenes and is not tailored to specific situations such as intersection management.

We demonstrate the efficacy of our method in multiple traffic scenarios from the CommonRoad benchmark suite featuring a roundabout, an urban road, a crossing, and a highway scenario. Although all these scenarios differ in the road geometry, the number of cooperative vehicles and the non-communicating road users, cooperative maneuvers could be planned successfully in all scenarios.

Author Contributions M. A. initiated the idea of using auctions for cooperative motion planning. **S. M.** developed the concept and algorithms. **S. M.** wrote the conference paper. **S. M.** designed, conducted, and evaluated the experiments. M. A. led the research project, provided feedback, and helped improving the manuscript.

Conference paper The author's version of the conference paper is reprinted in this thesis. The final version of record is available at `http://dx.doi.org/10.1109/ITSC.2018.8569560`.

Attachments The video attachment of this publication is available at `https://ieeexplore.ieee.org/document/8569560/media#media`.

Tactical Decision Making for Cooperative Vehicles Using Reachable Sets

Stefanie Manzinger and Matthias Althoff

Abstract— **Tactical maneuver planning of multiple, communicating vehicles provides the opportunity to increase passenger safety and comfort. We propose a unifying method to orchestrate the motion of cooperative vehicles based on the negotiation of conflicting road areas, which are determined by reachable set computation. As a result, each vehicle receives an individual driving corridor for trajectory planning. The presented conflict resolution scheme has polynomial runtime complexity and is guaranteed to find the optimal allocation of road areas for each negotiation round. Our method is not tailored to specific traffic situations but is applicable to general traffic scenes with manually driven and automated vehicles. We demonstrate the universal usability of our approach in numerical experiments.**

I. Introduction

There are many traffic situations in which individual navigation goals of traffic participants lead to conflicts. Human drivers often resolve these issues through implicit communication relying on the reasonable behavior of others. However, communicating automated vehicles offer more sophisticated solutions for collaborative maneuver planning. These vehicles can form a cooperative group, which jointly agrees on a common driving strategy to achieve conflict avoidance while maximizing individual utilities. One of the major challenges towards multi-vehicle motion planning is the development of coordination schemes which are computationally efficient without compromising optimality. We introduce a method for tactical decision making, which unambiguously assigns road areas to cooperating vehicles. Subsequently, we review literature focusing on similar conflict resolution principles.

Dresner et al. [1] pioneered the work on reservation-based algorithms with a focus on intersection management: the intersection space is discretized into tiles, which approaching vehicles can request via an intersection manager. This manager simulates trajectories of vehicles to determine the necessary tiles for passing the intersection and ensures that no tile is occupied by more than one vehicle. The original work has been extended in successive publications [2]–[4]. Sharon et al. [5] improve on the work of Dresner et al. [4] to be more efficient, particularly if the majority of vehicles present are driven by humans. In [6]–[8], the first-come, first-served policy for reservation assignment in [1] is replaced by auction-based methods. In [9], [10], conflict points instead of tiles are used as a resource for intersection management; Levin et al. [11] combine conflict points at intersections into conflict regions. In [12], a legacy algorithm based on reservations is proposed, which can handle a low percentage of non-communicating vehicles or vehicles with malfunctioning communication systems. A comprehensive overview of further techniques for cooperative intersection management can be found in [13].

While the former methods are applied to intersections, Marinescu et al. [14], [15] present virtual slots for traffic shaping. A virtual slot is a moving space-time corridor with a predefined behavior, e.g., lane following with constant speed or lane changing; vehicles assigned to slots must adopt their behavior. Their method is evaluated on a highway merging scenario. Alternative solutions to cooperative on-ramp merging onto highways can be found in [16]. Zhang et al. [17] propose a reservation-based scheduling technique to coordinate communicating vehicles through an intersection, which is divided into a set of static critical sections. Their goal is to establish a service-orientated traffic management, where high-priority vehicles are able to pass through the intersection first. In [18], dynamic critical sections are introduced among static critical sections to consider behaviors like lane-changing and overtaking. Cooperative maneuvers are defined as a sequence of states modeled within an event-triggered state automaton [18].

We intend to solve temporarily bounded conflicts, where vehicles collaborate for a limited amount of time based on reservations: road areas requested by several communicating traffic participants are distributed such that each vehicle receives its own driving corridor for trajectory planning. In contrast to previous work, we determine reservation conflicts by computing the drivable areas of all collaborative vehicles using reachability analysis. While many works assume that vehicles are highly automated, we intentionally deal with mixed traffic, where human-driven and automated vehicles share the road. Since scenarios with only automated vehicles are a special case, our algorithm can treat these situations as well. Moreover, our unifying method is not restricted to specific maneuvers or traffic situations.

Our approach can be categorized as a hybrid framework [13], where we optimally use the capabilities of each vehicle through decentralized computation of reachable sets, while the negotiation is performed by a leading vehicle. The presented paper is based on our previous work in [19]. The novelty of this work includes:

- Designing a general framework to incorporate individual goals of cooperative vehicles, whereas in [19], the value assessment of different road areas is exclusively based on geometric reasoning.

*This work was supported by the Deutsche Forschungsgemeinschaft (German Research Foundation) within the Priority Programme SPP 1835 Cooperative Interacting Automobiles (grant number: AL 1185/4-1).

All authors are with the Department of Computer Science, Technical University of Munich, 85748 Garching bei München, Germany `{stefanie.manzinger, althoff}@in.tum.de`

- Formulating a combinatorial optimization problem to optimally allocate conflicting road areas to the vehicles according to their preferences and reducing its computational complexity through hierarchical structuring of conflicting road areas. This makes it possible to resolve conflicts in polynomial runtime complexity in the number of road areas to be negotiated [20], but independently in the number of vehicles.
- Computing the reachable sets of cooperative traffic participants in a vehicle-specific, curvilinear coordinate system, whereas in [19], it is required that all computations are performed in a common coordinate system. This facilitates coping with different driving contexts as presented in Sec. VI.

Sec. II introduces the problem statement and Sec. III presents the necessary preliminaries. In Sec. IV-V, our applied methods and proposed algorithm are described. Sec. VI demonstrates our approach on numerical examples, followed by the conclusion in Sec.VII.

II. Problem Statement

Let $\mathcal{G} = \{g_0, g_1, \dots, g_i, \dots\}$ denote a grid with cells g_i of an arbitrary shape obtained through tessellation of the position domain in a Cartesian reference frame F_0 (see Fig. 1). The cells g_i are the individual assets of road areas which can be combined into unions of assets $\mathcal{C}_j \subseteq \mathcal{G}$, which we refer to as packages. We introduce the set $\mathcal{V} := \{V_1, V_2, \dots, V_N\}$ of cooperative vehicles acting as bidders, which can submit a bid for different sets $\mathcal{C}_j$. We restrict the set of permitted combinations $\mathcal{P}(t) \subseteq \mathtt{P}(\mathcal{G}^{\mathrm{c}}(t))$, where $\mathtt{P}()$ returns the powerset, to those $\mathcal{C}_j$ containing only conflicting cells $g_i \in \mathcal{G}^{\mathrm{c}}(t)$, $\mathcal{G}^{\mathrm{c}}(t) \subseteq \mathcal{G}$, requested from at least two vehicles V_n at time instance t (see Fig. 1). It is assumed that each vehicle V_n only bids its true value $b_n(t, \mathcal{C}_j)$ of a combination of assets $\mathcal{C}_j$. The maximum bid for a package $\mathcal{C}_j$ is $\bar{b}(t, \mathcal{C}_j)$, and any tie-breaking rule [20] is accepted to determine $\bar{b}(t, \mathcal{C}_j)$.

We aim to find a distribution of sets $\mathcal{C}_j \in \mathcal{P}(t)$ such that the revenue is maximized (1a) and no single asset is assigned more than once (1b) [20]:

$$\max_{\delta(t,\mathcal{C}_j)} \sum_{\mathcal{C}_j \in \mathcal{P}(t)} \delta(t, \mathcal{C}_j)\bar{b}(t, \mathcal{C}_j) \tag{1a}$$

such that

$$\forall g_i \in \mathcal{G}^{\mathrm{c}}(t): \sum_{\mathcal{C}_j:\ g_i \in \mathcal{C}_j} \delta(t, \mathcal{C}_j) \leq 1, \tag{1b}$$

$$\forall \mathcal{C}_j \in \mathcal{P}(t): \ \delta(t, \mathcal{C}_j) \in \{0, 1\}, \tag{1c}$$

where $\delta(t, \mathcal{C}_j)$ denotes the allocation of package $\mathcal{C}_j$ to the highest bidder V_n; $\delta(t, \mathcal{C}_j) = 1$ holds iff bidder V_n receives package $\mathcal{C}_j$ at time instance t.

The optimization problem (1) is known as the *winner determination problem*, which is NP-hard to solve [20], [21]. Moreover, allowing every possible combination of assets $g_i \in \mathcal{G}^{\mathrm{c}}(t)$ means that each bidder V_n has to evaluate $2^{|\mathcal{G}^{\mathrm{c}}(t)|} - 1$ packages. However, we are able to attain computational tractability by selecting a special structure of permitted combinations $\mathcal{P}(t)$: we require $\mathcal{P}(t)$ to form a tree structure; thus, an optimal allocation of packages can be found in $O(|\mathcal{G}^{\mathrm{c}}(t)|^2)$ time [20].

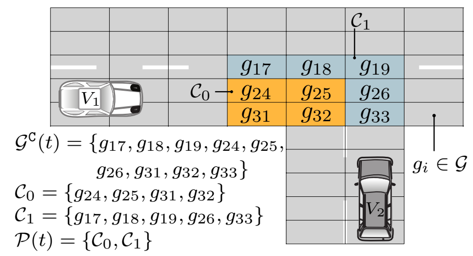

$\mathcal{C}_1$
g_{17}
g_{18}
g_{19}
V_1
$\mathcal{C}_0$
g_{24}
g_{25}
g_{26}
g_{31}
g_{32}
g_{33}
$\mathcal{G}^{\mathrm{c}}(t) = \{g_{17}, g_{18}, g_{19}, g_{24}, g_{25}, g_{26}, g_{31}, g_{32}, g_{33}\}$
$g_i \in \mathcal{G}$
$\mathcal{C}_0 = \{g_{24}, g_{25}, g_{31}, g_{32}\}$
$\mathcal{C}_1 = \{g_{17}, g_{18}, g_{19}, g_{26}, g_{33}\}$
V_2
$\mathcal{P}(t) = \{\mathcal{C}_0, \mathcal{C}_1\}$

Fig. 1. Visualization of the grid $\mathcal{G}$, the set of conflicting cells $\mathcal{G}^{\mathrm{c}}(t)$, the packages $\mathcal{C}_j$, and the set of permitted combinations $\mathcal{P}(t)$.

III. Preliminaries

We describe the system dynamics of the n-th cooperative vehicle in a local curvilinear coordinate system F_n by the differential equation

$$\dot{x}_n(t) = f_n(x_n(t), u_n(t)), \tag{2}$$

where $x_n \in \mathcal{X}_n \subseteq \mathbb{R}^p$ is the state, $u_n \in \mathcal{U}_n \subset \mathbb{R}^q$ is the input, and t is the time. We denote the solution of (2) for the input trajectory $u_n(\cdot)$ and initial state $x_n(0)$ with $\chi_n(t; x_n(0), u_n(\cdot))$. Please note that each vehicle may have its own coordinate system F_n.

The reachable set is defined as the set of all states which can be reached from an initial set of states $\mathcal{X}_{n,0}$ at a given time instance t. We extend this standard definition and restrict the reachable set $\mathcal{R}_n(\mathcal{X}_{n,0}, t)$ of vehicle V_n to all collision-free reachable states. We therefore introduce the set of forbidden states $\mathcal{F}_n(t) = \{x_n(t) | \mathcal{Q}_n(x_n(t)) \cap \mathcal{O}_n(t) \neq \emptyset\}$, where $\mathcal{Q}_n(x_n(t)) \subset \mathbb{R}^2$ and $\mathcal{O}_n(t) \subset \mathbb{R}^2$ denote the occupied space of the ego vehicle and the (time-varying) obstacles, respectively. Thus,

$$\mathcal{R}_n(\mathcal{X}_{n,0}, t) = \Big\{ \chi_n(t; x_n(0), u_n(\cdot)) \Big| x_n(0) \in \mathcal{X}_{n,0}, \tag{3}$$
$$\forall \tau \in [0, t]: u_n(\tau) \in \mathcal{U}_n, \chi_n(\tau; x_n(0), u_n(\cdot)) \notin \mathcal{F}_n(\tau) \Big\}.$$

Moreover, we specify the relation $h_n : \mathtt{P}(\mathcal{X}_n) \to \mathtt{P}(\mathcal{G})$, which returns the cells $g_i \in \mathcal{G}$ occupied by vehicle V_n due to its set of states $\mathcal{X}_n$ considering its shape, to determine the set $\mathcal{G}^{\mathrm{c}}(t)$ of conflicting cells claimed by multiple vehicles:

$$\mathcal{G}^{\mathrm{c}}(t) = \bigcup_{\mathcal{I} \in \mathtt{P}_{\geq 2}(\mathcal{N})} \bigcap_{n \in \mathcal{I}} h_n(\mathcal{R}_n(\mathcal{X}_{n,0}, t)), \tag{4}$$

where $\mathtt{P}_{\geq 2}(\mathcal{N})$ denotes all subsets of the power set $\mathtt{P}(\mathcal{N})$ with cardinality greater than one and $\mathcal{N} := \{1, 2, \dots, N\}$. We thereby assume the forward and backward transformation from F_n to F_0 to be given, see e.g. [22]. We further introduce the negotiated reachable set of vehicle V_n at time t:

$$\mathcal{R}^{\mathtt{N}}_n(\mathcal{X}_{n,0},t) = \Big\{ x_n(t) \in \mathcal{R}_n(\mathcal{X}_{n,0},t) \Big| \; h_n(\{x_n(t)\}) \cap \mathcal{G}^{\mathtt{L}}_n(t) = \emptyset \Big\}, \tag{5}$$

where $\mathcal{G}^{\mathtt{L}}_n(t) \subseteq \mathcal{G}^{\mathtt{C}}(t)$ denotes the set of unassigned grid cells g_i of vehicle V_n after the negotiation (1).

IV. Conflict Resolution

Conflict resolution is performed at discrete time steps k, which correspond to points in time $t_k = k\Delta t$, where $\Delta t \in \mathbb{R}^+$ is a constant time step. We identify the individual driving areas of vehicles V_n iteratively for each time step k by applying Alg. 1, which comprises the following steps:

1) computation of the reachable sets (3) (Alg. 1, line 3),
2) identification of conflicting cells $\mathcal{G}^{\mathtt{C}}(k)$ using (4) (Alg. 1, line 4),
3) negotiation of conflicting cells $\mathcal{G}^{\mathtt{C}}(k)$ to determine the optimal allocation $\mathcal{W}_{\mathtt{opt}}$ of $g_i \in \mathcal{G}^{\mathtt{C}}(k)$ (Alg. 1, line 5),
4) determination of the negotiated reachable sets (5) (Alg. 1, line 6).

Below, we elaborate steps 1) and 3) comprehensively and use the notation $[\square_n]_{n=1}^N = [\square_1,\dots,\square_n,\dots,\square_N]$ to denote a list of elements $\square_n$ of vehicles V_n.

Algorithm 1

```
1: function ConflictResolution($[\mathcal{B}^{\mathtt{N}}_n(0)]_{n=1}^N$, $\mathcal{G}$)
2:     for $k = 1$ to $T$ do
3:         $[\mathcal{B}_n(k)]_{n=1}^N \leftarrow$ReachableSets($[\mathcal{B}^{\mathtt{N}}_n(k-1)]_{n=1}^N$)
4:         $\mathcal{G}^{\mathtt{C}}(k) \leftarrow$ConflictingCells($[\mathcal{B}_n(k)]_{n=1}^N$, $\mathcal{G}$)
5:         $\mathcal{W}_{\mathtt{opt}} \leftarrow$Negotiate($[\mathcal{B}_n(k)]_{n=1}^N$, $\mathcal{G}^{\mathtt{C}}(k)$)
6:         $[\mathcal{B}^{\mathtt{N}}_n(k)]_{n=1}^N \leftarrow$ NegotiatedReachableSets(
                                   $[\mathcal{B}_n(k)]_{n=1}^N, \mathcal{W}_{\mathtt{opt}}$)
7:     end for
8:     return $[\cup_k \mathcal{B}^{\mathtt{N}}_n(k)]_{n=1}^N$
9: end function
```

A. Reachable Set Computation

1) Vehicle Dynamics: We model the dynamics of vehicle V_n in the local coordinate system F_n as two double integrators in longitudinal ζ_n- and lateral η_n-direction with bounded speed v_n and acceleration u_n. After introducing the notation $\underline{\square}$ and $\overline{\square}$ to specify the minimum and the maximum possible value of a variable $\square$, the dynamics is

$$\ddot{s}_{n,\zeta_n}(t) = u_{n,\zeta_n}(t), \qquad \ddot{s}_{n,\eta_n}(t) = u_{n,\eta_n}(t), \tag{6a}$$

$$\underline{v}_{n,\zeta_n} \le v_{n,\zeta_n}(t) \le \overline{v}_{n,\zeta_n}, \quad \underline{v}_{n,\eta_n} \le v_{n,\eta_n}(t) \le \overline{v}_{n,\eta_n}, \tag{6b}$$

$$|u_{n,\zeta_n}(t)| \le \overline{a}_{n,\zeta_n}, \quad |u_{n,\eta_n}(t)| \le \overline{a}_{n,\eta_n}, \tag{6c}$$

where $s_{n,\zeta_n}(t)$ and $s_{n,\eta_n}(t)$ denote the position in longitudinal and lateral direction, respectively.

Model (6) is an approximation of the real vehicle dynamics, which deviates increasingly from a real vehicle the larger the curvature of the road; the modeled vehicle would be able to make a turn with an arbitrarily high velocity. However, we compensate for this by setting appropriate constraints (6b)-(6c). The use of a curvilinear coordinate frame facilitates the formulation of certain properties and maneuvers, which are highly relevant to our approach, e.g., lane-following, stopping at an intersection, and avoiding driving backwards.

2) Reachable Set: Reachable sets are computed according to [23]: we approximate the reachable set at time step k by the union of base sets $\mathcal{B}^{(i)}_n(k)$, which are composed of the Cartesian product of two convex polytopes in the $(s_{n,\zeta_n}, v_{n,\zeta_n})$- and $(s_{n,\eta_n}, v_{n,\eta_n})$-plane:

$$\mathcal{R}_n(\mathcal{B}^{\mathtt{N}}_n(k-1), t_k) \approx \bigcup_i \mathcal{B}^{(i)}_n(k) =: \mathcal{B}_n(k),$$

where $\mathcal{B}^{\mathtt{N}}_n(k-1)$ denotes the negotiated reachable sets of the previous time step $k-1$. The projection of base sets $\mathcal{B}^{(i)}_n(k)$ in the position domain—in this paper referred to as drivable area—yields axis-aligned rectangles.

The reachable sets are computed with reference to the center of gravity of the vehicle; however, we need to consider the shapes of vehicles for collision detection. Söntges et al. [23] use the inner circle of the vehicle shape for collision detection, whereas we approximate the shape of vehicles V_n with three rotationally invariant disks with radius r_n and assume that the heading of the vehicles is aligned with ζ_n in F_n. Please note that the reachable set computation is not overapproximative due to the modified collision checks.

B. Negotiation of Conflicting Cells

Our negotiation scheme for resolving conflicts between collaborative vehicles is inspired by the idea of auctions. An auction requires a set of buyers competing for limited resources; bids are used to express preferences over the auctioned resources. In this work, cooperative vehicles act as bidders, and the limited resource is the drivable space on the road. In contrast to general auction design, we do not introduce a pricing mechanism [21]. Furthermore, we apply a common utility function to determine the bids for each vehicle. The negotiation of conflicting road cells comprises the following steps (see Alg. 2):

1) determination of permitted packages $\mathcal{C}_j \in \mathcal{P}(k)$ on the basis of $\mathcal{G}^{\mathtt{C}}(k)$ and their structuring in a tree $\mathcal{T}$ (see Fig. 2b, Alg. 2, line 4),
2) evaluation of bids $\overline{b}(k, \mathcal{C}_j)$ for $\mathcal{C}_j \in \mathcal{P}(k)$ (Alg. 2, line 5),
3) computation of the optimal allocation $\mathcal{W}_{\mathtt{opt}}$ of permitted packages $\mathcal{C}_j \in \mathcal{P}(k)$ (Alg. 2, line 6).

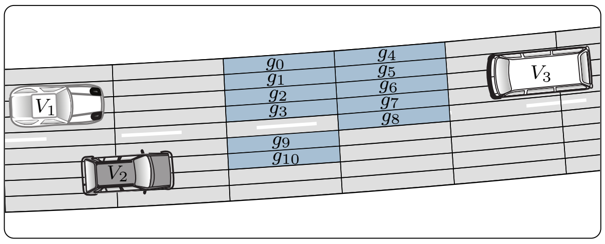
In the remainder of this section, all necessary steps are discussed in detail.

1) Tree Structure: The entire combination $\mathcal{G}^{\mathtt{C}}(k)$ including all negotiable assets is the root node of the tree (see Fig. 2b). At each level of the tree, we decompose the sets $\mathcal{C}_j$ into disjoint parts, of which each part represents a set $\mathcal{C}_l \in \mathcal{P}(k)$. Thus, for all $\mathcal{C}_j, \mathcal{C}_i \in \mathcal{P}(k)$, we have $\mathcal{C}_j \cap \mathcal{C}_i \in \{\emptyset, \mathcal{C}_j, \mathcal{C}_i\}$ [20] (see Fig. 2b).

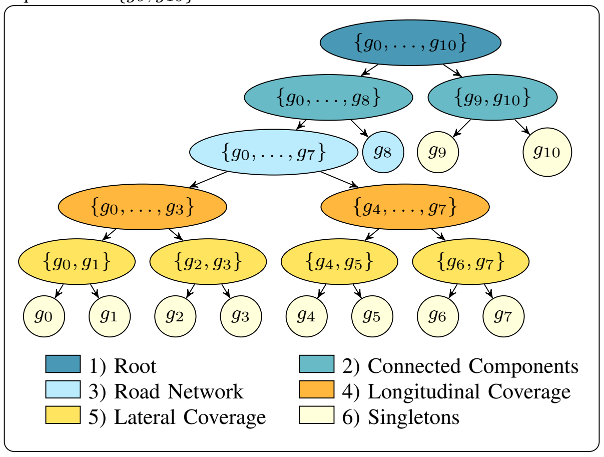
As mentioned in Sec. II, using a tree structure to express the set of permitted combinations $\mathcal{P}(k)$ of cells simplifies

Algorithm 2

```
1: function NEGOTIATE($[\mathcal{B}_n(k)]_{n=1}^{N}$, $\mathcal{G}^{\mathrm{c}}(k)$)
2:     $\mathcal{W}_{\mathtt{opt}} \leftarrow \emptyset$
3:     if $\mathcal{G}^{\mathrm{c}}(k) \neq \emptyset$ then
4:         $\mathcal{T} \leftarrow$TREEOFPERMITTEDPACKAGES($\mathcal{G}^{\mathrm{c}}(k)$)
5:         $\cup_{\mathcal{C}_j \in \mathcal{P}(k)} \bar{b}(k, \mathcal{C}_j) \leftarrow$BIDS($[\mathcal{B}_n(k)]_{n=1}^{N}$, $\mathcal{T}$)
6:         $\mathcal{W}_{\mathtt{opt}} \leftarrow$OPTIMALALLOCATION($\mathcal{T}$, $\cup_{\mathcal{C}_j \in \mathcal{P}(k)} \bar{b}(k, \mathcal{C}_j)$)
7:     end if
8:     return $\mathcal{W}_{\mathtt{opt}}$
9: end function
```



(a) Vehicle V_1 is overtaking vehicle V_2. Vehicle V_1 requests cells $\{g_0, \dots, g_{10}\}$, vehicle V_3 requests cells $\{g_0, \dots, g_8\}$, and vehicle V_2 requests cells $\{g_9, g_{10}\}$.



(b) Possible tree structure using the decomposition strategy as explained in Sec. IV-B.1.

Fig. 2. Exemplary grouping of conflicting cells according to Sec. IV-B.1.

problem (1) substantially. We further motivate the hierarchical structure of admissible packages through the following observations: we are interested in negotiating connected regions on the road surface to keep the driving area of vehicles from becoming disjointed, which complicates trajectory planning. Moreover, vehicles have to obey traffic regulations and restrictions imposed by the road network; thus, it is plausible to cluster conflicting cells according to lanes. Since using a tree structure facilitates the optimal allocation of the offered packages in $O(|\mathcal{G}^{\mathrm{c}}(k)|^2)$ time complexity, it is possible to partition the road surface fine-granularly and split packages such that they solely contain single assets. The restriction that each level of the tree contains only disjoint packages $\mathcal{C}_j$ can lead to a mismatch in offered packages and desired packages, but overall, the aforementioned advantages of the hierarchical structuring of admissible packages outweigh a potential mismatch in offered and desired packages. We therefore recommend applying the strategy below to group conflicting cells (see Fig. 2; the numbering below coincides with the legend in Fig. 2b):

1) *Root*: The root node consists of all conflicting cells $\mathcal{G}^{\mathrm{c}}(k)$.
2) *Connected Components:* The connected components are aggregated into packages $\mathcal{C}_j$.
3) *Road Network:* Packages of cells $\mathcal{C}_j$ are grouped according to the lanes of the road network. If a cell cannot be uniquely assigned to a lane, we categorize the cell randomly (see Fig. 2a, cell g_8).
4) *Longitudinal Spatial Coverage:* The packages are decomposed in longitudinal direction so that each new package does not exceed a maximum longitudinal spatial coverage.
5) *Lateral Spatial Coverage:* The packages are decomposed in lateral direction so that each new package does not exceed a maximum lateral spatial coverage.
6) *Singletons:* The packages comprise only singletons.

It should be noted that not all steps 1)–6) have to be executed, e.g., it is possible to apply 1) and 3) only such that the root node is split according to the road network.

2) Bids: We do not pose any specific constraint on the utility function to determine the bids $b_n(k, \mathcal{C}_j)$ for each package $\mathcal{C}_j \in \mathcal{P}(k)$ and vehicle V_n at time step k. Moreover, complementaries ($b(k, \{g_i, g_l\}) > b(k, \{g_i\}) + b(k, \{g_l\})$ for $i \neq l$) and substitutes [21] ($b(k, \{g_i, g_l\}) < b(k, \{g_i\}) + b(k, \{g_l\})$ for $i \neq l$) can be modeled. We use a common utility function to all vehicles in this paper to avoid that one vehicle could continuously outbid others due to different scales and weights used to calculate $b_n(k, \mathcal{C}_j)$. A conceivable countermeasure is the introduction of a pricing mechanism, which is the subject of future research. Furthermore, a vehicle V_n can only bid on a package $\mathcal{C}_j$ iff it contains at least one reachable cell of V_n: $\exists g_i \in \mathcal{C}_j : \ g_i \in h_n(\mathcal{B}_n(k))$.

3) Optimal Allocation: The algorithm for finding the optimal allocation of goods $\mathcal{C}_j \in \mathcal{P}(k)$ proposed by Rothkopf et al. [20] is recapitulated in Alg. 3: starting from the deepest leaf $\mathcal{C}_{\mathtt{max}}$ in the tree (Alg. 3, line 6), we determine its parent node $\mathcal{C}_{\mathtt{parent}}$ (Alg. 3, line 7) and the children $\mathcal{S}$ of $\mathcal{C}_{\mathtt{parent}}$ (Alg. 3, line 8). Next, we compare the revenue $\mathtt{rev}(\mathcal{S}) = \sum_{\mathcal{C}_s \in \mathcal{S}} \bar{b}(k, \mathcal{C}_s)$ of all children of $\mathcal{C}_{\mathtt{parent}}$ with bid $\bar{b}(k, \mathcal{C}_{\mathtt{parent}})$ (Alg. 3, lines 10-15):

- if $\bar{b}(k, \mathcal{C}_{\mathtt{parent}}) > \mathtt{rev}(\mathcal{S})$ holds, sets $\mathcal{C}_s \in \mathcal{S}$ cannot be part of the optimal allocation (Alg. 3, lines 10 - 12).
- if $\bar{b}(k, \mathcal{C}_{\mathtt{parent}}) \leq \mathtt{rev}(\mathcal{S})$ holds, $\mathcal{C}_{\mathtt{parent}}$ is excluded from the optimal assignment (Alg. 3, lines 13 - 15).

The children $\mathcal{S}$ are removed from the tree $\mathcal{T}$ (Alg. 3, line 5), and the process is repeated until $\mathcal{C}_{\mathtt{parent}}$ becomes the root node (Alg. 3, line 16): $\mathcal{C}_{\mathtt{parent}} = \mathcal{G}^{\mathrm{c}}(k)$.

C. Multiple Runs for Refinement

The negotiated driving corridors can be improved through multiple runs of Alg. 1. The reachable set computation is

Algorithm 3 Optimal Allocation of Packages [20].

```
1: function OptimalAllocation($\mathcal{T}$, $\cup_{\mathcal{C}_j \in \mathcal{P}(k)} \bar{b}(k, \mathcal{C}_j)$)
2:     $\mathcal{T}$.Initialize( )    ▷ Set $\mathcal{W}_{\mathrm{opt}}(\mathcal{C}_l) = \{\mathcal{C}_l\}$ for every leaf $\mathcal{C}_l$.
3:     $\mathcal{S} \leftarrow \emptyset$
4:     do
5:         $\mathcal{T}$.RemoveNodes($\mathcal{S}$)
6:         $\mathcal{C}_{\max} \leftarrow \mathcal{T}$.GetDeepestLeaf( )
7:         $\mathcal{C}_{\mathrm{parent}} \leftarrow \mathcal{C}_{\max}$.GetParent( )
8:         $\mathcal{S} \leftarrow \mathcal{C}_{\mathrm{parent}}$.GetChildren( )
9:         $\mathtt{rev}(\mathcal{S}) \leftarrow \sum_{\mathcal{C}_s \in \mathcal{S}} \bar{b}(k, \mathcal{C}_s)$
10:        if $\bar{b}(k, \mathcal{C}_{\mathrm{parent}}) > \mathtt{rev}(\mathcal{S})$ then
11:            $\mathcal{W}_{\mathrm{opt}}(\mathcal{C}_{\mathrm{parent}}) \leftarrow \{\mathcal{C}_{\mathrm{parent}}\}$
12:        else
13:            $\bar{b}(k, \mathcal{C}_{\mathrm{parent}}) \leftarrow \mathtt{rev}(\mathcal{S})$
14:            $\mathcal{W}_{\mathrm{opt}}(\mathcal{C}_{\mathrm{parent}}) \leftarrow \cup_{\mathcal{C}_s \in \mathcal{S}} \mathcal{W}_{\mathrm{opt}}(\mathcal{C}_s)$
15:        end if
16:    while $\mathcal{C}_{\mathrm{parent}} \neq \mathcal{G}^{\mathrm{c}}(k)$
17:    return $\mathcal{W}_{\mathrm{opt}}(\mathcal{G}^{\mathrm{c}}(k))$
18: end function
```

based on the results of the previous time step only. Thus, there might exist states in $\mathcal{B}_n^{\mathtt{N}}(k)$ from which a trajectory cannot be continued without leaving the negotiated driving corridor $\mathcal{B}_n^{\mathtt{N}}(i)$ in later time steps $i \in \{k+1, \ldots, T\}$ [23]. We are able to remove a subset of those states by running Alg. 1 multiple times, since information about future time steps from a previous run can be incorporated into the reachable set computation. The interested reader is referred to [23] for further information.

V. Utility Function and Tie-Breaking Rule

This section introduces the applied utility function (see Sec. IV-B.2) and tie-breaking rule (see Sec. II) used in this paper. Please note that both the utility function and tie-breaking rule can be exchanged by other rules.

A. Utility Function

Let us introduce:

- the conflict-free reachable set: $\mathcal{R}_n^{\mathtt{CF}}(k) = \{x_n(k) \in \mathcal{B}_n(k) | h_n(\{x_n(k)\}) \cap \mathcal{G}^{\mathrm{c}}(k) = \emptyset\}$;
- the conflicting reachable set depending on package $\mathcal{C}_j$ that would be lost if package $\mathcal{C}_j$ is not assigned to vehicle V_n: $\mathcal{R}_n^{\mathtt{C}}(k, \mathcal{C}_j) = \{x_n(k) \in \mathcal{B}_n(k) | h_n(\{x_n(k)\}) \cap \mathcal{C}_j \neq \emptyset\}$;
- the assigned reachable set which V_n can keep besides $\mathcal{R}_n^{\mathtt{CF}}(k)$ given that package $\mathcal{C}_j$ is assigned to vehicle V_n: $\mathcal{R}_n^{\mathtt{A}}(k, \mathcal{C}_j) = \{x_n(k) \in \mathcal{B}_n(k) \setminus \mathcal{R}_n^{\mathtt{CF}}(k) | h_n(\{x_n(k)\}) \cap (\mathcal{G}^{\mathrm{c}}(k) \setminus \mathcal{C}_j) = \emptyset\}$.

The above sets are the basis for computing the utility of $\mathcal{C}_j$ for each vehicle V_n to determine $b_n(k, \mathcal{C}_j)$. For computational reasons, we approximate sets $\mathcal{R}_n^{\mathtt{CF}}(k)$, $\mathcal{R}_n^{\mathtt{C}}(k, \mathcal{C}_j)$, and $\mathcal{R}_n^{\mathtt{A}}(k, \mathcal{C}_j)$ with the union of base sets (see Sec. IV-A.2); the approximated sets are denoted with $\mathcal{B}_n^{\mathtt{CF}}(k) := \cup_i \mathcal{B}_n^{\mathtt{CF}(i)}(k)$, $\mathcal{B}_n^{\mathtt{C}}(k, \mathcal{C}_j) := \cup_i \mathcal{B}_n^{\mathtt{C}(i)}(k)$, and $\mathcal{B}_n^{\mathtt{A}}(k, \mathcal{C}_j) := \cup_i \mathcal{B}_n^{\mathtt{A}(i)}(k)$, respectively. On the one hand, we take the objectives of cooperative vehicles V_n into account by applying utility function $U_n^{\mathtt{R}}(k, \mathcal{C}_j)$ in the regular mode; on the other hand, we introduce utility function $U_n^{\mathtt{S}}(k, \mathcal{C}_j)$ to prevent the complete loss of the reachable set $\mathcal{B}_n(k)$ of vehicles V_n in the survival mode, since this would correspond to an empty driving corridor for trajectory planning:

$$b_n(k, \mathcal{C}_j) = \begin{cases} U_n^{\mathtt{R}}(k, \mathcal{C}_j), & \mathtt{area}(\mathcal{B}_n^{\mathtt{CF}}(k)) > \underline{A}, \\ U_n^{\mathtt{S}}(k, \mathcal{C}_j), & \mathtt{area}(\mathcal{B}_n^{\mathtt{CF}}(k)) \leq \underline{A}, \end{cases} \tag{7}$$

where $\underline{A}$ is an adjustable threshold and $\mathtt{area}(\square)$ returns the size of the drivable area of sets $\square$. As a reminder, the reachable set projected onto the position domain is referred to as the drivable area (see Sec. IV-A.2). Below, we elaborate $U_n^{\mathtt{R}}(k, \mathcal{C}_j)$ and $U_n^{\mathtt{S}}(k, \mathcal{C}_j)$ applied in the regular and survival mode, respectively.

1) Regular Mode: If the conflict-free drivable area of vehicle V_n is greater than $\underline{A}$, we apply $U_n^{\mathtt{R}}(k, \mathcal{C}_j)$, which computes the ratio of the utility of the reachable set $\mathcal{B}_n^{\mathtt{A}}(k, \mathcal{C}_j)$ obtained through package $\mathcal{C}_j$ and the utility of the conflict-free reachable set $\mathcal{B}_n^{\mathtt{CF}}(k)$:

$$U_n^{\mathtt{R}} = \frac{\sum_i \Big(u_{\mathtt{vel}}(\mathcal{B}_n^{\mathtt{A}(i)}(k)) + u_{\mathtt{range}}(\mathcal{B}_n^{\mathtt{A}(i)}(k)) \Big) \cdot \mathtt{area}\Big(\mathcal{B}_n^{\mathtt{A}(i)}(k) \Big)}{\sum_i \Big(u_{\mathtt{vel}}(\mathcal{B}_n^{\mathtt{CF}(i)}(k)) + u_{\mathtt{range}}(\mathcal{B}_n^{\mathtt{CF}(i)}(k)) \Big) \cdot \mathtt{area}\Big(\mathcal{B}_n^{\mathtt{CF}(i)}(k) \Big)},$$

with partial utility functions $u_{\mathtt{vel}}$ and $u_{\mathtt{range}}$ presented next.

In order to increase traffic flow, we reward an increase in longitudinal speed from the previous time step $k-1$ to the current time step k:

$$u_{\mathtt{vel}}(\mathcal{B}^{(i)}) = y\left(\frac{\mathtt{vmax}_\zeta(\mathcal{B}^{(i)}) - \mathtt{vmax}_\zeta(\mathcal{B}_n^{\mathtt{N}}(k-1))}{\overline{a}_{n,\zeta_n} \cdot \Delta t} \right),$$

where $\mathtt{vmax}_\zeta(\square)$ returns the maximum velocity in ζ_n-direction of sets $\square$. We use the generalized logistic function y to scale the utility between $(0, 1]$.

Furthermore, we evaluate the covered distance in longitudinal direction from time step $k-1$ to k, since the cooperative vehicles should move forward along their reference path:

$$u_{\mathtt{range}}(\mathcal{B}^{(i)}) = y\left(\frac{\mathtt{pmax}_\zeta(\mathcal{B}^{(i)}) - \mathtt{pmax}_\zeta(\mathcal{B}_n^{\mathtt{N}}(k-1))}{\overline{v}_{n,\zeta_n} \cdot \Delta t + \frac{1}{2} \cdot \overline{a}_{n,\zeta_n} \cdot \Delta t^2} \right),$$

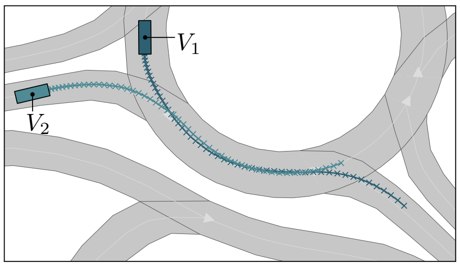
where $\mathtt{pmax}_\zeta(\square)$ returns the maximum longitudinal position of sets $\square$. Again, we use the generalized logistic function y to scale the utility between $(0, 1]$.

2) Survival Mode: We introduce two countermeasures to avoid that the reachable sets of vehicles vanish at a certain time instance: 1) if vehicle V_n has a reachable cell $g_i \in \mathcal{C}_j$ and $\mathtt{area}(\mathcal{B}_n^{\mathtt{CF}}(k)) \leq \underline{A}$, no other vehicle V_m with $\mathtt{area}(\mathcal{B}_m^{\mathtt{CF}}(k)) > \underline{A}$ is allowed to bid on package $\mathcal{C}_j$; 2) we switch the utility function as shown in (7) to:

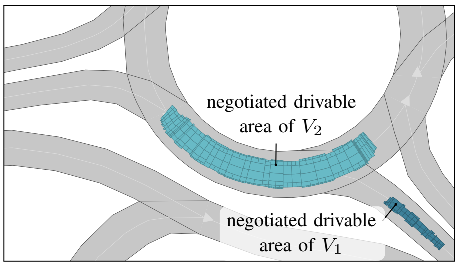
$$U_n^{\mathtt{S}}(k, \mathcal{C}_j) = \frac{\mathtt{area}(\mathcal{B}_n^{\mathtt{C}}(k, \mathcal{C}_j))}{\mathtt{area}(\mathcal{B}_n(k))}.$$

B. Tie-Breaking

Tie-breaking must be performed when multiple vehicles V_n bid $b_n(k, \mathcal{C}_j) = \bar{b}(k, \mathcal{C}_j)$, since this means that several optimal allocations of package $\mathcal{C}_j$ exist. In this paper, we accept the bid of the vehicle with the largest conflicting drivable area $\mathtt{area}(\mathcal{B}_n(k) \setminus \mathcal{B}_n^{\mathtt{CF}}(k))$; if there is a tie again,



(a) Vehicles V_1 and V_2 at time step $k = 0$ and their planned trajectories within the negotiated road areas of the second run.


negotiated drivable area of V_2
negotiated drivable area of V_1

(b) Negotiated drivable areas of vehicles V_1 and V_2 at time step $k = 55$ (first run).

Fig. 3. Scenario I: Roundabout. The driving direction is indicated by the white arrows.

TABLE I
PARAMETERS FOR NUMERICAL EXPERIMENTS.

parameter		**scenario identifier**			
symbol	**unit**	I	II	III	IV
Δt	[s]	0.1	0.1	0.1	0.1
T	/	55	50	45	34
$\overline{v}_{n,\zeta_n}$	[m/s]	15.0	28.0	13.0	18.0
$\underline{v}_{n,\zeta_n}$	[m/s]	4.0	0.0	0.0	0.0
$\overline{v}_{n,\eta_n}$	[m/s]	4.0	6.0	6.0	4.0
$\underline{v}_{n,\eta_n}$	[m/s]	−4.0	−6.0	−6.0	−4.0
$\overline{a}_{n,\zeta_n}$	[m/s^2]	2.5	4.0	4.5	2.0
$\overline{a}_{n,\eta_n}$	[m/s^2]	3.0	6.0	4.5	6.5
$\underline{A}$	[m^2]	0.0	0.0	0.0	0.0
r_n	[m]	1.3	1.2	1.2	1.3

we select the bid randomly.

VI. Evaluation

We demonstrate the universal applicability of our algorithm on four different scenarios. The selected parameters for each scenario can be found in Tab. I and are similar for all cooperative vehicles involved in a traffic scene. Please note that we only depict the drivable area with reference to the vehicle's center of gravity in the following figures to illustrate the available solution space for trajectory planning. To demonstrate that the negotiated road areas can be used for multi-vehicle trajectory planning, we show our first results for each scenario. However, trajectory planning in reachable sets is an ongoing research project and out of scope for this paper.

A. Scenario I: Roundabout

We start with the deliberately simple scenario C-DEU_B471-2_1:2018a from the CommonRoad[1] benchmark collection [24], where two communicating vehicles V_1 and V_2 cooperate such that vehicle V_2 can safely enter the roundabout (see Fig. 3a). Vehicle V_1 plans to take the first exit, while vehicle V_2 aims to take the second exit. A regular grid with tile size 0.5m×0.5m in the Cartesian reference frame F_0 is employed.

Fig. 4 shows the projected reachable sets $\mathcal{B}_n(18)$ of both vehicles and the corresponding conflicting grid cells $\mathcal{G}^{\mathtt{C}}(18)$ at time step $k = 18$ (left) and illustrates the projected negotiated reachable sets $\mathcal{B}^{\mathtt{N}}_n(18)$ (right). Vehicle V_1 accelerates and passes vehicle V_2 which enters the roundabout afterwards. The negotiated drivable areas of the final time step are depicted in Fig. 3b.

[1]http://commonroad.in.tum.de

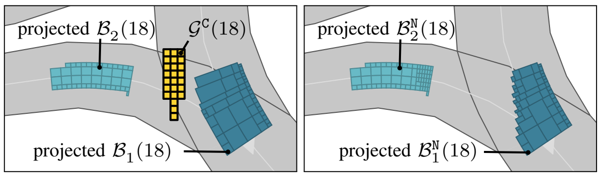

projected $\mathcal{B}_2(18)$
$\mathcal{G}^{\mathtt{C}}(18)$
projected $\mathcal{B}_1(18)$
projected $\mathcal{B}^{\mathtt{N}}_2(18)$
projected $\mathcal{B}^{\mathtt{N}}_1(18)$

Fig. 4. (Left) Projected reachable sets $\mathcal{B}_n(18)$ of V_1 and V_2 and conflicting cells $\mathcal{G}^{\mathtt{C}}(18)$ at time step $k = 18$; (Right) Projected negotiated reachable sets $\mathcal{B}^{\mathtt{N}}_n(18)$.

B. Scenario II: Urban Road

In this scenario (CommonRoad-ID: C-DEU_B471-1_1_T-1:2018a), vehicle V_1 cooperates with the oncoming vehicle V_2 to evade the static obstacle in its lane (see Fig. 5). In Fig. 5, we illustrate the result of the first and second run of the algorithm: vehicle V_1 swerves as soon as vehicle V_2 passes. At time step $k = 12$, we are able to improve the negotiated drivable area by running Alg. 1 a second time, since states of the driving corridor of vehicle V_1 leading to a collision with the static obstacle or with vehicle V_2 in a future time step are removed. Additionally, Fig. 5 visualizes the planned trajectories of vehicles V_1 and V_2 using the negotiated drivable areas of the second run.

C. Scenario III: Crossing

Following the idea of Dresner et al. [4], we allow autonomous vehicles to enter an intersection whenever it is possible. As can be seen in Fig. 6, there are two cooperating vehicles V_1 and V_2; vehicle V_1 intends to turn left, while vehicle V_2 plans to move straight ahead. We show the results after the first run of Alg. 1 in Fig. 6 in the left column. Two driving strategies occur for vehicle V_2: V_2 can either pass through the intersection before vehicle V_1 (see Fig. 6, left column, $k = 30$) or it can stop and wait until V_1 has the left the intersection (see Fig. 6, left column, $k = 45$). In this scenario, it becomes apparent that our set-based method does not only detect different cooperative maneuver options, but

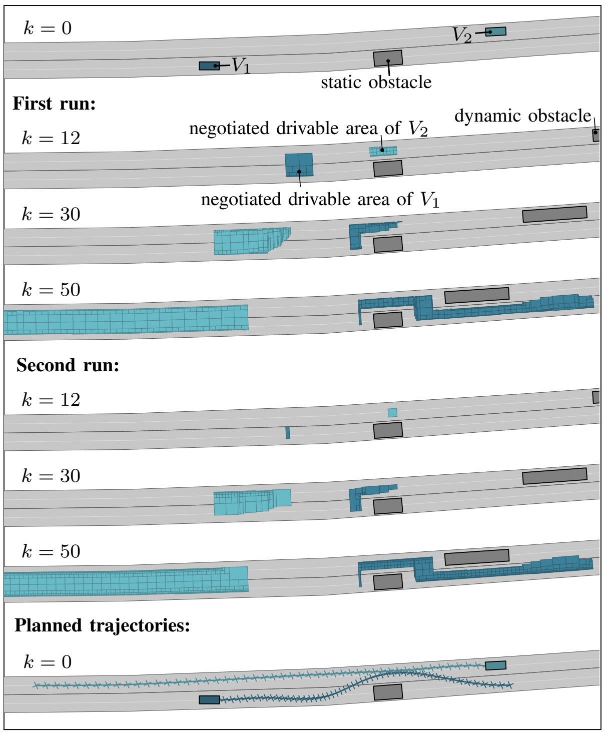


Fig. 5. Scenario II: Urban Road.

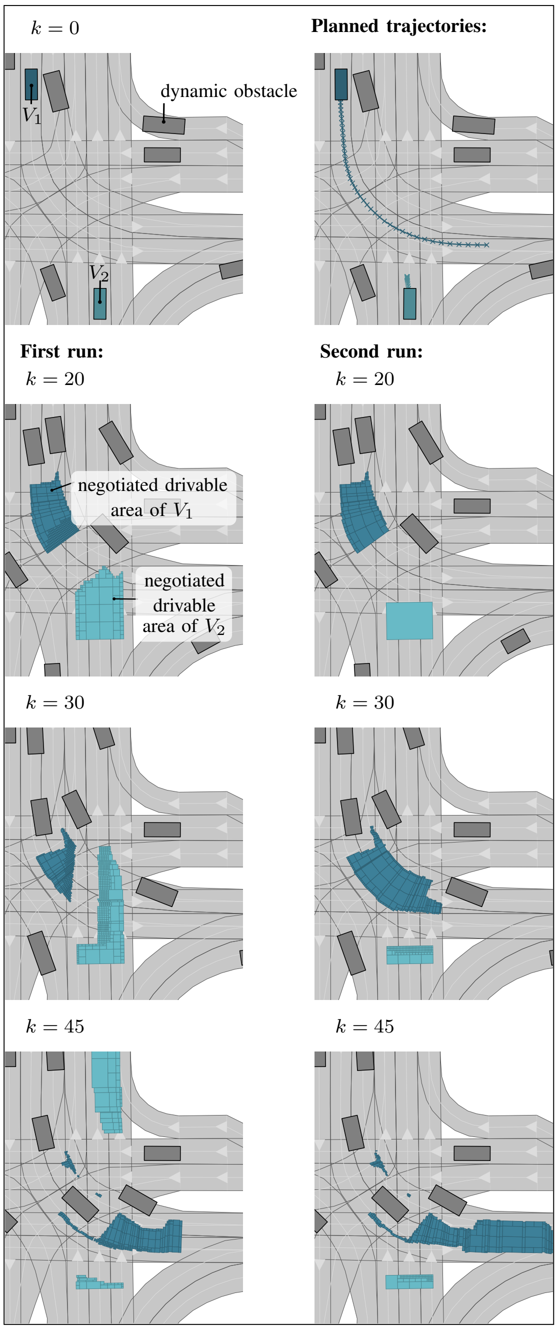


Fig. 6. Scenario III: Crossing. The driving direction is indicated by the white arrows.

also facilitates selecting high-level plans. Driving corridors without bottlenecks are preferred, since they are more robust in terms of unpredicted changes in the environment. In this scenario, crossing the intersection can irritate human drivers and may lead to a collision, since the negotiated driving corridor for this maneuver becomes temporally tight. As illustrated in Fig. 6, the second maneuver variant—stopping at the intersection—is preferred. We therefore restrict the reachable set of V_2 to stopping at the intersection for the second run of Alg. 1 (see Sec. IV-C). However, we do not incorporate information from the first run of Alg. 1 for the reachable set computation of vehicle V_1 during the second run in order to fully utilize the released space of vehicle V_2 (see Fig. 6, right column).

D. Scenario IV: Highway

We apply our algorithm on the mixed-traffic scenario C-NGSIM_US101_1:2017a from the CommonRoad benchmark collection. We coordinate the motion of four cooperating vehicles and restrict their movement to five lanes excluding the highway on-ramp. As can be seen, our method is able to allocate road areas for cooperative vehicles in challenging traffic situations with many non-communicating traffic participants.

VII. Conclusion

We present an approach for negotiating road areas requested by multiple vehicles to determine individual driving corridors for these vehicles. The optimal allocation of offered packages can be performed with polynomial runtime complexity. Since available combinations of assets are matched to the vehicles valuing them the most, the conflict resolution is transparent. This is particularly important when considering legal issues that may arise if sub-optimality is introduced. Future research will focus on the evaluation of different utility functions to determine the bids of the cooperative vehicles.

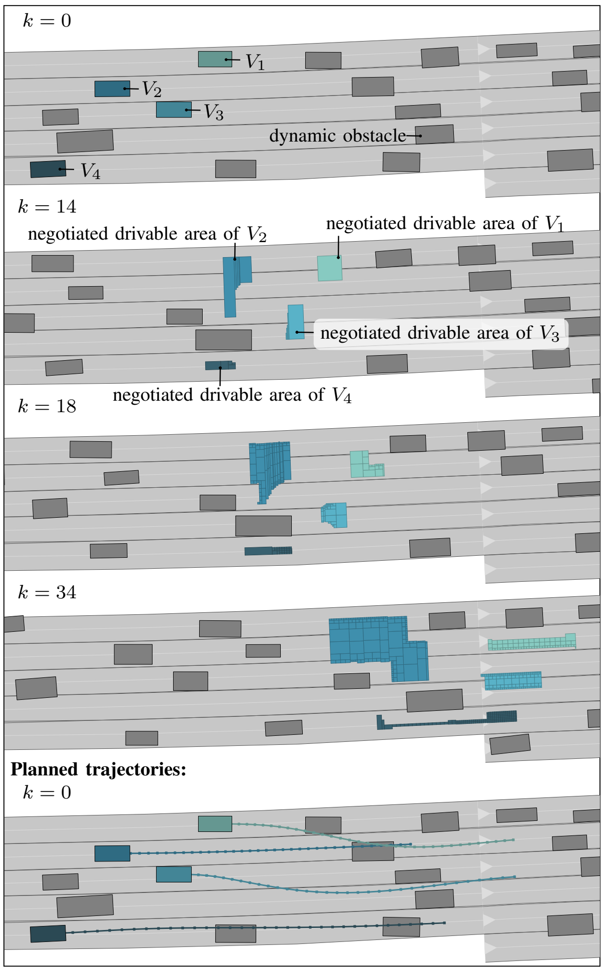

$k = 0$
V_1
V_2
V_3
dynamic obstacle
V_4
$k = 14$
negotiated drivable area of V_2
negotiated drivable area of V_1
negotiated drivable area of V_3
negotiated drivable area of V_4
$k = 18$
$k = 34$
Planned trajectories:
$k = 0$

Fig. 7. Scenario IV: Highway. The driving direction is indicated by the white arrows.

Furthermore, we plan to develop a pricing mechanism for compensating vehicles handing over driving areas.

REFERENCES

[1] K. Dresner and P. Stone, "Multiagent traffic management: a reservation-based intersection control mechanism," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, 2004, pp. 530–537.
[2] ——, "Turning the corner: improved intersection control for autonomous vehicles," in *IEEE Intelligent Vehicles Symposium*, 2005, pp. 423–428.
[3] ——, "Multiagent traffic management: an improved intersection control mechanism," in *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, 2005, pp. 471–477.
[4] ——, "Human-usable and emergency vehicle-aware control policies for autonomous intersection management," in *The 4th Workshop on Agents in Traffic and Transportation*, 2006, pp. 17–25.
[5] G. Sharon and P. Stone, "A protocol for mixed autonomous and human-operated vehicles at intersections," in *Autonomous Agents and Multiagent Systems*, 2017, pp. 151–167.
[6] M. Vasirani and S. Ossowski, "A market-inspired approach for intersection management in urban road traffic networks," *Journal of Artificial Intelligence Research*, vol. 43, pp. 621–659, 2012.
[7] D. Carlino, S. D. Boyles, and P. Stone, "Auction-based autonomous intersection management," in *16th International IEEE Conference on Intelligent Transportation Systems*, 2013, pp. 529–534.
[8] H. Schepperle and K. Böhm, "Auction-based traffic management: Towards effective concurrent utilization of road intersections," in *10th IEEE Conference on E-Commerce Technology and the 5th IEEE Conference on Enterprise Computing, E-Commerce and E-Services*, 2008, pp. 105–112.
[9] F. Zhu and S. V. Ukkusuri, "A linear programming formulation for autonomous intersection control within a dynamic traffic assignment and connected vehicle environment," *Transportation Research Part C: Emerging Technologies*, vol. 55, pp. 363 – 378, 2015.
[10] A. de La Fortelle, "Analysis of reservation algorithms for cooperative planning at intersections," in *13th International IEEE Conference on Intelligent Transportation Systems*, 2010, pp. 445–449.
[11] M. W. Levin, H. Fritz, and S. D. Boyles, "On optimizing reservation-based intersection controls," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 3, pp. 505–515, 2017.
[12] L. C. Bento, R. Parafita, S. Santos, and U. Nunes, "Intelligent traffic management at intersections: Legacy mode for vehicles not equipped with V2V and V2I communications," in *16th International IEEE Conference on Intelligent Transportation Systems*, 2013, pp. 726–731.
[13] L. Chen and C. Englund, "Cooperative intersection management: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 570–586, 2016.
[14] D. Marinescu, J. Čurn, M. Bouroche, and V. Cahill, "On-ramp traffic merging using cooperative intelligent vehicles: A slot-based approach," in *15th International IEEE Conference on Intelligent Transportation Systems*, 2012, pp. 900–906.
[15] D. Marinescu, J. Čurn, M. Slot, M. Bouroche, and V. Cahill, "An active approach to guaranteed arrival times based on traffic shaping," in *13th International IEEE Conference on Intelligent Transportation Systems*, 2010, pp. 1711–1717.
[16] J. Rios-Torres and A. A. Malikopoulos, "A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1066–1077, 2017.
[17] K. Zhang, A. Yang, H. Su, A. de La Fortelle, K. Miao, and Y. Yao, "Service-oriented cooperation models and mechanisms for heterogeneous driverless vehicles at continuous static critical sections," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 7, pp. 1867–1881, 2017.
[18] K. Zhang, A. Yang, H. Su, A. de La Fortelle, and X. Wu, "Unified modeling and design of reservation-based cooperation mechanisms for intelligent vehicles," in *IEEE 19th International Conference on Intelligent Transportation Systems*, 2016, pp. 1192–1199.
[19] S. Manzinger and M. Althoff, "Negotiation of drivable areas of cooperative vehicles for conflict resolution," in *IEEE 20th International Conference on Intelligent Transportation Systems*, 2017, pp. 1388–1395.
[20] M. H. Rothkopf, A. Pekeč, and R. M. Harstad, "Computationally manageable combinational auctions," vol. 44, no. 8, pp. 1131–1147, 1998.
[21] P. Cramton, Y. Shoham, and R. Steinberg, *Combinatorial Auctions*. MIT Press, 2006.
[22] E. Héry, S. Masi, P. Xu, and P. Bonnifait, "Map-based curvilinear coordinates for autonomous vehicles," in *IEEE 20th International Conference on Intelligent Transportation Systems*, 2017, pp. 1699–1705.
[23] S. Söntges and M. Althoff, "Computing the drivable area of autonomous road vehicles in dynamic road scenes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 6, pp. 1855–1866, 2018.
[24] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 719–726.

A.6 CommonRoad: Composable Benchmarks for Motion Planning on Roads [6]

Summary In order to increase the reproducibility of scientific results in publications and to help accelerating the motion planning research, we have developed the CommonRoad benchmark suite for motion planning of road vehicles. A CommonRoad benchmark consists of several components, namely the vehicle model to determine the drivability of planned motions, the cost function to be evaluated, and a specific traffic scenario available in our public database. The traffic scenarios in our database are partly handcrafted, but are also based on real traffic that has been recorded. Each scenario also contains a planning problem for one or multiple vehicles that needs to be solved. CommonRoad defines a standardized file format for the precise description of traffic scenarios using XML, thus ensuring platform-independence.

Researchers can use their motion planning module to solve a benchmark and upload their results to our CommonRoad website for comparison with existing solutions. Since each benchmark is fully described by an unique ID that encodes all required information, reproducibility is greatly improved and the required details of test cases no longer need to be tediously described in publications. CommonRoad provides realistic vehicle dynamics parameters and realistic traffic situations, and offers the freedom to compose a benchmark tailored to the user's own needs.

Author Contributions M. A. initiated the idea of creating benchmarks for the motion planning on roads. **S. M.**, M. A., and M. K. developed the concept of composable benchmarks. M. A. selected the vehicle models. **S. M.** and M. A. specified the standardized cost functions. **S. M.** and M. K. developed the scenario specification. M. K. coordinated the generation of the scenario database. M. A. predominately wrote the conference paper with the support of **S. M.** and M. K. M. A. led the research project.

Conference paper The author's version of the conference paper is reprinted in this thesis. The final version of record is available at `http://dx.doi.org/10.1109/IVS.2017.7995802`.

CommonRoad: Composable Benchmarks for Motion Planning on Roads

Matthias Althoff, Markus Koschi, and Stefanie Manzinger

***Abstract*— Numerical experiments for motion planning of road vehicles require numerous components: vehicle dynamics, a road network, static obstacles, dynamic obstacles and their movement over time, goal regions, a cost function, etc. Providing a description of the numerical experiment precise enough to reproduce it might require several pages of information. Thus, only key aspects are typically described in scientific publications, making it impossible to reproduce results—yet, reproducibility is an important asset of good science. Composable benchmarks for motion planning on roads (CommonRoad) are proposed so that numerical experiments are fully defined by a unique ID; all information required to reconstruct the experiment can be found on the CommonRoad website. Each benchmark is composed by a vehicle model, a cost function, and a scenario (including goals and constraints). The scenarios are partly recorded from real traffic and partly hand-crafted to create dangerous situations. We hope that CommonRoad saves researchers time since one does not have to search for realistic parameters of vehicle dynamics or realistic traffic situations, yet provides the freedom to compose a benchmark that fits one's needs.**

I. Introduction

Reproducibility of results is a cornerstone of science [1], [2]. One obstacle towards reproducibility in motion planning of road vehicles is that details of the experimental results are often not fully provided—some reasons are page limitations of publications, an overwhelming number of required details, or simply because some details are taken for granted. Providing detailed benchmarks would help in this regard and also simplify comparing different planning methods.

First attempts to improve reproducibility and comparability of motion planning algorithms have been made in the robotics community, but mostly for (mobile) robotic manipulators and not for motion planning in the automotive sector. This work provides the first benchmark collection for motion planning on roads, which specifies in depth the motion planning problem consisting of initial state, goal region, road network, static and dynamic obstacles, and the model of the ego vehicle (vehicle for which motion planning is conducted). Before highlighting the main features of CommonRoad, we present a literature review that is categorized into *benchmark problems*, *datasets*, and *motion planning libraries*. Most previous work in robotic motion planning focused on providing libraries that facilitate benchmarking, without providing a set of benchmark problems in a standardized form. We address this problem by providing composable benchmarks that can be referenced to with a unique ID. Our proposed collection also facilitates benchmarking, but this paper does not provide performance metrics—this should be better determined by workshops to reach consensus.

a) Benchmarks: We would first like to note that we only reference benchmarks that are still publicly available. The need of benchmarks in robotics is formulated in [3], but this early work does not provide a specific benchmark. Several European projects for benchmarking in robotics have been conducted in the 2000s (e.g. [4]–[6]), but none has considered motion planning on roads. Detailed benchmarks have been developed in particular for robotic grasping [7], [8] and for robotic manipulators with a focus on indoor human environments [9]. More abstract benchmark problems for motion planning are provided by the Texas A&M University[1] and by Rice University[2].

b) Datasets: While no benchmarks for motion planning on roads exist, recordings of vehicle movements are available; however, none of them is a benchmark problem since initial state, goal regions, and a dynamic vehicle model are missing. Furthermore, there exists no data format commonly used by different research groups. One of the most popular datasets of recorded traffic participants is from the *Next Generation Simulation (NGSIM)* program [10], [11]. Other datasets exist, but they have not recorded all relevant vehicles in a common reference frame, see e.g. [12]. Another class of works provides results on recorded data, but the data has never been or is no longer publicly available, e.g. [13]–[15].

c) Motion planning libraries: One of the most successful motion planning libraries in robotics is the *Open Motion Planning Library (OMPL)* [16], which implements many of the most important sampling-based approaches. The OMPL has also been integrated into *MoveIt!* [17], but remains to be a stand-alone software. *MoveIt!* itself is integrated into the *Robot Operating System (ROS)* [18]. Currently, further infrastructure to facilitate benchmarking with OMPL is developed [19]. Earlier libraries for sampling-based motion planning are the *Online, Open-source, Programming System for Motion Planning (OOPS$_{MP}$)* [20] and the *Open Robotics and Animation Virtual Environment (OpenRAVE)* [21] with similar goals as OMPL. Both libraries contain some benchmark problems (none for automated driving), but their focus is on the implementation of planning algorithms. A library implementing graph-based search is the *Search-Based*

*This work was supported by the Deutsche Forschungsgemeinschaft (German Research Foundation) within the Priority Programme SPP 1835 Cooperative Interacting Automobiles (grant number: AL 1185/4-1) and by the BMW Group within the Car@TUM project.

All authors have equally contributed to this work and are with Faculty of Informatics, Technische Universität München, 85748 Garching, Germany {althoff, markus.koschi, stefanie.manzinger}@tum.de

[1]parasol.tamu.edu/groups/amatogroup/benchmarks
[2]plannerarena.org

Planning Library (SBPL)[3], which is useful if one e.g. uses motion primitives that span a search tree [22]. Besides graph-based techniques, there also exists the *Covariant Hamiltonian Optimization for Motion Planning (CHOMP)* library for gradient-based optimization techniques [23].

d) Automotive benchmarks beyond motion planning: One of the most successful automotive benchmarks is the KITTI benchmark targeting computer vision [24]. Another important aspect is simultaneous localization and mapping; the OpenSLAM[4] project and the Radish project[5] host a collection of benchmarks and libraries for SLAM.

e) Novelty and key features: CommonRoad is a benchmark collection for motion planning of road vehicles (available at **commonroad.in.tum.de**) with the following features:

- **Reproducibility/unambiguity:** All information required to reproduce the results of a motion planner is provided in an unambiguous way and explained by manuals on our website.
- **Composability:** Our benchmarks are composed of vehicle models, cost functions, and scenarios (including goals and constraints). All components are carefully chosen to easily combine and interchange them.
- **Representativeness:** Our benchmark problems contain recorded traffic to faithfully represent real traffic and hand-crafted problems since most recorded traffic situations are not critical/dangerous.
- **Portability:** We use XML to describe our scenarios, which is platform-independent. We also provide executable vehicle models implemented in MATLAB and Python, which are also platform-independent.
- **Scalability:** Our benchmark examples range from simple static scenarios with a few obstacles and a large driving corridor (i.e. region where collisions cannot take place) to complex scenarios with many dynamic obstacles and a small driving corridor.
- **Openness:** All benchmarks are freely available from our website with the possibility to suggest new ones.
- **Independence:** Our benchmarks are independent from planning libraries and our scenario representation could serve as an interchange format between other tools.

II. Benchmark Composition and Planning Problem

As previously mentioned, we compose benchmarks using *vehicle models*, *cost functions*, and *scenarios (including goals and constraints)*. This modularity makes it easy to generate many benchmarks from a smaller set of components and also simplifies comparing the effects of vehicle models or cost functions by only changing those components.

A. Benchmark Composition

Let us introduce with M, C, S, and B the respective IDs of the model, the cost function, the scenario, and the benchmark. The benchmark ID is constructed by separating partial IDs by colons in the following order:

$$\text{B} = \text{M:C:S}.$$

For instance, for M=PM2, C=JB1, S=OV001, the benchmark ID is B = PM2:JB1:OV001. If using one's own component is preferred, one can use the ID IND (for individual). For instance, if one uses an individual cost function for the previous example, the ID becomes PM2:IND:OV001. If one prefers to build upon an existing component, which is modified, the new ID should have *M-* as a prefix. After modifying the model of the first example, the new ID is M-PM2:JB1:OV001 (of course, the modification should be described in detail). In case a collaborative planning benchmark is composed, we list the models and cost functions for n controllable traffic participants by n-dimensional lists:

$$\text{B} = [\text{M}_1,\ldots,\text{M}_n]:[\text{C}_1,\ldots,\text{C}_n]:\text{C-S},$$

where the index refers to the vehicle and a collaborative scenario is indicated by using the prefix *C-*, where *M-* should be used first if both are required. The i-th model and cost function corresponds to the i-th ego vehicle specified in the scenario XML file as introduced later in Sec. V-C. If only one model and/or cost function is used, it is assumed that all controlled vehicles use the same one. For instance, the benchmark ID for M_1=PM1, C_1=JB1, M_2=PM3, C_2=JB1, M_3=ST2, C_3=SA1, and S=C-OV011, is B = [PM1,PM3,ST2]:[JB1,JB1,SA1]:C-OV011.

B. Motion Planning Problem

The proposed benchmarks codify an optimization problem whose solution is the motion plan. Let us denote by $f_M(x(t), u(t))$ the right hand side of the state space model of vehicle model M so that

$$\dot{x}(t) = f_M(x(t), u(t)), \tag{1}$$

where $x \in \mathbb{R}^n$ is the state vector and $u \in \mathbb{R}^m$ is the input vector. We further require the initial state $x_{0,S} \in \mathbb{R}^n$ ($x(t_0) = x_{0,S}$) provided by scenario S, the initial time t_0, and the final time t_f. More details on the models can be found in Sec. III. The cost function J_C of ID C consisting of terminal costs Φ_C and running costs L_C is

$$\begin{aligned} &J_C(x(t), u(t), t_0, t_f) \\ &= \Phi_C(x(t_0), t_0, x(t_f), t_f) + \int_{t_0}^{t_f} L_C(x(t), u(t), t)\, \mathtt{d}t, \end{aligned}$$

which is detailed in Sec. IV. We denote the time-varying, free drivable space on the road surface as $\mathcal{W}_{\mathtt{S,free}}(t) \subset \mathbb{R}^2$ and introduce $O(x(t)) : \mathbb{R}^n \to P(\mathbb{R}^2)$ ($P()$ returns the power set) as the function that returns the occupancy of a vehicle given its state. A possible solution has to ensure that the occupancy of the vehicle is in the free space ($\forall t \in [t_0, t_f] : O(x(t)) \in \mathcal{W}_{\mathtt{S,free}}(t)$) and respects additional constraints $g_S(x(t), u(t), t) \leq 0$ provided by scenario S, such as speed limits or other traffic rules [25]. Equality constraints can be

[3]wiki.ros.org/sbpl
[4]www.openslam.org
[5]radish.sourceforge.net

constructed from inequality constraints (e.g. $x \leq 0 \wedge -x \leq 0 \equiv x = 0$). Let us further denote the goal region $\mathcal{G}_S \subset \mathbb{R}^n$ of scenario S, which can be disjoint sets (see Sec. V-C). As soon as $x(t) \in \mathcal{G}_S$ at time $t = t_f$, a feasible solution is found. After introducing an input trajectory as $u(\cdot)$ (in contrast to a value $u(t)$ at time t), we can finally formulate the motion planning problem as finding

$$u^*(\cdot) = \operatorname*{arg\,min}_{u(\cdot)} J_C(x(t), u(t), t_0, t_f) \tag{2}$$

subject to

$$\begin{aligned} &\dot{x}(t) = f_M(x(t), u(t)), \quad O(x(t)) \in \mathcal{W}_{\mathtt{S,free}}(t), \\ &g_S(x(t), u(t), t) \leq 0, \quad x(t_0) = x_{0,S}, \quad x(t_f) \in \mathcal{G}_S. \end{aligned}$$

Associated with the optimal input trajectory $u^*(\cdot)$ in (2) is an optimal state trajectory $x^*(\cdot)$ that can be obtained by a forward simulation of (1). Directly solving (2) is referred to as *trajectory planning* (see [26, Sec. 4.]). An alternative is to first find a path that the vehicle should follow for which an optimal velocity profile is computed, which we refer to as *path planning* with subsequent *velocity optimization* (see [26, Sec. 4.]). Both techniques can be used to solve our proposed benchmarks.

III. Vehicle Models

This section presents models for vehicle dynamics ranging from simple to complex. For each model it is assumed that underlying controllers exist that can realize a commanded acceleration (positive and negative within given limits). For adaptive cruise control in particular, numerous works already exist that realize a commanded acceleration, see e.g. [27], [28]. The effects of engine characteristics in terms of fuel consumption can be considered in the cost function (see Sec. IV).

The lateral dynamics, however, cannot be abstracted away to the same extent using controllers, especially when constraints such as the danger of roll-over must be considered in extreme maneuvers [29], [30]. For this reason, our models consider increasingly complex lateral vehicle dynamics and tire models: point-mass model, kinematic single-track model, single-track model, and a multi-body model. Some details of the first two models are presented subsequently, whereas due to space restrictions, the full detailed description of the single-track model and the multi-body model can be found in our *vehicle model documentation* on our website. Executable MATLAB and Python implementations of all presented models are also available. We have not included Dubin or Reeds-Shepp cars since they require changing the steering angle infinitely fast (see e.g. [31]).

The model IDs are constructed by first choosing the model type (e.g. ST for single-track) followed by a number, which refers to the parameterization in the *vehicle model documentation* of our repository.

A. Point-Mass Model (M=PM)

The point-mass model is the simplest model that is commonly used for motion planning, see e.g. [32], [33]. This model abstracts the vehicle as a point mass whose absolute acceleration is bounded (Kamm's circle). Let us introduce $\square$ as the placeholder for a variable and $\square_x$ and $\square_y$ to denote the value of the corresponding variable in x and y direction (world coordinates), respectively. After further introducing position s, acceleration a, and maximum absolute acceleration $a_{\mathtt{max}}$, the dynamics is

$$\ddot{s}_x = a_x, \quad \ddot{s}_y = a_y, \quad \sqrt{a_x^2 + a_y^2} \leq a_{\mathtt{max}}.$$

The point-mass model ignores the minimum turning circle, which is considered next in the kinematic single-track model.

B. Kinematic Single-Track Model (M=KS)

The kinematic single-track model (also known as the kinematic bicycle model) considers only two wheels, where the front and rear wheel pairs are each lumped into one wheel, because the roll dynamics is neglected (see Fig. 1 and [34, Sec. 2.2]). This also explains the term *single-track model*. Tire slip is not considered, but the kinematic single-track model can be used when the vehicle does not operate close to its physical capabilities [26], [35]. For instance, when planning a parking maneuver, tire slip is not important, but the point-mass model would not be sufficient since the non-holonomic behavior and, in particular, the minimum turning radius would not be considered.

In addition to the variables already introduced, we also require the velocity of the steering angle v_δ, the steering angle δ, the heading Ψ, and the parameter l describing the wheelbase as well as the parameter v_S describing the velocity above which the engine power is limiting maximum positive acceleration rather than maximum tire forces (see Fig. 1). We further denote by $\underline{\square}$ the minimum possible value, by $\overline{\square}$ the maximum possible value, by $\square_{\mathtt{lat}}$ the value of a variable in lateral direction, and by $\square_{\mathtt{long}}$ the value in longitudinal direction (vehicle-fixed coordinates). The differential equations of the kinematic single-track model as defined in this work are

$$\begin{aligned} &\dot{\delta} = v_\delta, \quad \dot{\Psi} = \frac{v}{l}\tan(\delta), \quad \dot{v} = a_{\mathtt{long}}, \\ &\dot{s}_x = v\cos(\Psi), \quad \dot{s}_y = v\sin(\Psi), \end{aligned}$$

under consideration of the constraints

$$v_\delta \in [\underline{v}_\delta, \overline{v}_\delta], \quad \delta \in [\underline{\delta}, \overline{\delta}], \quad v \in [\underline{v}, \overline{v}], \tag{3}$$

$$a_{\mathtt{long}} \in [-a_{\mathtt{max}}, \overline{a}], \quad \overline{a} = \begin{cases} a_{\mathtt{max}} \frac{v_S}{v} & \text{for } v > v_S, \\ a_{\mathtt{max}} & \text{otherwise,} \end{cases} \tag{4}$$

$$\sqrt{a_{\mathtt{long}}^2 + (v\,\dot{\Psi})^2} \leq a_{\mathtt{max}} \qquad (a_{\mathtt{lat}} = v\,\dot{\Psi}). \tag{5}$$

Constraint (3) considers that the steering velocity, the steering angle, and the vehicle velocity are bounded. Limited engine power and braking power as detailed in [36, Sec. III.B] are considered by (4). Finally, as in the point-mass model, constraint (5) models Kamm's circle.

Note that kinematic single-track models differ slightly in publications, depending on whether one considers that 1) the steering angle or the steering velocity is an input, 2) the vehicle velocity or the vehicle acceleration is an input, or 3) the front or rear wheel is the reference point (here: rear wheel, see Fig. 1). For instance, in [26, eq. (8)], the vehicle velocity and the steering velocity are inputs. Additionally, other works do not provide all the constraints of our model (which can be easily removed, but a removal should be stated since this simplifies motion planning).

C. Single-Track Model (M=ST)

The natural extension of the kinematic single-track model is the single-track model (also known as the bicycle model), which considers tire slip [34, Sec. 2.3] influencing the slip angle β, which is illustrated in Fig. 1 as the angle between the velocity vector v and the vehicle orientation Ψ. Works that perform planning of evasive maneuvers closer to physical limits require the single-track model, see e.g. [37], [38]. We additionally consider the load transfer of the vehicle due to longitudinal acceleration $a_{\texttt{long}}$ (neglecting suspension dynamics). Due to space limitations, we refer the reader to our *vehicle model documentation* for a detailed description and derivation of the single-track model.

Since the single-track model uses a linear relationship between slip angle and tire force (thus ignoring saturation effects), constraint (5) is important for limiting possible tire forces. Please note that in contrast to this work, other works often only consider constant velocity when referring to a single-track model (see e.g. [34, Sec. 2.3]). Also, the weight transfer between the front and rear axle is often neglected in single-track models (see e.g. [37]).

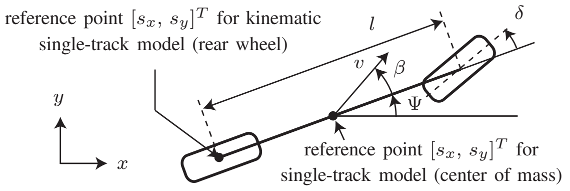

reference point $[s_x,\, s_y]^T$ for kinematic single-track model (rear wheel)
l
δ
v
β
Ψ
y
x
reference point $[s_x,\, s_y]^T$ for single-track model (center of mass)

Fig. 1. Combined illustration of kinematic/standard single-track model.

D. Multi-Body Model (M=MB)

Although the previously introduced single-track model already considers many important effects of vehicle dynamics, it does not consider the vertical load of all 4 wheels due to roll, pitch, and yaw, their individual spin and slip, and nonlinear tire dynamics. An example of a multi-body model used for motion planning of a road vehicle can be found in [39]. Although many commercial multi-body models for vehicle dynamics exist[6], those models are proprietary and thus not appropriate for a benchmark that requires public accessibility. Our multi-body model is taken out of [40, Appendix A], which is one of few detailed and accessible multi-body dynamics descriptions. Due to the complexity of the multi-body model, we refer to the *vehicle model documentation* of our repository and only mention the main features.

[6]www.carsim.com, www.tesis-dynaware.com, www.mscsoftware.com

The multi-body dynamics is described by 3 masses: The unsprung mass and the sprung masses of the front and rear axles. The forces between these masses are described by the dynamics of the suspension and the tire model. We consider all suspension forces in [40, Appendix A] originating from springs, dampers, and anti-roll bars. For the tire dynamics we use the PAC2002 Magic-Formula tire model, which is widely used in industry [41]. Rewriting all equations as a state space model yields 29 state variables.

E. Numerical Experiments and Interchangeability of Models

In order to facilitate switching between different models and to compare results as done in this subsection, we describe in our *vehicle model documentation* how parameter sets and initial states can be converted in the best possible way between models. There, we further provide state-space formats of all models so that it is easier to build one's own executable models in addition to the ones in MATLAB and Python.

To illustrate better differences between models, we briefly present numerical experiments for a BMW 320i (parameter set 2 in *vehicle model documentation*). The duration of each experiment is 1 s, and the initial velocity is 15 m/s; further details of the experiments can be found in the *vehicle model documentation*. First, we compare the kinematic single-track model, the single-track model, and the multi-body model when driving a left curve. It can be easily seen in Fig. 2(a) that the kinematic single-track model realizes the tightest bend since it does not consider tire slip; the single-track model is a little wider due to considering tire slip. This effect is even stronger for the multi-body model since it already considers saturation of tire forces before constraint (5) is active. This can be seen even better when comparing the slip angles of the single-track model and the multi-body model in Fig. 2(b).

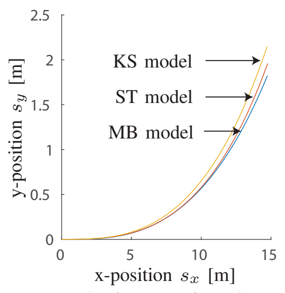
Second, we demonstrate understeering and oversteering (see [34, Sec. 3.3]) for the multi-body model during cornering by braking into the corner ($a_{\texttt{long}} = -0.7$ g, g represents the gravity constant), coasting ($a_{\texttt{long}} = 0$ g), and heavily accelerating ($a_{\texttt{long}} = 0.63$ g) the rear-wheel-driven vehicle (power oversteer) as shown by the slip angle in Fig. 3(a). It can also be easily observed, by plotting the pitch in Fig. 3(b), that the vehicle is "diving" during braking while the front lifts during acceleration.

IV. Cost Functions

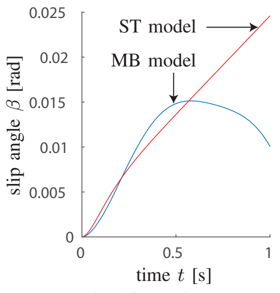
This section proposes standardized cost functions for the motion planning problem in (2). Analogously to the composability of the benchmarks, we compose different types of partial cost functions to a single cost function. The partial cost functions have a unique ID p and the set $\mathcal{P}$ contains all IDs of the proposed partial cost functions. The overall cost function is obtained by the weighted sum

$$J_C(x(t), u(t), t_0, t_f) = \sum_{i \in \mathcal{I}} w_i \, J_i(x(t), u(t), t_0, t_f),$$

where $\mathcal{I} \subset \mathcal{P}$ contains the IDs of the applied partial cost functions and $w_i \in \mathbb{R}^+$ are weights. We first present popular

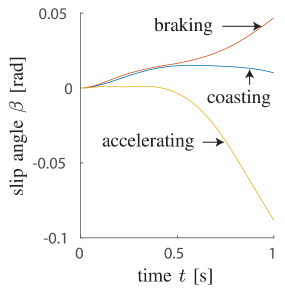


(a) Path of center of gravity.

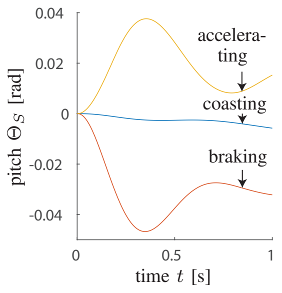


(b) Slip angle.

Fig. 2. Comparing the kinematic single-track (KS) model, the single-track (ST) model, and the multi-body (MB) model during cornering.



(a) Slip angle.



(b) Pitch.

Fig. 3. Investigating oversteering and understeering as well as pitch for the multi-body model.

partial cost functions using the variables already introduced in Sec. III:

- **Time**: $J_T = t_f$ (see [42, eq. 2]).
- **Acceleration**: $J_A = \int_{t_0}^{t_f} a(t)^2 \, \mathrm{d}t$ (see [43, Sec. III.B]).
- **Jerk**: $J_J = \int_{t_0}^{t_f} \dot{a}(t)^2 \, \mathrm{d}t$ (see [44, Sec. III]).
- **Steering angle**: $J_{SA} = \int_{t_0}^{t_f} \delta(t)^2 \, \mathrm{d}t$ (see [45]).
- **Steering rate**: $J_{SR} = \int_{t_0}^{t_f} v_\delta(t)^2 \, \mathrm{d}t$ (see [45]).
- **Energy**: $J_E = \int_{t_0}^{t_f} P(x(t), u(t)) \, \mathrm{d}t$, where $P(x(t), u(t))$ is the required power of the engine for the state x and the input u, which can be obtained from engine mappings (see [28, Sec. III.B]).
- **Yaw rate**: $J_Y = \int_{t_0}^{t_f} \dot{\Psi}(t)^2 \, \mathrm{d}t$ (see [43, Sec. III.B]).
- **Lane center offset**: $J_{LC} = \int_{t_0}^{t_f} d(t)^2 \, \mathrm{d}t$, where d is the distance to the lane center or a driving corridor (see [43, Sec. III.B]).
- **Velocity offset**: $J_V = \int_{t_0}^{t_f} (v_{\mathtt{des}}(x(t)) - v(t))^2 \, \mathrm{d}t$, where $v_{\mathtt{des}}(x(t))$ is the desired velocity for the vehicle state x (see [43, Sec. III.B]).
- **Orientation offset**: $J_O = \int_{t_0}^{t_f} (\theta_{\mathtt{des}}(x(t)) - \theta(t))^2 \, \mathrm{d}t$, where $\theta_{\mathtt{des}}(x(t))$ is the desired orientation for the vehicle state x (see [45]).
- **Distance to obstacles**: $J_D = \int_{t_0}^{t_f} \max(\xi_1(t), \ldots, \xi_o(t)) \, \mathrm{d}t$, where o is the number of obstacles, $\xi_i(t) = e^{-w_{\mathtt{dist}} d_i(t)}$, $d_i(t)$ is the distance of the ego vehicle to an obstacle, and $w_{\mathtt{dist}}$ is an additional required weight (see [46, eq. 7-8]).
- **Path length**: $J_L = \int_{t_0}^{t_f} v(t) \, \mathrm{d}t$ (see [46, Tab. 1]).
- **Terminal offset**: $J_{TO} = d(t_f)^2$ (see [44, eq. 2]).
- **Terminal distance to goal**: $J_{TG} = d_{\mathtt{goal}}(t_f)^2$, where $d_{\mathtt{goal}}$ is the distance to the goal (see [47, Sec. IV.D]).

Let us now introduce a notation for writing the used weights compactly. We write $w_T = 0.1$, $w_{SA} = 0.4$, and $w_Y = 0.7$ in short as $[(T|0.1), (SA|0.4), (Y|0.7)]$. After agreeing that we use SI units for all variables, this notation uniquely defines a cost function. Most works, however, do not provide such weights, so we cannot include their values in the current version of the benchmark. We therefore hope that once the structure is fixed, other researchers will contribute their used weights. Works that published their used weights are listed below, where the cost function ID is chosen as the initials of the first authors plus a running number:

- J_{JB1} from [42, eq. 2]: $[(T|1)]$.
- J_{SA1} inspired by [48, eq. 2]: $[(SA|0.1), (SR|0.1), (D|10^5)]$ (we use fewer parameters).
- J_{WX1} inspired by [46, Tab. IV]: $[(T|10), (V|1), (A|0.1), (J|0.1), (D|0.1), (LC|10)]$ (we use fewer parameters and velocity difference instead of absolute velocity).

V. Scenarios

As a last component, we introduce scenarios specified by an XML file, which is composed of 1) a formal representation of the road network, 2) static and dynamic obstacles, and 3) the planning problem of the ego vehicle(s) as shown in Fig. 4, where details of child elements are omitted for clarity. In the following subsections we briefly describe each data format in more detail. A detailed description can be found in the *XML documentation* on our website. We also provide a *scenario documentation* listing all available scenarios.

A. Road Network

For our benchmarks we use *lanelets* [49] as atomic, interconnected, and drivable road segments to represent the road network. A lanelet is defined by its *left* and *right bound*, where each bound is represented by an array of points (a polyline), as shown in Fig. 5. We have chosen lanelets since they are as expressive as other formats, such as e.g. OpenDRIVE[7], yet have a lightweight and extensible representation. Using lanelets allows the road network to be modeled as a *directed graph*, where each node has four types of outgoing edges: successor, predecessor, adjacentLeft, and adjacentRight (see Fig. 4; predecessor is not required but added for implementation reasons). Lanelets additionally contain *traffic regulations*, e.g. the speed limit. All road networks are stored using XML. The XML data structure of OpenStreetMap[8] can represent lanelets in the WGS84 coordinate frame using references between lanelets and primitive elements as described in [49]. Since we require Cartesian coordinates and a compact element structure to also represent obstacles and the planning problem, we propose our *CommonRoad* XML data format as specified on our website.

[7]opendrive.org
[8]openstreetmap.org

```
/
├─lanelet
│ ├─leftBound
│ │ ├─point
│ │ └─lineMarking
│ ├─rightBound
│ │ ├─point
│ │ └─lineMarking
│ ├─predecessor (ref to lanelet)
│ ├─successor (ref to lanelet)
│ ├─adjacentLeft (ref to lanelet)
│ ├─adjacentRight (ref to lanelet)
│ └─trafficRegulations
├─obstacle
│ ├─role: static
│ ├─type: parkedVehicle/.../unknown
│ └─shape
├─obstacle
│ ├─role: dynamic
│ ├─type: car/truck/.../unknown
│ ├─shape
│ └─trajectory
│   └─state
├─obstacle
│ ├─role: dynamic
│ ├─type: car/truck/.../unknown
│ └─occupancySet
│   └─occupancy
├─obstacle
│ ├─role: dynamic
│ ├─type: car/truck/.../unknown
│ ├─shape
│ └─probabilityDistribution
└─planningProblem
  ├─initialState
  └─goalRegion
    └─state
```

Fig. 4. Structure of the XML files encoding each scenario. For clarity we do not show all elements of the XML structure.

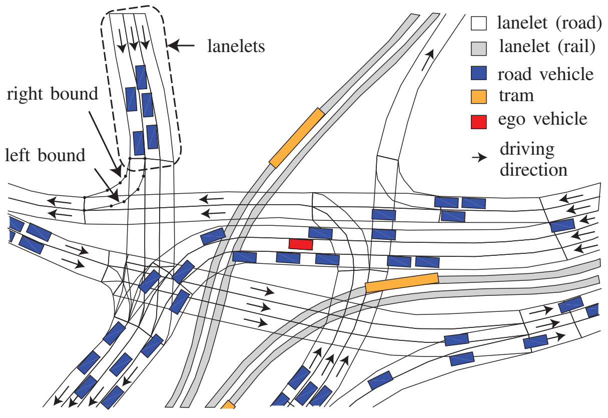


Fig. 5. Lanelets of a complex intersection in the city center of Munich (scenario ID S=GER_Muc_1a). Besides roads, tram rails are also modeled by lanelets.

B. Obstacles

Obstacles are characterized by their role (static/dynamic), type (car/truck/bus/bicycle/pedestrian/constructionZone/parkedVehicle/priorityVehicle/unknown), shape (rectangle/circle/polygon), and movement over time (if the obstacle is dynamic). We have restricted ourself to the shapes rectangle, circle, and polygon since rectangles are a good description for cars and trucks, circles are a good description of pedestrians, and any other two-dimensional shape can be modeled by a polygon if the number of points approaches infinity. If motion planners depend on other representations, one has to enclose the provided shape, see e.g. [50].

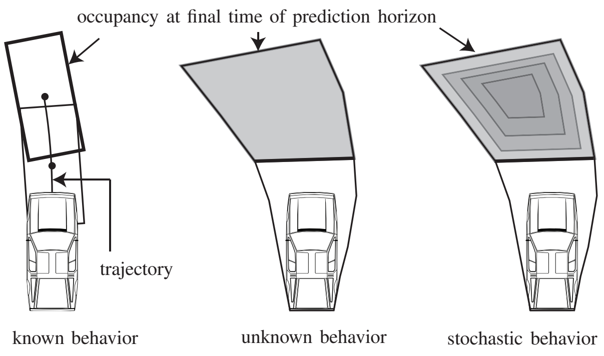


Fig. 6. Supported occupancy representation of predicted obstacle movements.

When the obstacle is dynamic, we provide three possibilities to describe the movement over time as illustrated in Fig. 6: known behavior, unknown behavior bounded by sets, and unknown behavior described by probability distributions.

a) Known behavior: We describe known behavior with a trajectory, which is modeled as state sequence containing position and orientation. After defining the reference points of shapes of obstacles, the occupancy of an obstacle along a trajectory is uniquely defined: the reference point of a rectangle and a circle is their geometric center and the reference point of a polygon is its first point (polygons are stored as an ordered list of points).

b) Unknown behavior: Occupancy sets that evolve over time are used to represent unknown behavior [36]. For occupancy sets we only allow polygons as a representation that can be obtained from our tool SPOT [51]. Please note that one can also represent known behavior by evolving occupancy sets, which do not change their size over time.

c) Unknown stochastic behavior: One can describe unknown stochastic behavior with probability distributions of states. Since many different probability distributions are used (e.g. Gaussian [52], piecewise constant [53], etc.), we provide a placeholder for probability distributions in our XML structure. Please note that for stochastic behavior, the distribution of the state and the dimension of the vehicle have to be stored separately to correctly compute crash probabilities [53, Sec. VI]. For this reason, we also store the shape of obstacles as we do for known behavior.

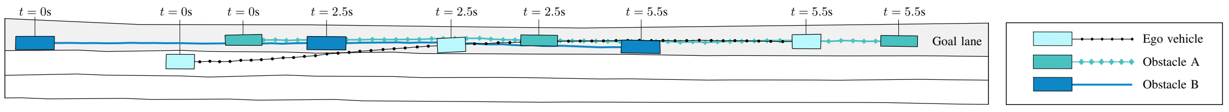


Fig. 7. Solution of our applied trajectory planner for scenario "NGSIM_US101_0".

C. Planning Problem

Each ego vehicle has an initial state as well as one or several goal regions. If several goal regions are provided, we implicitly assume that only one of them has to be reached, modeling options like overtaking or staying behind a vehicle. The position of the goal region is defined by a point, shape (rectangle/circle/polygon), or lanelet. For orientation, velocity, and time, intervals or exact values can be provided. Since different vehicle models can be used (see Sec. III), the shape of the ego vehicle is part of the parameterization of the model. Despite the fact that the different models have different state variables, we can initialize all models by the initial state of a single-track model as described in the *vehicle model documentation*.

VI. Example

We demonstrate our proposed benchmark collection with a deliberately simple scenario, which is based on recorded traffic data from the NGSIM U.S. 101 dataset (07:50 a.m. to 08:05 a.m.). Fig. 7 shows the trajectories of two vehicles and the initial position of the ego vehicle. We consider all lanes of the U.S. 101 highway provided by the NGSIM dataset; however, we only depict three out of six lanes in Fig. 7 for the sake of clarity. The goal of this scenario is to plan a lane-change maneuver for the ego vehicle to the left-most lane within a time horizon of $t_f \in [5.5, 6.0]$ s (see Fig. 7).

The applied trajectory planner is based on numerical optimization; for a detailed explanation of the algorithm, the interested reader is referred to [45, Sec. III.1]. In this paper, we use a kinematic single-track model as described in Sec. III based on the parameters KS1 described in the *vehicle model documentation*. However, in order to demonstrate how parameters can be modified, the parameter v_S is changed to $v_S \rightarrow \infty$. The cost function is chosen as

$$\begin{aligned} J_{SM1}(x(t), u(t), t_0, t_f) = \quad & w_A J_A + w_{SA} J_{SA} \\ & + w_{SR} J_{SR} + w_{LC} J_{LC} + w_V J_V + w_O J_O, \end{aligned}$$

which minimizes the acceleration (J_A), steering effort (J_{SA} and J_{SR}), the distance and orientation offset to a reference path (J_{LC} and J_O), and the velocity offset (J_V). The chosen weights are

$$[(A|50), (SA|50), (SR|50), (LC|1), (V|20), (O|50)].$$

Since the ego vehicle should perform a lane-change to the left lane, the reference path is set to the center of the goal lane for computing the costs J_{LC} and J_O. Furthermore, the optimization horizon is 5.5 s and the desired velocity is $v_{\text{des}} = 25$ m/s.

The unique ID of the benchmark is B = M-KS1:SM1:NGSIM_US101_0, with $v_S \rightarrow \infty$. Our obtained trajectory has a total cost of $J_{SM1}(x(t), u(t), t_0, t_f) = 5.69 \cdot 10^4$. In contrast to other work, all details on the vehicle model, the cost function, and the scenario are precisely given by our unique ID. Please note that without the ID we also would not have had the space to present all the details of the scenario in this work, although it is quite simple.

VII. Conclusions

To the best of our knowledge, we provide the first set of composable benchmark problems for motion planning on roads accessible from **commonroad.in.tum.de**. While this paper only provides a rough overview, all details can be found in the provided documentation on our website. Each composed benchmark has a unique ID that can be used in publications or for one's own organization of benchmarks. This is demonstrated by an example for which we also provide a solution. Our benchmark collection contains a mix of recorded and constructed scenarios as well as scenarios on highways, on rural roads, and in urban settings. Our platform-independent repository can be extended by other researchers and will also be extended by ourselves.

References

[1] F. Amigoni, M. Reggiani, and V. Schiaffonati, "An insightful comparison between experiments in mobile robotics and in science," *Autonomous Robots*, vol. 27, pp. 313–325, 2009.

[2] F. Bonsignorio and A. P. del Pobil, "Toward replicable and measurable robotics research," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 32–35, 2015.

[3] J. Baltes, "A benchmark suite for mobile robots," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000, pp. 1101–1106.

[4] R. Dillmann. (2004) Benchmarks for robotics research. Deliverable KA 1.10 of the EU project EURON.

[5] K. Pfeiffer, A. Bubeck, N. Blümlein, and M. Hägele. (2007) Action plan and recommendation on benchmarks for mobile manipulation and service robots. Deliverable D4.3 of the EU project Robot Standards and Reference Architectures (RoSta).

[6] W. Nowak, A. Zakharov, S. Blumenthal, and E. Prassler. (2010) Benchmarks for mobile manipulation and robust obstacle avoidance and navigation. Deliverable D3.1 of EU Project Best Practice in Robotics (BRICS).

[7] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, "The Columbia Grasp Database," in *Proc. of IEEE International Conference on Robotics and Automation*, 2009, pp. 1710–1716.

[8] S. Ulbrich, D. Kappler, T. Asfour, N. Vahrenkamp, A. Bierbaum, M. Przybylski, and R. Dillmann, "The OpenGRASP benchmarking suite: An environment for the comparative analysis of grasping and dexterous manipulation," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 1761–1767.

[9] B. Cohen, I. A. Şucan, and S. Chitta, "A generic infrastructure for benchmarking motion planners," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 589–595.

[10] V. G. Kovvali, V. Alexiadis, and L. Zhang, "Video-based vehicle trajectory data collection," in *Proc. of the Transportation Research Board 86th Annual Meeting*, 2007.

[11] V. Punzo, M. T. Borzacchiello, and B. Ciuffo, "On the assessment of vehicle trajectory data accuracy and application to the Next Generation SIMulation (NGSIM) program data," *Transportation Research Part C*, vol. 19, pp. 1243–1262, 2011.

[12] E. Romera, L. M. Bergasa, and R. Arroyo, "Need data for driver behaviour analysis? Presenting the public UAH-DriveSet," in *Proc. of the 19th IEEE International Conference on Intelligent Transportation Systems*, 2016, pp. 387–392.

[13] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank, "A system for learning statistical motion patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 1450–1464, 2006.

[14] B. T. Morris and M. M. Trivedi, "Learning, modeling, and classification of vehicle track patterns from live video," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, pp. 425–437, 2008.

[15] Y. Zheng, J. Wang, X. Li, C. Yu, K. Kodaka, and K. Li, "Driving risk assessment using cluster analysis based on naturalistic driving data," in *Proc. of the 17th IEEE International Conference on Intelligent Transportation Systems*, 2014, pp. 2584–2589.

[16] I. A. Şucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[17] S. Chitta, I. Sucan, and S. Cousins, "MoveIt!" *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[18] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[19] M. Moll, I. A. Şucan, and L. E. Kavraki, "Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 96–102, 2015.

[20] E. Plaku, K. E. Bekris, and L. E. Kavraki, "OOPS for motion planning: An online, open-source, programming system," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 3711–3716.

[21] R. Diankov and J. Kuffner, "OpenRAVE: A planning architecture for autonomous robotics," Carnegie Mellon University, Tech. Rep. CMU-RI-TR-08-34, 2008.

[22] E. Frazzoli, M. A. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1077–1091, 2005.

[23] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Proc. of IEEE International Conference on Robotics and Automation*, 2009, pp. 489–494.

[24] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[25] A. Rizaldi and M. Althoff, "Formalising traffic rules for accountability of autonomous vehicles," in *Proc. of the 18th IEEE International Conference on Intelligent Transportation Systems*, 2015, pp. 1658–1665.

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

[27] S. E. Shladover, C. A. Desoer, J. K. Hedrick, M. Tomizuka, J. Walrand, W.-B. Zhang, D. H. McMahon, H. Peng, S. Sheikholeslam, and N. McKeown, "Automated vehicle control developments in the PATH program," *IEEE Transactions on Vehicular Technology*, vol. 40, no. 1, pp. 114–130, 1991.

[28] D. Kim, H. Peng, S. Bai, and J. M. Maguire, "Control of integrated powertrain with electronic throttle and automatic transmission," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 3, pp. 474–482, 2007.

[29] D. Odenthal, T. Bünte, and J. Ackermann, "Nonlinear steering and braking control for vehicle rollover avoidance," in *Proc. of the European Control Conference*, 1999, pp. 598–603.

[30] P. Gaspar, I. Szaszi, and J. Bokor, "Reconfigurable control structure to prevent the rollover of heavy vehicles," *Control Engineering Practice*, vol. 13, pp. 699–711, 2005.

[31] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990.

[32] J.-B. Tomas-Gabarron, E. Egea-Lopez, and J. Garcia-Haro, "Vehicular trajectory optimization for cooperative collision avoidance at high speeds," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1930–1941, 2013.

[33] D. N. Godbole, V. Hagenmeyer, R. Sengupta, and D. Swaroop, "Design of emergency maneuvers for automated highway system: Obstacle avoidance problem," in *Proc. of the 36th Conference on Decision & Control*, 1997, pp. 4774–4779.

[34] R. Rajamani, *Vehicle Dynamics and Control*. Springer, 2012.

[35] S. Petti and T. Fraichard, "Safe motion planning in dynamic environments," in *Proc. of the Conference on Intelligent Robots and Systems*, 2005.

[36] M. Althoff and S. Magdici, "Set-based prediction of traffic participants on arbitrary road networks," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 2, pp. 187–202, 2016.

[37] J. H. Jeon, S. Karaman, and E. Frazzoli, "Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*," in *Proc. of the 50th IEEE Conference on Decision and Control and European Control Conference*, 2011, pp. 3276–3282.

[38] Z. Shiller and Y.-R. Gwo, "Dynamic motion planning of autonomous vehicles," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 2, pp. 241–249, 1991.

[39] E. Bertolazzi, F. Biral, and M. Da Lio, "Real-time motion planning for multibody systems," *Multibody System Dynamics*, vol. 17, no. 2, pp. 119–139, 2007.

[40] R. W. Allen, H. T. Szostak, D. H. Klyde, T. J. Rosenthal, and K. J. Owens, "Vehicle dynamic stability and rollover," U.S. Department of Transportation, Final Report DOT HS 807 956, 1992.

[41] *Adams/Tire help*, MSC Software, 2 MacArthur Place, Santa Ana, CA 92707, April 2011, documentation ID: DOC9805. [Online]. Available: http://simcompanion.mscsoftware.com/infocenter

[42] J. E. Bobrow, "Optimal robot path planning using the minimum-time criterion," *IEEE Journal of Robotics and Automation*, vol. 4, no. 4, pp. 443–450, 1988.

[43] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for BERTHA – a local, continuous method," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2014, pp. 450–457.

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

[45] S. Magdici and M. Althoff, "Fail-safe motion planning of autonomous vehicles," in *Proc. of the 19th International IEEE Conference on Intelligent Transportation Systems*, 2016, pp. 452–458.

[46] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2012, pp. 2061–2067.

[47] J. Suh and S. Oh, "A cost-aware path planning algorithm for mobile robots," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 4724–4729.

[48] S. J. Anderson and S. C. Peters, "An optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios," *International Journal of Vehicle Autonomous Systems*, vol. 8, pp. 190–216, 2010.

[49] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2014, pp. 420–425.

[50] J. Ziegler and C. Stiller, "Fast collision checking for intelligent vehicle motion planning," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2010, pp. 518–522.

[51] M. Koschi and M. Althoff, "SPOT: A tool for set-based prediction of traffic participants," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017.

[52] A. Lambert, D. Gruyer, G. S. Pierre, and A. N. Ndjeng, "Collision probability assessment for speed control," in *Proc. of the 11th International IEEE Conference on Intelligent Transportation Systems*, 2008, pp. 1043–1048.

[53] M. Althoff, O. Stursberg, and M. Buss, "Model-based probabilistic collision detection in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 2, pp. 299 – 310, 2009.

Appendix B

Supervised Theses

[129] P. Halder, “Traffic scene modeling for semantic labeling of reachable sets and driving corridor identification for autonomous vehicles,” Master Thesis, Technical University of Munich, 2019.

[130] M. Althaus, “Consideration of safe distances in online verification for motion planning of autonomous vehicles,” Master Thesis, Technical University of Munich, 2019.

[131] H. Kirchner, “Randomized generation of road networks and 3D visualization of traffic scenarios in Gazebo,” Bachelor Thesis, Technical University of Munich, 2017.

[132] F. Böck, “Combining variational-based trajectory planning with driving corridor computation for automated vehicles,” Master Thesis, Technical University of Munich, 2017.

[133] A. Weiß, “Trajectory planning for cooperative vehicles combining A* search with state sampling,” Bachelor Thesis, Technical University of Munich, 2017.

[134] A. Zhu, “Automated approximation of lane boundaries of arbitrary road networks with simple geometric shapes,” Bachelor Thesis, Technical University of Munich, 2017.

[135] L. Mittag, “Assignment of individual driving areas for cooperative vehicles considering safety and efficiency,” Master Thesis, Technical University of Munich, 2018.

[136] L. Schäfer, “Combining driving corridor computation and cooperative distributed optimal control for trajectory planning for automated vehicles,” Bachelor Thesis, Technical University of Munich, 2018.