2019 IEEE Intelligent Vehicles Symposium (IV)
Paris, France. June 9-12, 2019

Cooperative Multi-Vehicle Behavior Coordination for Autonomous Driving

Tobias Kessler[1] and Alois Knoll[2]

***Abstract*— Creating rational driving options and designing the decision process to select the best solution in a traffic situation with multiple participants present is a challenging problem. Other participants could be cooperating communication-enabled autonomous vehicles or vehicles controlled by human drivers with egoistic goals.**

This work introduces a novel approach to coordinate the behavior of multiple vehicles in generic traffic scenes. Our three-step method generates motion options neglecting vehicle interactions at first. Afterward, a mixed-integer linear optimization problem is solved to find the optimally coordinated motion patterns, followed by an online re-calibration based on the observed behaviors in reality.

We demonstrate and evaluate the applicability in an evasive maneuver requiring vehicle interaction in detail and also present an intersection scenario. We further show that cooperative behavior, as well as egoistic driver intentions, can be handled safely and analyze the properties of the proposed solution.

I. Introduction

Traffic scenarios with multiple vehicles interacting are challenging for autonomous vehicles. Even if the rough intention of another traffic participant is known, all interacting vehicles have to – implicitly or explicitly – agree on a coordinated and conflict-free motion plan. The motion has to be comfortable and safe for each vehicle and must fit all individual goals and desires. As an alternative to centralized coordination by an intelligent infrastructure, we implement these coordination capacities on the autonomous vehicle itself.

If we model a traffic scenario as a cooperative multi-agent system, the global optimal solution is a maximized throughput without deadlocks. Human drivers in contrast often do not aim for this global optimum but show egoistic behavior. This behavior can only be observed and future actions only estimated.

Several algorithms have been proposed to solve these various scenarios [1]; we discuss some in Section III. However, performance varies in different scenarios. Up to now, no method has outperformed all others. One major issue is the handling of mixed traffic scenarios with human-driven and autonomous vehicles. For a comprehensive interaction-aware behavior planning the predicted behavior of every participant in the traffic scene has to be included. In state of the art algorithms, often the strategic planning layer for behavioral decisions is separated from the motion planning layer. This separation yields the problem that high-level strategy decisions can sometimes not be realized due to different constraints.

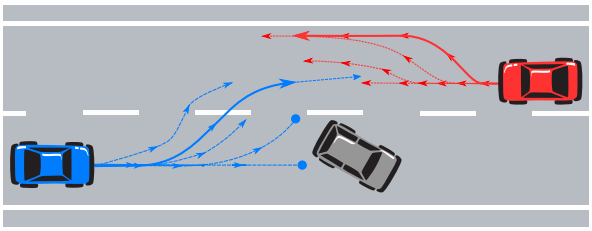


Fig. 1. Sketch of a traffic scene with multiple behavior options: The blue (leftmost) ego vehicle might be forced to stop in front of the gray static obstacle (middle) depending on the behavior of the red (rightmost) vehicle. The set of possible options is depicted as dashed lines and the optimal solution leveraging both interests as bold solid line.

This raises the need for integrated motion planning (smooth and collision-free trajectories) and strategy planning (tactic decisions and rules) layer. Also, a unified behavior coordination approach in all locations and scenarios that can deal with all possible constellations of human-driven, fully automated, communication-enabled or non-communicating vehicles has to be implemented.

In this work, we propose a novel three-step approach for this problem using an integrated motion and strategy planning layer. First, we generate a tree-like graph structure of all possible motion options for each traffic participant in the field of view of the vehicle. Each path in the tree corresponds to one possible behavior. From estimated intentions costs are assigned to each behavior. Second, we integrate all graphs into one symmetric mixed-integer linear optimization problem. Suitable constraints avoid colliding behavior options and an optimal set of behavior motion options according to previously assigned costs is computed. Third, we observe how the traffic scene evolves to update intentions estimations and tune the parameters. Also, ego cost terms can be updated to change the driving style.

Fig. 1 sketches a simple coordination problem. With a coordinated motion, both vehicles can pass the obstacle without coming to a complete stop. Still, several other reasonable driving options are possible. The work at hand puts a focus on fully cooperative scenarios but also shows results for mixed traffic scenarios.

II. Problem Definition and System Overview

In this work, we develop a strategic behavior planning component that calculates a coordinated motion for all vehicles in the current traffic scene. The algorithm runs in a decentralized manner with limited information exchange among

[1]fortiss GmbH, An-Institut Technische Universität München, Munich, Germany
[2]Robotics, Artificial Intelligence and Real-time Systems, Technische Universität München, Munich, Germany

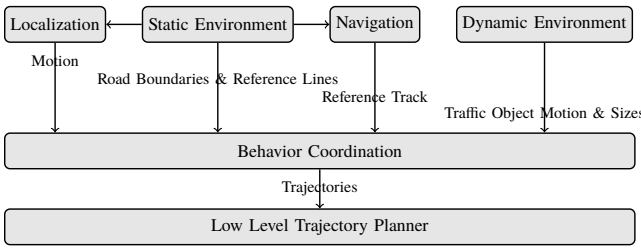


Fig. 2. Overview of the components interacting with the behavior coordination system

multiple cooperating autonomous vehicles. It also takes non-cooperating vehicles into account by predicting and estimating their future motion and strategic plans. The solutions are collision free and drivable by a (non-holonomic) vehicle. The behavior generalizes for multiple different scenarios and is not tailored to a specific traffic scenario. We will refer to this component as *Behavior Coordination*. Fig. 2 shows the main interfaces.

Four components serve as main inputs of the behavior coordination component. The first is a static environment representation containing road and lane reference lines including possible connections (e.g., at an intersection). Also, it represents the road boundaries and known static obstacles in a polygonal way. This component will be referred to as *Static Environment* model.

Second, we interfere with a *Dynamic Environment* model. It delivers the current states (pose, orientation, velocity) of all traffic participants in the scene. The source can be the output of a sensor data fusion and classification pipeline or received data from an infrastructure or car to car communication unit. Also, we assume an approximate shape and size of the traffic participant to be estimated.

The third data source is the vehicle's *Localization* component. It provides the ego vehicle states (pose, orientation, velocity) and the localization on the static map.

A *Navigation* entity provides a goal reference track within the static environment map.

By *Intention* we denote the current reference track of a vehicle plus the desired speed, acceleration, and curvature goals.

The behavior coordination outputs a future trajectory for each participant in the traffic scene. This property makes our approach particularly suited in a cooperative setup where each vehicle follows its calculated motion. For other traffic participants, the output trajectory represents the predicted motion and intention. For the ego vehicle, a low-level trajectory planner with a subsequent controller is assumed to track the computed trajectory as close as possible and react to safety-critical events.

III. Related Work

In the recent literature, several strategic planning or behavior generation algorithms can be found [1]. Distinctions are (1) the discussed traffic scenarios, (2) the usage of communication and the presence of non-communicating agents, and (3) the applied solution methods.

Rios-Torres and Malikopoulos [2] give an overview of coordination approaches for connected and automated vehicles (CAVs) in merging and intersection scenarios. Human-driven vehicles are not considered. Chen and Englund [3] also review intersection management including motion planning strategies. One scenario-independent conflict resolution strategy among CAVs is proposed by Lehmann *et al.* [4]. Each vehicle communicates its planned trajectory along with a potentially different desired trajectory. Based on these, other vehicles can adapt their plans without explicit negotiation or acknowledgments. The coordination is computed on a trajectory level in the Frenét frame without assumptions how these are computed. Wang *et al.* [5] formulate the coordination of CAVs as centralized optimization and rephrase it as consensus optimization. By decomposition, convexification, and parallelization, a specialized solver computes the solution fast even for a high number of vehicles. Düring and Pascheka [6] base the coordination of CAVs on a set of possible maneuvers represented as motion primitives with associated costs. Based on the exchange of these sets, each vehicle selects a cooperative maneuver combination. This work also takes non-communicating vehicles into account as also e.g., the work of Kurzer *et al.* [7]. Here maneuver primitives are used to compute a coordinated motion of vehicles without communication using Monte Carlo Tree Search. The interaction is modeled using a joint reward function with scaling factors for egoistic or cooperative behavior. The authors' extended work to continuous state spaces [8] resolves situations comparable to this work. The prediction of the ego vehicle if other vehicles' will cooperate influences the cooperative character of the solution.

We, in contrast, use mixed-integer linear programming (MILP) to compute optimally coordinated motions. MILP is a promising algorithm for cooperative motion planning strategy [9] on a vehicle level.Burger and Lauer [10] extend the work of Qian *et al.* [11] to cooperative, communication-enabled autonomous vehicles. A joint cost function is optimized using mixed-integer quadratic programming and avoids collision on a vehicle motion level. Receding horizon conflict resolution on an infrastructure level using intersection managers [12] can also be based on MILP. Our previous work [13] proposed a trajectory coordination system for automated parking scenarios. As in this work, MILP is used to find a suitable assignment of vehicles to a precalculated motion. In contrast, the approach presented here, we solve generic scenarios and do not assign velocity profiles to precomputed paths.

The work at hand takes a static environment map as input. From this, especially the available free-space for maneuvering and a set of possible reference tracks to follow has to be available. These can be drawn from a high-definition map or a system proposed by Kessler *et al.* [14] can be used.

We adopt the quantitative measure of two cooperative maneuvers in relation to a reference maneuver in our evaluation from the work of Düring and Pascheka [6].

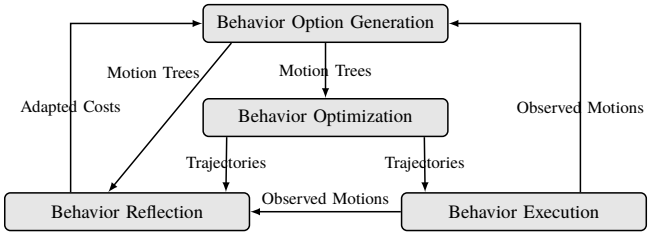


Fig. 3. Components of the behavior coordination system

IV. Behavior Coordination Approach

Our behavior coordination approach consists of four major components. Their interaction is visualized in Fig. 3 and described in detail in the following subsections.

A. Behavior Option Generation

For each traffic participant, a tree-like graph structure of possible motions over time is calculated. Starting from the position of a vehicle at the current time as root state, we generate new future states by sampling new vehicle motion options. Per timestep, we sample a number N of expansions resulting in new states. One central idea is to not recalculate the whole motion tree at every time instance but to reuse old information along the horizon. The graphs for each vehicle are independent.

One node of the graph represents a dynamic state

$$s = (x,\ y,\ \theta,\ v,\ t,\ c_s) \tag{1}$$

with the pose x, y, orientation θ and velocity v of the traffic participant. Also the time t and cost c_s of the state is stored. Each time we add a new node to the motion tree, all possible motion patterns expanding the new node are added to an expansion list for future exploration.

These expansion list entries form the edges of the graph and encode how to transfer from one state to another and can be any physically admissible motion. We rely on a fixed input acceleration a and curvature κ for the time interval of one edge defined as

$$e = (a,\ \kappa,\ c_e) \tag{2}$$

with costs c_e. Using the Euler discretization of the single track vehicle model,

$$\frac{d}{dt}\left(x,\ y,\ \theta,\ v\right) = \left(v\cos\theta,\ v\sin\theta,\ v\kappa,\ a\right) \tag{3}$$

we derive motion primitives from these acceleration-curvature pairs. The usage of further, probably longer, driving patterns is possible but has not been evaluated yet as it did not prove to be necessary for the discussed scenarios.

If two nodes result in the same dynamic state, the states are merged, and only one is kept in the graph. This process forms a tree-like graph with possible interconnections. Fig. 4 schematically shows how the tree is expanded and Algorithm 1 states the implementation of one algorithm step.

With adding a node to the (initially empty) graph by `addNode()` also a list of possible next expansions is added by

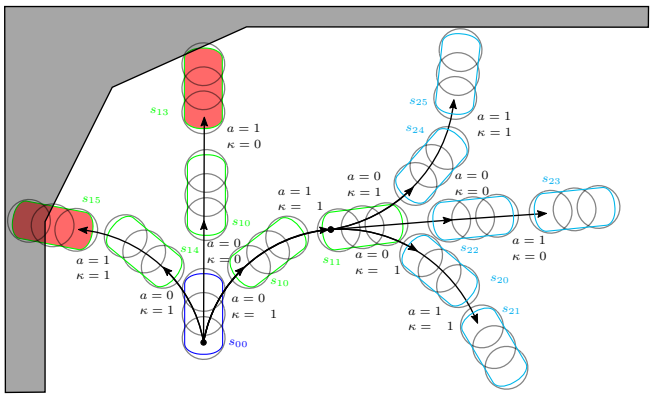


Fig. 4. Schematic state expansion starting from state s_{00} (blue) with different edges corresponding to acceleration a and curvature κ values. Starting from the initial state s_{00} (blue), the states s_{1*} (green) indicate the first expansion step, s_{2*} (light blue) the second expansion step. The vehicle shape circle approximation is visualized. The states colliding with the static obstacle (gray) s_{13} and s_{15} (solid red) are dropped from the graph.

Algorithm 1 One Step of the Behavior Option Generation

```
Parameter N                                   ▷ Number of Expansions
Input s                                       ▷ Current Vehicle State
Input env                                     ▷ Static Environment
Input t                                       ▷ Motion Tree Graph
Input e                                       ▷ Expansion List
for 1 : N do
    if not empty(t) then
        nextExpansion ← popFront(e)
        nextNode ← calculateNextState(nextExpansion)
        if not collide(nextNode, env) then
            if not stateAlreadyInGraph(nextNode, t) then
                addNode(nextNode, t)
                appendNewEdges(e)
                removeDeadEnds(e)
            else
                addConnections(t)
            end if
        end if
    else
        addNode(s, t)
        appendNewEdges(e)
    end if
end for
return t, e
```

`appendNewEdges()`. The list holds pairs of a state (parent nodes) and an edge leading to a possible new node. The function `calculateNextState()` computes a new possible node from one entry of the expansion list. Afterward, the new state is checked for collisions with the static environment by `collide()` and added to the graph if no collision is found. For the collision check, we approximate the vehicle shape with a set of circles [15]. We used three circles per vehicle. If a similar node is already part of the graph (e.g., reached from another trace in the graph), checked by

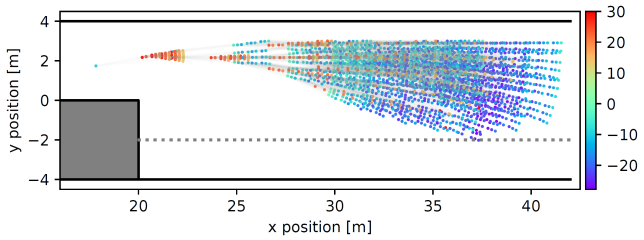


Fig. 5. Example of one full expansion step. The color coding refers to the state costs. The states' x, y positions correspond to the vehicle positions. The gray obstacle is avoided as well as the solid black road boundaries; the reference track is depicted in dashed gray. Note that the edge curvatures are omitted as well as the vehicle orientation θ.

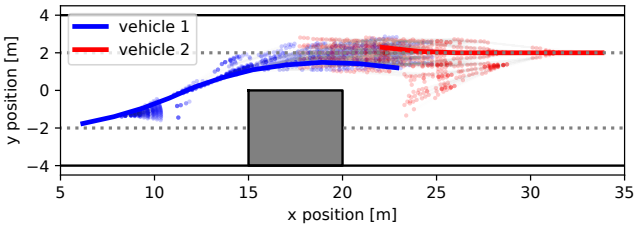


Fig. 6. Example of two (blue, red) optimized trajectories (thick lines) in the motion trees avoiding the gray obstacle and staying within the road boundaries. Note that the edge curvatures are omitted as well as the vehicle orientation θ.

`stateAlreadyInGraph()`, we do not add the node again but only add a fitting edge in the graph. Note that the time t is part of the dynamic state which shrinks the search space. Consequently, we speak of a tree-like structure as there might be more than one path through the (directed acyclic) graph leading to a node following a different series of edges. If a node does not have any children (a dead end) or yields an immanent collision (e.g. node s_{25} in Fig. 4), we remove this node from the graph and recursively all completely explored nodes that only connect to the dead end node by `removeDeadEnds()`. Fig. 5 shows an example of one motion tree.

The costs c_s of a state s calculate to

$$c_s = \xi_{ref} dist(s, \vec{s}_{ref}) + \xi_v abs(v_s - v_{ref}) + \xi_{root} dist(s, s_{root}) \tag{4}$$

with scaling factors ξ and appropriate distance functions $dist$. For each vehicle, we either know the intention, and therefore the reference track and speed or the intention is estimated. We put positive costs scaling on the deviation from the reference track $\vec{s}_{ref}$ and the reference speed v_{ref} and negative costs (reward) on the distance from the root state s_{root}. We chose the Euclidean distance of s to the nearest point of the reference track from the static environment as the distance function. As the distance from s to the root node we set the distance from the root node to the nearest point along the reference track.

The costs c_e of a edge e calculate to

$$c_e = \xi_a a + \xi_\kappa \kappa \tag{5}$$

with acceleration a, curvature κ and scaling factors ξ.

B. Behavior Optimization

The purpose of this step is to find a conflict- and collision-free set of traces within the combined motion trees (cf. Section IV-A) of each vehicle minimizing the total costs. Recall that we are calculating this motion tree individually for each vehicle in the scene. The trees are collision-free regarding the static environment but not regarding other vehicles. We formulate this search as mixed-integer linear optimization program.

We formulate the optimization problem as a flow problem in terms of graph edges. n_e denotes the number of edges, n_s the number of nodes of the graph. Decision variables are the flows f on the edges of the graph

$$f_{st}^v \in \{0,1\}^{n_e} \tag{6}$$

with the source node s and the target node t for each vehicle v. The flow conservation constraints are formulated to

$$\sum_{\text{Nodes j}} f_{ij}^v - \sum_{\text{Nodes j}} f_{ji}^v = \begin{cases} 1, & i \text{ source} \\ -1, & i \text{ sink} \\ 0, & \text{otherwise} \end{cases} \quad \begin{matrix} \forall \text{ Nodes } i \\ \forall \text{ Vehicles } v \end{matrix} \tag{7}$$

As edge costs we chose the sum of the actual edge costs and the costs of the source node. To make sure the optimizer does not get stuck at intermediate nodes, we connect all possible end nodes to a virtual sink node and force the flow formulation to start at the root node (source) and end at this sink node that is unique per vehicle. Possible end nodes are all non-fully expanded nodes or nodes with zero speed. Each fully expanded node has at least one valid successor node that represents a vehicle state further in time.

The objective function is to minimize the total costs along the combined graph traces which is equivalent to finding the minimal flow in the connected graphs. It calculates to

$$\sum_{\text{Vehicles } v} \lambda_v \sum_{\text{Nodes } i,j} f_{ij}^v (c_{ij} + c_i) \tag{8}$$

with the respective edge costs c_{ij} and the node costs c_i. We further introduce a vehicle scaling factor λ to adjust the level of cooperation for each specific vehicle. We initially set $\lambda = 1$.

We implemented two ways to check the inter-vehicle collisions. First, we externally check the collisions for each pair of vehicles by

$$\sum_{\text{Nodes k}} f_{k,i}^v + \sum_{\text{Nodes k}} f_{k,j}^w \leq 1 \tag{9}$$
$$\forall \text{ Nodes } i,j \text{ collide } \forall \text{ Vehicles } v \neq w$$

before calling the optimization program to constrain the usage of conflicting nodes. The collision calculation is again done by approximating both vehicle shapes as a series of circles and checking these for intersections. The naive implementation of looping through every node for all vehicles and checking the collisions is computationally intractable.

TABLE I

OVERVIEW OF THE CHOSEN PARAMETERS WITH DISTINCTION OF THE FOUR EVALUATED SCENARIOS

l	r	dt	ξ_v	ξ_{ref}	ξ_{root}	ξ_a	ξ_δ	v_{min}	v_{max}	Expansions	v_{ref}	a	κ
$2.67m$	$1m$	$1s$	V-B: 1 V-C: 0.75 V-D: 1 V-F: 1	V-B: 1 V-C: 1.5 V-D: 1 V-F: 1	V-B: -20 V-C: -20 V-D: -20 V-F: 0	0	0	$0\frac{m}{s}$	$10\frac{m}{s}$	2000 per step	$4\frac{m}{s}$	$\begin{pmatrix} 0.5 \\ 0.25 \\ 0 \\ 0.25 \\ 0.5 \end{pmatrix} \frac{m}{s^2}$	$\begin{pmatrix} 0.18 \\ 0.09 \\ 0 \\ 0.09 \\ 0.18 \end{pmatrix} \frac{1}{m}$

Therefore, we first perform an approximate collision check based on the shape and position of the vehicles to mark definitely non-colliding states. To further boost the performance we iterate through the graphs with increasing time and only compare nodes with the same timestamp. We also stop to iterate deeper in the graph if a collision was found.

As a second method we formulate the circle approximation collision checker as constraints internally within the optimization problem as

$$\begin{aligned} &(x_i + l_v \cos(\theta_i) - x_j - l_w \cos(\theta_j))^2 + \\ &(y_i + l_v \sin(\theta_i) - y_j - l_w \sin(\theta_j))^2 \geq (r_v + r_w)^2 \\ &- M(1 - \sum_{\text{Nodes k}} f^v_{i,k}) - M(1 - \sum_{\text{Nodes k}} f^w_{j,k}) \qquad (10) \\ &\forall \text{ Nodes } i, j \ \forall \text{ Vehicles } v \neq w \end{aligned}$$

for each two nodes i, j with positions x, y and orientation θ for each pair of vehicles v, w. A vector l denotes the positions of the vehicle approximation circle centers, whereas 0 indicates a circle around the rear axle center point. A reasonable upper bound is the vehicle's wheelbase. r denotes the radius of these circles. If, for example, each vehicle is approximated by three circles Eq. (10) resolves to nine constraints. We use the well-known Big-M method with a sufficiently large number M. Our experiments showed a better overall runtime performance using external collision constraints.

The optimization problem is consequently formulated as

$$\begin{aligned} &\text{minimize } (8) \\ &\text{subject to } (7), (9). \end{aligned}$$

The result of the optimization is an optimal edge sequence for each vehicle referring to a trajectory cf. Fig. 6.

C. Behavior Reflection

In the behavior reflection step the cost scaling factors ξ and λ are tuned to fit the individual vehicle intents or observations. Also, the intention of each non-communicating vehicle is re-estimated. Hence, the used reference track and reference speed can change.

In the case of a fully cooperative scenario, cost terms and references are known for each vehicle. As each vehicle follows the optimized trajectory, there is only a need for adoption if the intention changes.

For a non-cooperating vehicle, we observe the deviation of the observed vehicle motion and the optimized vehicle motion. Based on this, the intention is estimated in terms of the reference track and the cost factors. In this work, we only adopt the scaling factors λ online to account for cooperative or egoistic behavior. λ is increased by a constant factor per step if a vehicle has moved further than the optimal motion and decreased otherwise. A formalized and more elaborative scaling is currently under development.

D. Behavior Execution

In the behavior execution step, the ego-vehicle performs its optimal action, and the other vehicle motions are observed or received by vehicle to vehicle communication. This step runs in parallel to the behavior reflection. As our computed trajectories are discretized with a rather high time discretization, we propose to smooth the trajectories with a suitable low-level trajectory planner (cf. e.g., [16]) keeping as close to the original trajectory as possible. Also, vehicles leaving the ego vehicle's field of view are dropped and new vehicles added if present. In this work, we assume an optimal trajectory tracking.

V. EVALUATION

We demonstrate the capabilities of the algorithm in a simple scenario with two vehicles and a roadblock as illustrated in Fig. 1. The left (blue) ego vehicle 1 has to decide whether to pass the roadblock before or after an oncoming (red) vehicle 2 or if the oncoming vehicle gives way. With different settings, a reasonable behavior is always achieved. In the following, we discuss three different examples.

A. Implementation Remarks

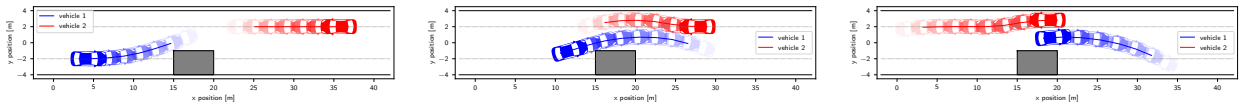
We implemented our approach in C++ without a focus on performance. The graph structures are represented using the boost graph library[1] and the geometric computations use the boost geometry library[2]. The optimization problem is formulated and solved using the commercial solver CPLEX[3]. The evaluations are carried out in a simulation framework with simplified vehicle dynamics. Cooperative vehicles are moved ideally along the optimized trajectories and non-cooperative vehicles along a predefined trajectory. A more sophisticated behavior model for non-cooperative vehicles is planned to be included in future work.

The parameters are summarized in Table I. We trigger the behavior coordination with a fixed time discretization dt. For the vehicle model integration and the collision

[1] https://www.boost.org/doc/libs/1_68_0/libs/graph/doc/index.html

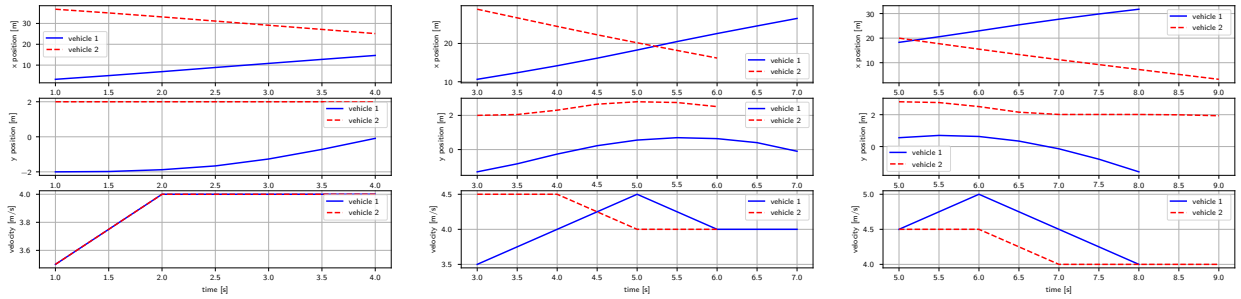
[2] https://www.boost.org/doc/libs/1_68_0/libs/geometry/doc/html/index.html

[3] https://www.ibm.com/analytics/cplex-optimizer



(a) $t = 1s$ (b) $t = 3s$ (c) $t = 5s$

Fig. 7. Evolution of a two vehicle scenario with cooperative behavior on an Cartesian plot. Dark to light colors depict increasing time. The trajectories start at the current vehicle positions. The reference lines are dashed gray. Fig. 7(b) illustrates the evasive maneuver.



(a) $t = 1s$ (b) $t = 3s$ (c) $t = 5s$

Fig. 8. Quantitative evaluation of a two vehicle scenario with cooperative behavior depicted in Fig. 7. The plots show the planned trajectory x, y positions and velocities over time.

checks, we chose a suitable subsampling corresponding to the maximum vehicle velocities. All parameters can be set individually per vehicle. In the shown evaluations, we choose equal parameters for simplicity. The reference tracks and velocities are predefined for each vehicle. With different cost terms, different behaviors are realized as shown in the subsequent sections. The cost scaling parameters ξ were chosen to demonstrate the capabilities of the algorithm. An elaborated strategy on how to chose these parameters including a sensitivity analysis is planned as future work.

B. Fully Cooperative Solution – Evasive Maneuver

Fig. 7 shows the evaluation of the traffic scene over time at selected time instances including the currently planned trajectories and occupied spaces. In this fully cooperative scenario, the optimal solution is for vehicle 2 to perform an evasive maneuver so both vehicles can pass the obstacle at the same time. Note, that with constant control inputs per timestep the motion is smooth but with linear velocity profiles as shown in Fig. 8. The lengths of the chosen trajectories do not necessarily have to be equal as the behavior graphs for each vehicle can have different time depth. Both vehicles also deviate from the reference speed to be able to pass each other at an optimal space-time location.

C. Fully Cooperative Solution – Speed Change

By adopting the parameters, different cooperative solutions can be enforced. With a lower focus on the reference speed tracking, a higher focus on keeping the reference track, and reduced initial speed of vehicle 2 of $2m/s$, it decelerates to let vehicle 1 one pass the obstacle first. The evolution of this scenario is depicted in Fig. 9. In contrast to the uncooperative case (cf. Section V-D), neither vehicle has to come to a complete stop. As both vehicles have agreed on the cooperative motion plan, vehicle 1 can start the overtaking maneuver at $t = 2s$. For this and the following scenarios, we omit the plots over time.

D. Uncooperative Behavior

In case the oncoming vehicle is not cooperating and keeps constant speed of $4m/s$, vehicle 1 has to stop and pass the obstacle after the oncoming vehicle 2. The evolution of this scenario is depicted in Fig. 10. Beginning from timestep $t = 2$ vehicle 1 decelerates and awaits if vehicle 2 gives way. As vehicle 2 does not show cooperative behavior, vehicle 1 brakes and continues at low speed until vehicle 2 has passed the obstacle at $t = 9s$. We observe that also in the presence of a non-cooperating vehicle our approach yields a safe solution.

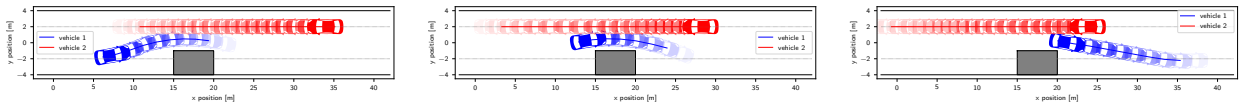
E. Discussion

The time to pass the obstacle can serve as a measure of cooperation. Table II states the time it takes for both to pass the obstacle and to travel with the targeted speed on the respective reference line again in the three discussed scenarios. As a baseline, we simulate the scenario for each vehicle individually. By cooperation, clearly, a balanced behavior is achieved. Note that in this qualitative evaluation the speed settings in the scenarios differed slightly which also influences the travel times.

TABLE II

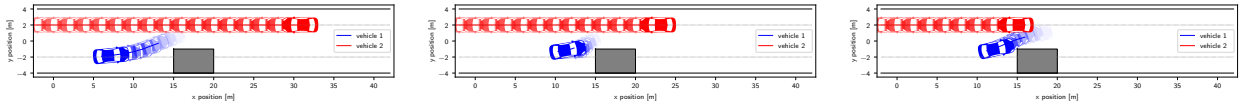
TIME NEEDED FOR BOTH VEHICLES TO FINISH THE SCENARIO

	V-B	V-C	V-D	Vehicle 1 only	Vehicle 2 only
Vehicle 1	$12s$	$10s$	$19s$	$10s$	-
Vehicle 2	$7s$	$10s$	$7s$	-	$7s$



(a) $t = 2s$ (b) $t = 4s$ (c) $t = 6s$

Fig. 9. Evolution of a two vehicle scenario with cooperative behavior on an Cartesian plot. Dark to light colors depict increasing time. Vehicle 2 decelerates to let vehicle 1 pass the obstacle first.



(a) $t = 2s$ (b) $t = 4s$ (c) $t = 6s$

Fig. 10. Evolution of a two vehicle scenario with the oncoming vehicle 2 keeping constant velocity on an Cartesian plot. Dark to light colors depict increasing time. As the behavior of vehicle 2 is unclear a feasible solution is chosen if vehicle 2 accelerates or not. The evasive maneuver is started after vehicle 2 has passed the obstacle.

To quantify how cooperative a maneuver has been performed, we evaluate the normalized contribution of both vehicles to the total costs [6]. The lower the absolute cost values are for a vehicle, the more desirable is the solution for the respected vehicle. Fig. 11 shows the individual cost contributions of vehicle 1 divided by the cost contributions of vehicle 2 per step. Cooperative solutions should be close to the ideal value of 1.0 as no vehicle is favored according to the cost function. Deviations from 1.0 indicate favor towards one vehicle and values lower than zero indicate a strong favor towards one vehicle. Here states are rewarded for one vehicle that result in costly states for the other one. In our example, for the uncooperative case (V-D) the divided costs drop below zero at the time instance vehicle 1 has to decelerate harshly to let vehicle 2 pass. For the cooperative solution (V-B), the absolute cost values per vehicle are close, and parameter adaptions can leverage the (changing) slight favor towards one vehicle. This example illustrates how we distinguish between cooperative and uncooperative and that we can generate a safe behavior in either setting.

With the current implementation, we do not always meet real-time constraints. Even for a high number of expanded nodes, the evaluation time of the behavior generation step is neglectable. The optimization problem solution time mainly scales with the number of collision constraints. This effect makes scenes with decoupled participants fast and scenes with multiple densely interacting participants slower to solve. A quantitative evaluation of how the computation time grows with an increasing number of vehicles is planned in the future.

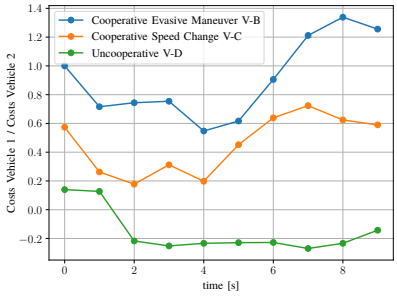


Fig. 11. Relative contributions of both vehicles to the total costs in the different scenarios. 1.0 is an ideal symmetric cost contribution of each vehicle, greater values favor vehicle 1, lower values vehicle 2. Values smaller than zero in our discussed scenario strongly favor vehicle 2.

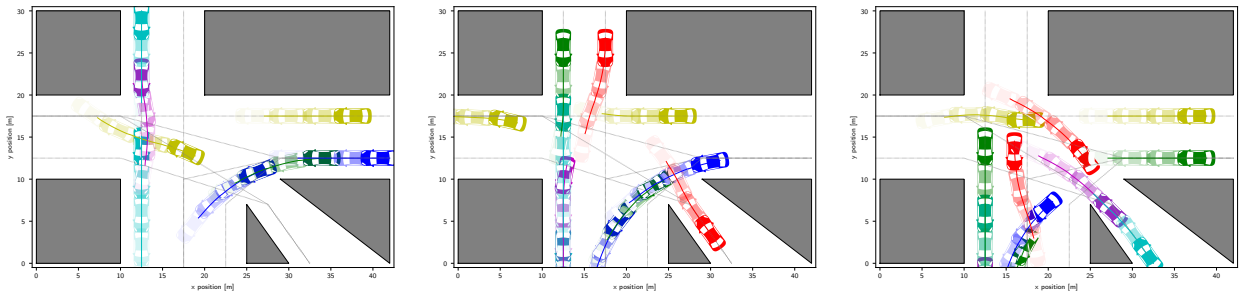
F. A Lookout on an Intersection Scenario

Without a comprehensive discussion, we show how the behavior coordination performs in an intersection scenario assuming a fully cooperative setting, cf. Fig. 12. The simulation demonstrates the universal applicability of our algorithm in generic scenarios. We simulate in total 12 vehicles simultaneously approaching from different directions at an artificial 5-way intersection. Ignoring classical right-of-way rules, the algorithm generates a solution such that each vehicle tracks its reference speed as close a possible and reaches its desired destination as fast as possible without collisions. Multiple collisions occur if all vehicles keep a constant reference speed on their reference tracks.

VI. Conclusion and Future Work

In this work, we introduced a novel behavior planning and coordination method to compute a cooperative vehicle motion strategy. It also incorporates that the behavior of other vehicles can be uncooperative. First, our approach expands individual behavior options that are quantified and stored in a tree-like structure for each vehicle. Afterward, a linear mixed-integer based optimization strategy selects the best cooperative and collision-free set of behaviors within the explored options. The vehicle intentions are adopted online to fit the actual behaviors.

The approach applies to generic traffic scenarios. It solves fully cooperative and communication-enabled scenarios as well as mixed traffic scenarios with human-driven vehicles potentially acting egoistically.



(a) $t = 5$ (b) $t = 8$ (c) $t = 11$

Fig. 12. Evolution of an intersection scenario with fully cooperative behaviors on an Cartesian plot. All vehicles move from north or east lanes to south or west lanes. Dark to light colors depict increasing time. The trajectories start at the current vehicle positions. The reference lines are dashed gray. We can e.g. observe that both red vehicles deviate from the straight reference lines to let other vehicles pass.

Our evaluations in an evasive maneuver with different behaviors, levels of cooperation and cost terms demonstrates the capabilities of the approach. The sketched intersection scenario shows the universal applicability. Already conducted experiments in further scenarios show promising results. We plan to include these in future work along with a comparison to other approaches.

As the next steps, we plan to further emphasize the interaction-awareness to other traffic participants by improving the simple implementation of the behavior reflection step by evaluating game theoretic or learning-based approaches. To bring the method to our institute's autonomous driving prototype vehicle [17], we further aim for smoother motion profiles and a more efficient implementation using e.g. parallelization or a tailored optimization solver.

References

[1] 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.

[2] 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.

[3] L. Chen and C. Englund, "Cooperative Intersection Management: A Survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 570–586, 2016.

[4] B. Lehmann, H.-J. Günther, and L. Wolf, "A Generic Approach towards Maneuver Coordination for Automated Vehicles," in *21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 3333–3339.

[5] Z. Wang, Y. Zheng, S. E. Li, K. You, and K. Li, "Parallel Optimal Control for Cooperative Automation of Large-scale Connected Vehicles via ADMM," in *21st IEEE International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2018, pp. 1633–1639.

[6] M. Düring and P. Pascheka, "Cooperative decentralized decision making for conflict resolution among autonomous agents," in *IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA)*, 2014, pp. 154–161.

[7] K. Kurzer, C. Zhou, and J. M. Zöllner, "Decentralized Cooperative Planning for Automated Vehicles with Hierarchical Monte Carlo Tree Search," in *IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 529–536.

[8] K. Kurzer, F. Engelhorn, and J. M. Zöllner, "Decentralized Cooperative Planning for Automated Vehicles with Continous Monte Carlo Tree Search," in *21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 452–459.

[9] C. Frese and J. Beyerer, "A comparison of motion planning algorithms for cooperative collision avoidance of multiple cognitive automobiles," in *IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 1156–1162.

[10] C. Burger and M. Lauer, "Cooperative Multiple Vehicle Trajectory Planning using MIQP," in *21st IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 602–607.

[11] 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 *19th IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 205–210.

[12] M. W. Levin and D. Rey, "Conflict-point formulation of intersection control for autonomous vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 85, no. 9, pp. 528–547, 2017.

[13] T. Kessler and A. Knoll, "Multi Vehicle Trajectory Coordination for Automated Parking," in *IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 661–666.

[14] T. Kessler, P. Minnerup, K. Esterle, *et al.*, "Roadgraph Generation and Free-Space Estimation in Unknown Structured Environments for Autonomous Vehicle Motion Planning," in *21st IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2831–2838.

[15] D. Lenz, T. Kessler, and A. Knoll, "Stochastic Model Predictive Controller with Chance Constraints for Comfortable and Safe Driving Behavior of Autonomous Vehicles," in *IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 292–297.

[16] 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.

[17] T. Kessler, J. Bernhard, M. Buechel, *et al.*, "Bridging the Gap between Open Source Software and Vehicle Hardware for Autonomous Driving," in *IEEE Intelligent Vehicles Symposium (IV)*, 2019.