

INFORMATION AND COMMUNICATION TECHNOLOGIES
COLLABORATIVE PROJECT

“V-Charge”

Autonomous Valet Parking and Charging for e-Mobility

Grant Agreement Number 269916



V-CHARGE

Deliverable Number	4.2
Deliverable Name	Motion Planning and Vehicle Control
WP Number	4
Date of Preparation	July 2, 2013
Date of Submission	June 30, 2013
Editor/Beneficiary	U. Schwesinger, M. Rufli/ETH Zürich

Contents

1	Introduction	2
2	Navigation Framework	2
3	Navigation Modules Progress Overview	3
3.1	Global Task Planner (VW)	4
3.2	Vehicle Model Identification (ETHZ-ASL)	5
3.3	Trajectory Follower (ETHZ-ASL)	5
3.4	Local Trajectory Planning Task Processor (ETHZ-ASL)	8
3.5	Interactive Motion Prediction (ETHZ-ASL)	10
3.6	Parking Spot Task Processor (RBG)	12
4	Auxiliary Modules	14
4.1	Simulation Environment V-Rep (ETHZ-ASL)	14
5	Conclusion	14
A	Appendices	15

1 Introduction

The V-Charge project aims to develop automated automotive navigation functionality within a restricted hybrid structured-unstructured parking lot area. Automated cars will approach, charge and return from parking zones while safely avoiding other cars and traffic participants. One of the novel aspects and key characteristics of the project is that localization, mapping and obstacle detection are solely handled by vision—supported by sensors readily available in production cars, including wheel-encoders, IMUs and ultrasound. The lack of more accurate laser-scanners and (differential) GPS poses challenges for localization, object detection, mapping, and, consequently, motion planning. This last aspect forms the research basis of work package 4. It deals with the developments of algorithms for safe navigation in semi-structured and dynamic environments and comprises the following tasks (as defined in Annex 1):

- Task 4.1: Specification of motion planning and vehicle control concepts (ETHZ (Lead), VW, RBG)
- Task 4.2: Global path planning (ETHZ (Lead), VW, RBG)
- Task 4.3: Control Interface and trajectory following (VW (Lead), ETHZ)
- Task 4.4: Partial motion planning and collision avoidance (ETHZ (Lead), VW, RBG)

Task 4.1 has been thoroughly addressed by Deliverable 4.1. For convenience, [Section 2](#) again illustrates an overview of the navigation architecture developed therein. The remainder of this deliverable then details the progress on the various motion planning task processors as well as the control functionality that form said navigation architecture. These parts as a whole are aligned with Tasks 4.2–4.4.

2 Navigation Framework

This section briefly reviews the V-Charge navigation framework developed in D4.1. [Figure 1](#) displays the conceptual layout—both for WP 4 and adjacent modules. It shares key aspects with the architectures employed by the top Darpa Urban Challenge finishers, including a behavioral layer that handles priority between a set of specialized planning instances (called *task processors*). The selected hierarchical architecture encourages low-complexity modules. In addition it enables the definition of clean interfaces in between. Finally, it improves safety of operation, in that individual module failure does not jeopardize the overall system.

Due to the special setup of the V-Charge project centered around a parking lot, we distinguish modules operating on the *central server side* (light blue) from those on each of the *automated vehicles* (dark blue). Information between modules flows in the form of broadcasts or point-to-point connections using the DDS middle-ware. The *parking manager* is located on server side. The responsibility of this module is to process requests of incoming and outgoing vehicles. It will assign free parking spots and charging areas by considering charging needs and expected parking time. It is furthermore responsible for ordering vehicles back to the pick-up zone upon request. The parking manager communicates the resulting mission goals (one or more node IDs of the road graph) to the global task planner on the target vehicle. The *global task planner* operates on a regularly updated road graph (obtained from the road graph server). It is responsible for topological route planning on the road-graph, which is performed using exhaustive graph search. Once a solution to the mission is available in the form of a sequence of topological edge transitions, the global task planner divides it into a feasible sequence of tasks that will be executed by specialized task processors (i.e., behaviors).

The *mission executive* located on vehicle side is responsible for task assignment to the individual task processors, management of task processor exceptions and the overall correct execution of events. It contains

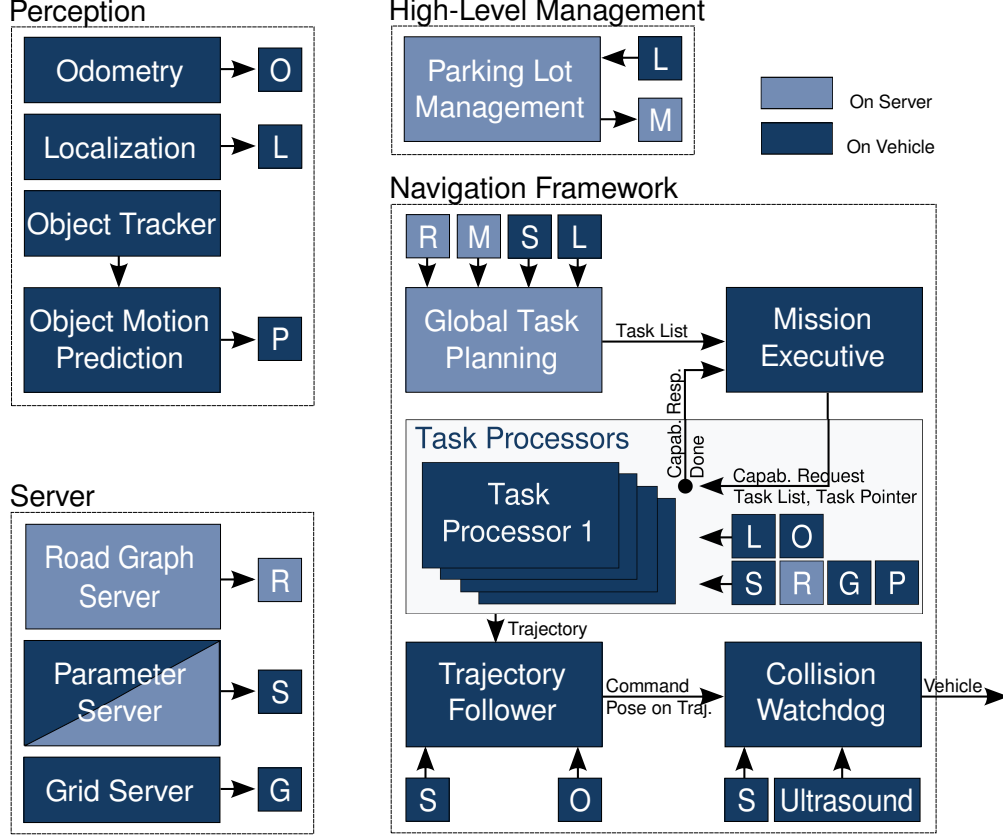


Figure 1: The V-Charge navigation architecture including adjacent modules to highlight the relevant interfaces. Modules colored in light blue operate on the central server side, those colored in dark blue on the automated cars.

a task scheduler in the form of a state machine, which advances a *task pointer* once a given task has been completed by the responsible task processor. *Task processors* are modules, which are responsible for one or more task types.

The remainder of this deliverable details progress on the low-level *trajectory following module*, the *local trajectory planning* and *parking* task processors, as well as the *global task planner*.

3 Navigation Modules Progress Overview

With the submission of D 4.1 the navigation architecture specification has been finalized. In the meantime WP 4 has progressed substantially along tasks 4.2–4.4. Particular attention has been laid on the preparation for handling of dynamicity in the environment. The extension of the low-level trajectory follower and the design of the model-based local trajectory planning task processor represent important building blocks towards realizing safe navigation among other cars and pedestrians.

3.1 Global Task Planner (VW)

The global task planner operates on two levels. Firstly, a global path planner returns a kinematically feasible path from the current vehicle pose to a (set of) feasible goal pose(s) by taking into account the static environment that has been surveyed and processed into a road graph off-line. Dynamic objects are neglected in the global planning stage, and roads are assumed to be inherently drivable. Secondly, the solution path is split into a sequence of tasks (i.e., behaviors), which are then passed on to the mission executive in the form of a task list. Elements of this list are subsequently processed by the relevant task processors described in [Section 3.4](#) and [Section 3.6](#).

Computation of the global reference path is based on the RoadGraph which constitutes a graph structure representing the traffic infrastructure on the parking lot. For instance, it carries information about road lanes, poses of the parking spots, peculiarities of intersections, and much more. At the current stage of the project, the RoadGraph is being constructed manually by using the Java Open Street Map (JOSM) editor. JOSM facilitates to import aerial images. Hence, map images of the parking lot are imported as aerial images by attaching world files to the map images. Then roads and parking spots may be accurately hand-drawn since the map is visible in the background of the editor.

The global planning process is subdivided into two consecutive steps. The first step is a simple topological search of the RoadGraph yielding a set of edges guiding from the start to the goal. The RoadGraph search is performed employing the standard A* algorithm. The edges of the RoadGraph are typically not continuously differentiable and thus not drivable. Therefore, as a second step, the edges are sampled in discrete intervals. The resulting point set is smoothed using an efficient implementation of the conjugate gradient descent method. An example of this process is depicted in [Figure 2](#). The RoadGraph is represented by the yellow edges while the resulting global reference path is highlighted by the green curve.

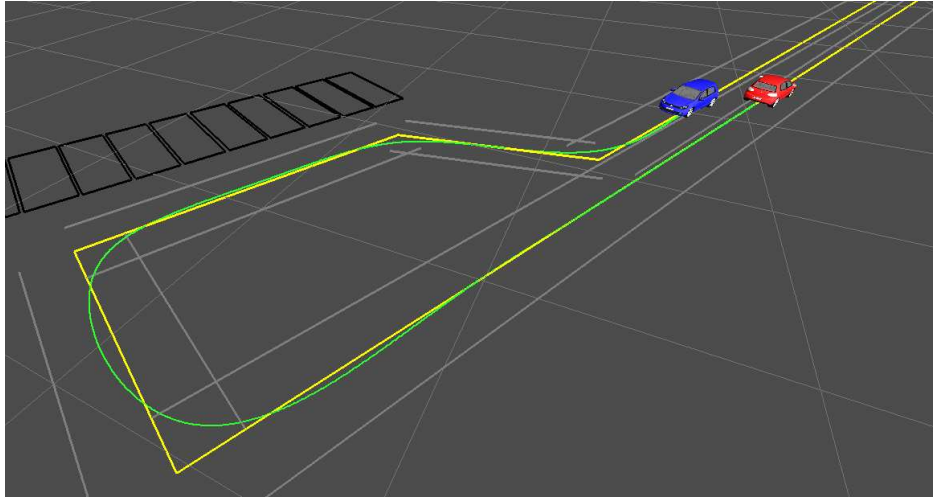


Figure 2: Example of the manually constructed RoadGraph (yellow), and the smoothed global path (green).

In order to realize navigation along the reference path a sequence of single tasks is computed organizing all essential steps:

- Vehicle initialization: Switch on the engine, release the parking brake, etc.
- Navigation along the reference path: Vehicle follows the roads and reacts to dynamic obstacles and unforeseen events, respectively.

- Passing intersections: Intersections are treated separately since special traffic rules might apply.
- Parking: As soon as the vehicle has reached the final parking spot or charging station, a parking routine is triggered.
- Vehicle deinitialization: Switch off the engine, engage the parking brake, etc.

A new set of tasks is computed again if the vehicle is ordered back to the pick up zone. In this case, the tasks appear in a reversed order since the vehicle needs to pull out of the parking spot at the beginning.

3.2 Vehicle Model Identification (ETHZ-ASL)

The V-Charge planning framework is model-based and thus makes a model-identification procedure necessary. The results from the system identification step facilitate the tuning procedure of the longitudinal control scheme within the trajectory controller as described in [Section 3.3](#). Furthermore, accuracy in this step improves the quality of the motion planner’s feed forward commands—and hence increases operational safety.

Input to the V-Charge system is via desired (i.e., target) longitudinal acceleration and steering angle, resulting in the following a priori unknown transfer functions: $f_a : u_a \rightarrow a$ and $f_\phi : u_\phi \rightarrow \phi$. Based on the results of a Diploma Thesis performed at Volkswagen, the transfer function from target acceleration to measured acceleration has been approximated by a second-order system (see [Figure 3](#))

$$G_a(s) = \frac{1.0}{0.4249s^2 + 1.352s + 1.0}. \quad (1)$$

The steering angle transfer function, by contrast was selected as a first-order system

$$G_\phi(s) = \frac{1.0}{0.2s + 1.0} \quad (2)$$

by inspection.

3.3 Trajectory Follower (ETHZ-ASL)

Planning in dynamic environments mandates controller functionality that allows for the precise tracking of time-dependent reference trajectories. This observation has been at the heart of the decision to extend the path controller presented in D 4.1 to handle timing information in addition to path shape. At the same time, it was decided that the controller should remain backward-compatible and continue to provide a convenient interface for modules that do not or cannot provide timing information along with the reference path to be followed.

The revised control strategy, described below, is separated into longitudinal and lateral control schemes. They operate independently of each other and thereby reduce control complexity substantially. The separation is justified by the lane-like structure within parking lots.

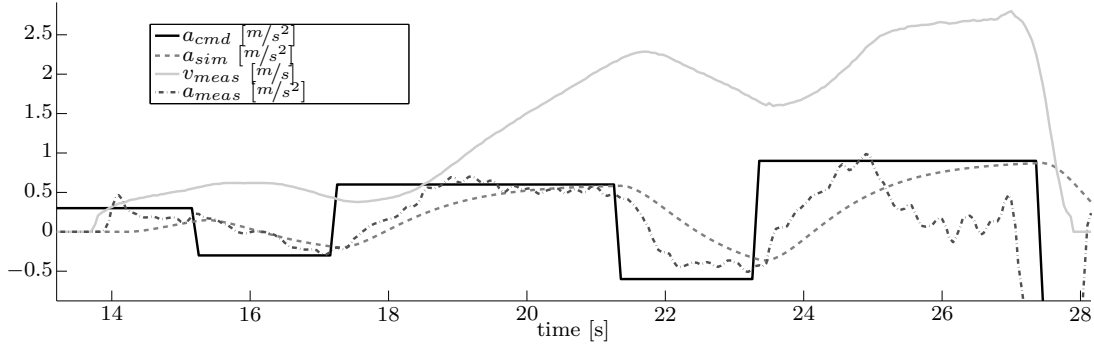


Figure 3: Simulated (a_{sim}) and measured (a_{meas}) longitudinal acceleration response to step inputs (a_{cmd}) with varying amplitude. Note that around $t=25$ s, the longitudinal vehicle speed approaches the maximum permitted vehicle speed 10 km h^{-1} and a low-level stabilizer kicks in. At $t=27$ s an emergency brake is triggered due to the violation of the speed limit.

Lateral Control Lateral control uses a nonlinear state-feedback rule to compute a steering angle correction term based on [Samson and Ait-Abderrahim, 1991]. This method requires a reference state on the reference signal that is either identified by an orthogonal projection onto the reference path (in the path case) or via a look-up by time-stamp (trajectory case). The error state computation is depicted in Figure 4. Additionally, a feed-forward command for the steering angle from the trajectory message is directly added to the control output. This feed-forward term obtained from the planner is crucial for tracking performance and eliminates the need for a look-ahead in the trajectory.

Longitudinal Control The longitudinal part is based on a cascaded acceleration/velocity/(position) control loop. and can be activated through the configuration file but its use is not intended and therefore position control is not included in the drawing. In this setup an inner PI controller regulates the acceleration error to zero resulting in an improved closed-loop system response. Note that acceleration cannot be measured directly due to a lack of appropriate sensors, but needs to be computed from odometric velocity information, which inadvertently leads to delays. An outer velocity control loop provides the inner loop's set-point. Together with the second-order engine response model and the tuned PI controller parameters, a third-order closed-loop transfer function is obtained:

$$G(s) = \frac{3.328s + 2.681}{s^3 + 3.1817s^2 + 5.6813s + 2.681} \quad (3)$$

Figure 6 depicts a sample open and closed loop acceleration response. $G(s)$ is directly used inside the local motion planner in order to obtain high-quality future motion predictions.

Operation We distinguish two modes of operation: when the local motion planner is active it supplies both velocity and acceleration feed-forward commands. Internally it employs a model of the acceleration controlled car to forward simulate its behavior to piece-wise continuous reference acceleration inputs. In this mode, the velocity feedback loop is rendered ineffective due to the planner's re-planning functionality. All other task processors supply a reference velocity signal but no model-based feed-forward acceleration commands. In these situations the velocity feedback loop takes over all responsibility.

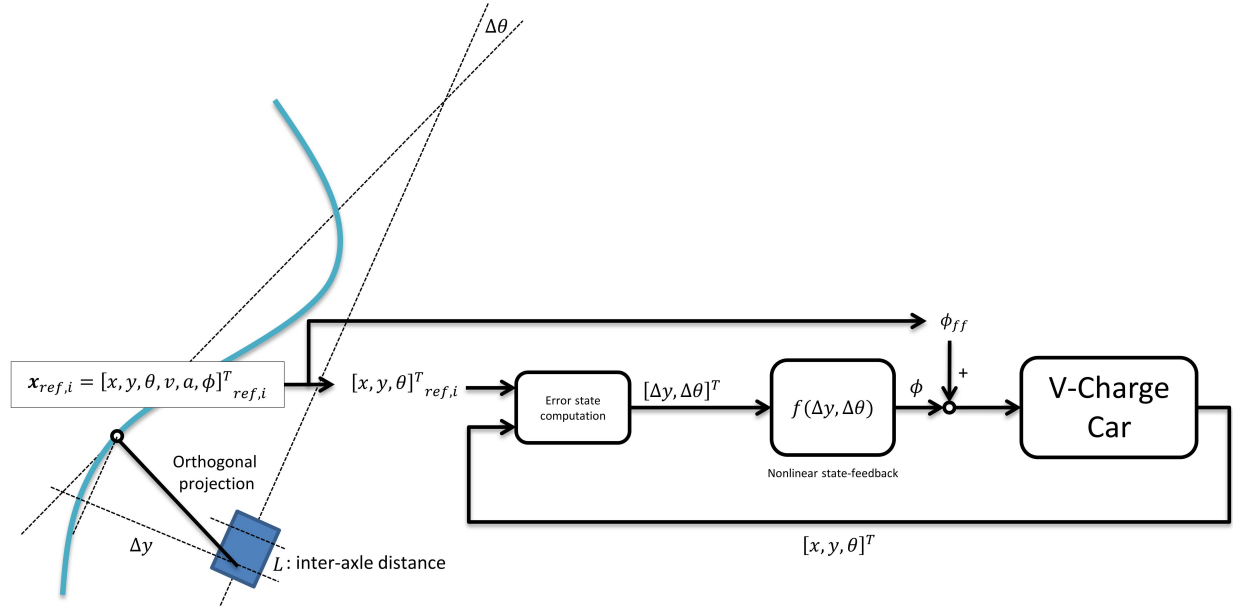


Figure 4: Lateral control scheme: Via an orthogonal projection onto the reference path (or look-up by timestamp) a single reference state is determined. The lateral offset Δy and heading offset $\Delta \theta$ are used for nonlinear state feedback control. Applying additional direct feed-forward commands Φ_{ff} eliminates the necessity for a look-ahead.

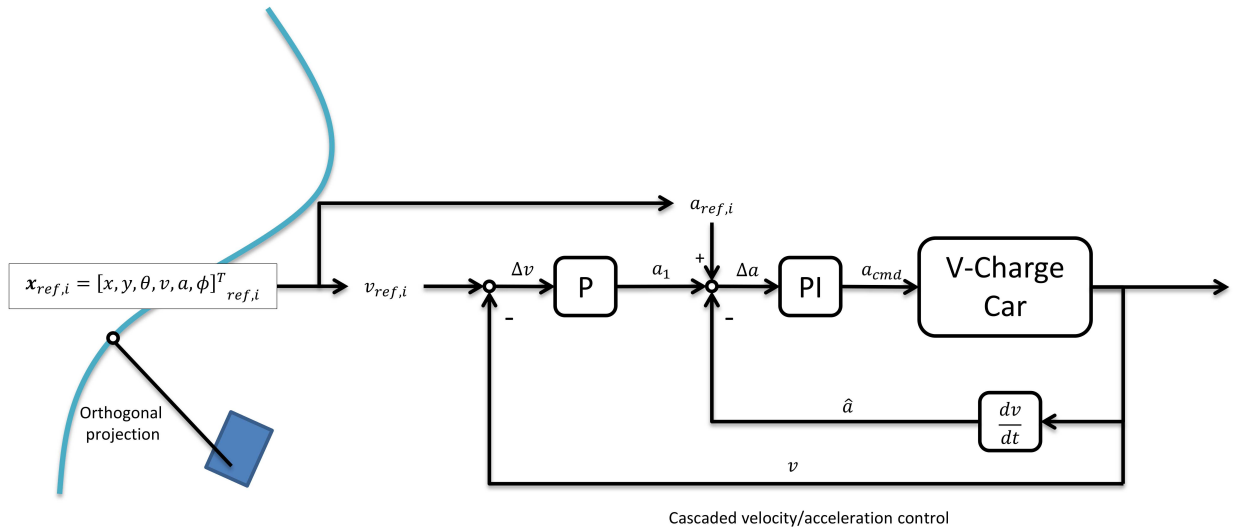


Figure 5: Longitudinal control scheme for the V-Charge platform. The longitudinal control is based on cascaded PI/P-controllers. The innermost loop controls the vehicle's acceleration. In case of mismatches between the model employed in the local motion planner and the physical controlled platform, the acceleration control loop is able to accumulate and correct the resulting error.

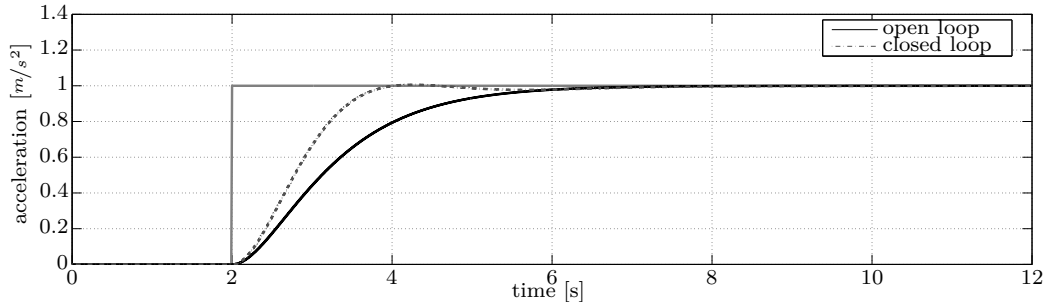


Figure 6: Simulated closed-loop system response for the longitudinal acceleration control. The resulting closed-loop system is of third order.

3.4 Local Trajectory Planning Task Processor (ETHZ-ASL)

The local motion planning and collision avoidance module’s task is to handle unforeseen changes to and unmapped objects in the environment *on-line*. To this end, it requires functionality for automatically and safely avoiding newly appearing objects obstructing parts of the reference path obtained from the high-level road-graph planning module. Owing to the camera-heavy setup within the V-Charge project, the resulting motion should at the same time be constrained to areas where a significant overlap between the cameras’ current field of view (FOV) and the ones attained during the mapping phase can be guaranteed.

Our proposed sampling-based motion planning framework enforces path alignment—and therefore generally a large overlap in the FOV—by shaping a tree of system-compliant motions along the road graph solution obtained from the global planner. System-compliance is attained via a user-supplied system model and control law (see [Section 3.2](#)). The system model is (numerically) forward simulated towards samples drawn from a state-manifold aligned with the road graph. Within the framework, the state-manifold is defined as the heading and curvature aligned subspace of the robot’s state-space. The control law of [Section 3.3](#) then regulates the simulated system towards a sampled version of this manifold. Within the forward simulation of the system model, any kind of system constraints can be enforced, making the approach appealing to input and state-constrained (e.g., non-holonomic) systems. Below we provide an overview on the various elements of the local planning module. Please consult [\[Schwesinger et al., 2013\]](#), appended to this deliverable, for a more detailed exposition. A simulated run including both static and dynamic objects can be viewed in [Figure 7](#).

System Model The highly configurable planning framework is able to incorporate arbitrary system models (suitable for forward simulation). The V-Charge system is modeled as a kino-dynamic bicycle in conjunction with black-box transfer functions for acceleration and steering angle response. The model identification is described in [Section 3.2](#).

Sampling Scheme The sampling scheme of the state-manifold used within the framework is also partially configurable. Our implementation suitable for road graph following generates candidate evasive trajectories by sampling lateral offsets to said graph. To allow for longitudinal speed adaptations while driving, several target speeds of the robot are created by sampling target speed offsets from the supplied reference speed.

Collision Detection Collision detection with *static* objects is performed on a two-dimensional binary occupancy grid. *Dynamic* objects identified by the detection and classification module of WP 3 are removed

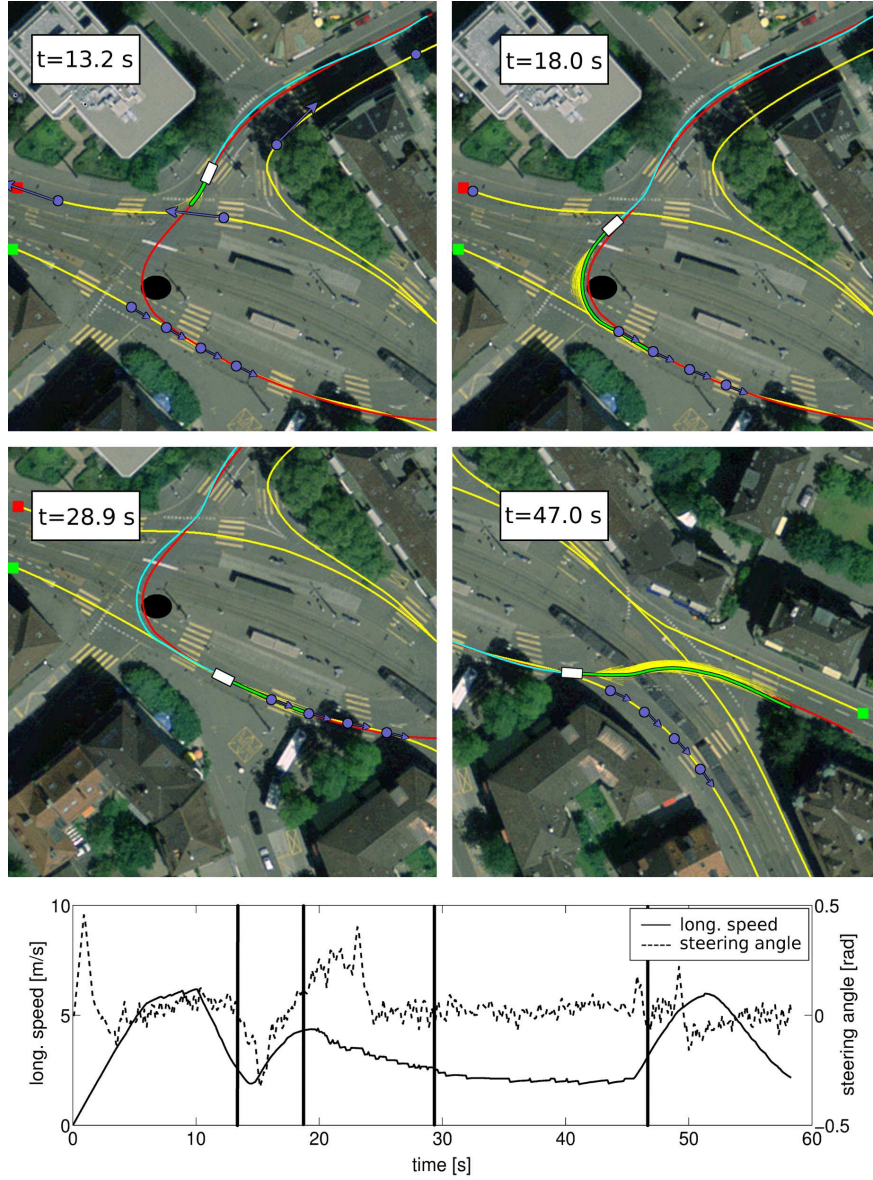


Figure 7: Local trajectory planner. *Top*: Snapshots of the simulated run. The vehicle (odometry solution in cyan) successfully avoids dynamic obstacles (blue circles) as well as static (black) ones while following the reference path (red line). From a set of explored trajectories (yellow lines), the best partial motion plan (green line) is chosen. *Bottom*: The graph shows longitudinal speed as well as steering angle of the ego vehicle. Vertical lines refer to the timestamps of the snapshots.

from the occupancy grid and are currently accounted for via collision checking in state-time space. Note that our sampling-based approach enables handling of arbitrary future object trajectories. However to this point a constant velocity assumption is used. In the future, we will furthermore integrate the motion prediction framework described in [Section 3.5](#), thereby rendering the current requirement for deterministic motion extrapolation superfluous.

Optimization Criteria As per the mentioned project-specific requirements, the V-Charge car is expected to follow the road graph as closely as possible while trying to keep a user-defined reference speed. Analogous to the control strategy and the sampling scheme described above, the cost function is hence also separated into lateral and longitudinal terms. In our implementation, the lateral part $J_{\perp}$ is composed out of the integrated absolute distance d between road graph and the candidate trajectory, as well as a terminal cost penalizing distance from the center of the reference lane at the end of a sample at the simulation time horizon T_{sim}

$$J_{\perp} = c_1 \int_0^{T_{sim}} d(t) dt + c_2 d(T_{sim}). \quad (4)$$

Similarly, the longitudinal cost term integrates speed offsets with respect to the reference speed and also contains term that penalizes terminal speed error

$$J_{\parallel} = c_3 \int_0^{T_{sim}} |v(t) - v_{ref}(t)| dt + c_4 |v(T_{sim}) - v_{ref}(T_{sim})|. \quad (5)$$

Note that trajectories in collision with any object in the scene are assigned infinite costs. We furthermore perform a (soft) penalization of trajectories passing in close proximity to objects in order to keep a desirable safety distance whenever possible. This strategy also mitigates boundary effects inherent to sampling-based methods.

Additionally, a regularization term penalizing deviations from the previous solution in between consecutive planning cycles was introduced in order to provide a smooth driving behaviour.

3.5 Interactive Motion Prediction (ETHZ-ASL)

The model-based motion planning strategy proposed in [Section 3.4](#) has been shown to work reliably and effectively in quasi-static or completely predictable environments. In an application such as the V-Charge parking scenario, multiple decision-making agents may share the same environment, however. It is then natural to ask how to model interaction between decision-making agents and how any remaining unmodeled effects may influence motion planning performance and motion safety. Clearly, the assumption that other agents remain at their current location (characteristic for zero-order methods, such as (social) potential-field approaches) or move predictably (e.g. via open-loop first-order extrapolations, see [\[Kuwata et al., 2009\]](#)) becomes inaccurate for even moderate environmental dynamicity. By contrast, we propose to model *interaction* (i.e. the steady-state solution which emerges after an agreement on avoidance has been reached), directly. This requires a model of other agents' behavior and is hence inherently linked to motion planning and control. It is this observation that led to the decision to report on motion prediction functionality in WP 4 instead of WP 3 (as originally planned).

Local Interaction A common method to model interaction (and the uncertainty associated with it) is via probabilistic methods. Such methods’ prohibitive complexity typically requires them to be run within a short-horizon partial motion planning [Petti and Fraichard, 2005] or multi-resolution [Kushleyev and Likhachev, 2009] framework, however. As a consequence, it has been shown that the build-up of uncertainty during interaction may lead to local minima, making them unsuitable for interactive navigation. This is known as the frozen robot problem [Trautman and Krause, 2010]. Attempts at fixing a maximal uncertainty a priori [DuToit, 2010] have led to somewhat disappointing results.

Our proposed method, by contrast, circumvents this problem by specifying a predictive-interactive element that all agents need to adhere to. This is a particularly attractive option for cases, where all agents are under the control of a central operator (such as in the V-Charge project), but where central optimization is too costly for real-time operation. We use the term *predictive* to denote the method’s foresightedness. *Interactive*, on the other hand, indicates its ability to implicitly negotiate steady-state avoidance strategies among nearby agents based on the *Golden Rule*. Applied to agent interaction, the Golden Rule requires that each agent partakes at least 50% to the avoidance maneuver. The Reciprocal Velocity Obstacles (RVO) method [van den Berg et al., 2008], a popular local collision avoidance approach, incorporates this strategy. It allows to create a (deterministic) partition in velocity space that is deemed *fair* for the respective agent to select motion commands from (corresponding to at least 50% participation in the arising avoidance maneuver), under the assumption that other agents adhere to a similar rule set. Note that the method, despite being deterministic, does not predict the future location of interacting agents directly, but rather specifies unfair regions, towards which other agents are not allowed to move. In situations where other agents do not completely adhere to these rules, re-planning gradually resolves any remaining inconsistencies.

However, RVO has in closed-form until recently only been formulated for single integrators, i.e., agents modeled according to a kinematic point mass. Our contribution, the C^n -CO method, generalizes this framework to robots able to follow C^n -continuous trajectories, which encompasses many (simple) models of robots, including cars. We do not know of any other method which provides this kind of system-compliance and interactive fidelity in absence of centralized optimization. In Figure 9 eight agents—employing the proposed method and constrained to C^1 -continuous trajectories—exchange their position diametrically on a circle.

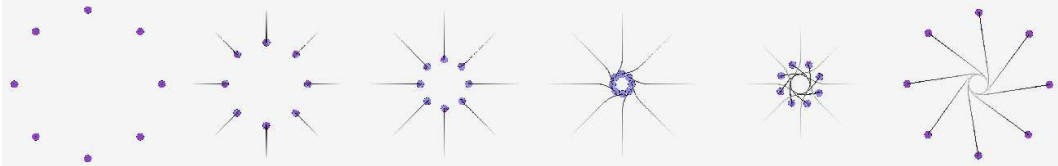


Figure 8: Predictive-interactive planning. Eight agents arranged on a circle exchange their position diametrically which leads to a bottle-neck at the center of the circle. By introducing a slight avoidance preference to a common side, an efficient avoidance maneuver reminiscent of a roundabout emerges.

Integration into Partial Search VO-based methods are local by construction and are thus prone to local minima when navigating in cluttered environments. We approach this issue by proposing to merge C^n -CO with the partial motion planning strategy presented in Section 3.4. Following [Kushleyev and Likhachev, 2009], we introduce a time horizon τ (with $\tau \geq t_{\text{stop}}$, the minimal stopping time) linked to the dynamic capability of the vehicles and to the edge traversal time of the partial motion planning method. For time steps larger than τ , linked to prediction fidelity, dynamic objects may then be safely neglected. Edges of the partial motion planning method with a search depth of one are then checked for fairness per the C^n -CO method, while edges at higher depth are not. Figure 9 illustrates this concept.

The full journal article describing C^n -CO [Ruffi et al., 2013], is appended to this deliverable. Please refer to Appendix A. At this stage, the method is implemented in C++ and has been thoroughly tested in

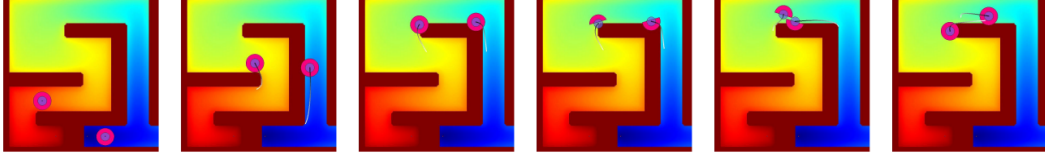


Figure 9: Predictive-interactive planning. Two agents employ the proposed interactive prediction method locally. Globally they follow a (non-interactive) heuristic function that guides them to their respective global goal at the other end of the maze.

simulation, but not yet on the V-Charge car.

3.6 Parking Spot Task Processor (RBG)

The parking spot task processor provides collision-free trajectories to park into or pull out of assigned parking spots and charging stations. The focus lies on short computation time and intuitive forms of trajectories. In agreement with these priorities an algorithm using a combination of geometric path generation (based on polynomials) and graph-search (based on A* in a 3D state-lattice, see [Pivtoraiko et al., 2009]) has been developed.

Method Overview The algorithm attempts to generate a geometric path first (Figure 10, Step-1). The geometric path generation computes a fifth-order polynomial to connect two poses in a 4D state-space (including 2D position, heading and curvature) and considers kinematic constraints of the vehicle. If the generated trajectory is collision-free, then the trajectory is provided as a solution (Figure 10, Case a). If the method cannot connect the two poses due to kinematic constraints, or if the resulting trajectory causes a collision with obstacles (Case b), a more costly graph-search is triggered (Step-2). The A* graph search is implemented on a 3D state space with currently a priori fixed resolution of 0.1 m to 0.2 m in position and 15° to 22.5° in heading. In order to guarantee accuracy at the goal pose, the state space is aligned so that the goal pose is exactly located on one of the states. Situation-dependent variable resolution remains as possible further work.

The geometric method provides smooth trajectories between two poses quickly. However, its flexibility is limited because it is impossible to compute solutions that are a priori collision-free or implement driving direction changes. On the other hand, since there is guaranteed to be no driving direction change, all trajectories provided by the geometric trajectory generation method are assumed to comply with our intuitive solution quality metric.

The graph-search method, on the other hand, finds collision-free trajectories with high probability in various situations (high flexibility), but it takes longer time to find a trajectory due to necessary iterations. In order to overcome this disadvantage of the graph-search, some precomputations have been implemented: Motion primitives and driving swath (area passed over by the vehicle) are calculated off-line and looked up during on-line planning. Furthermore, a Heuristic Look-Up Table is created off-line and used during on-line planning to reduce the number of iterations required for convergence [Knepper and Kelly, 2006].

Experimental Validation The presented parking method has been implemented in C++ and tested in simulation as well as on the V-Charge platform. Below we provide some preliminary performance evaluation results. The performance metric considers computation time and “trajectory quality”.

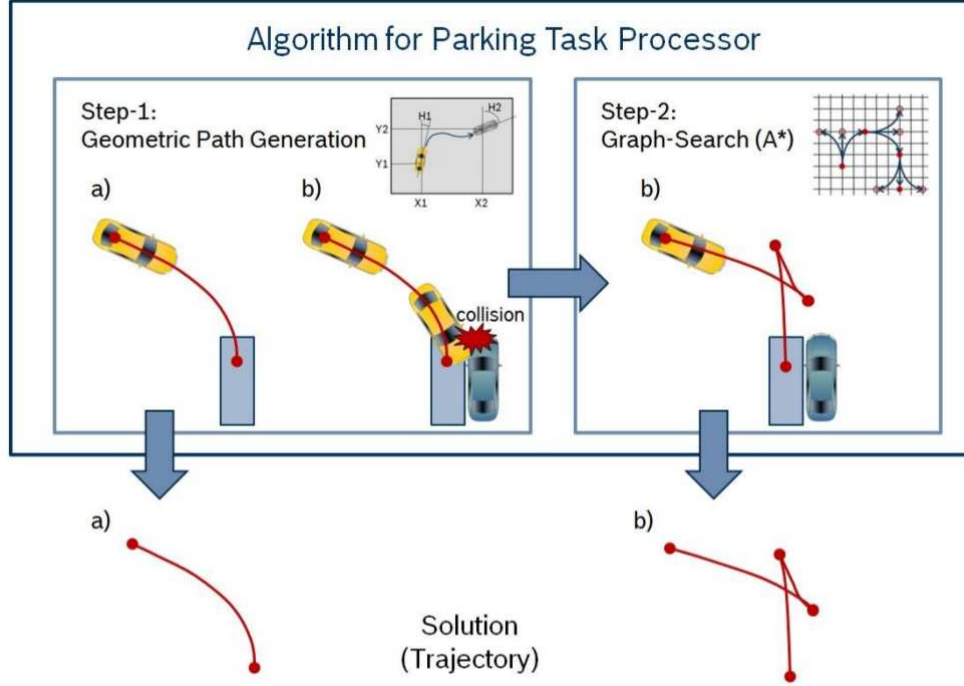


Figure 10: Parking task processor. The algorithm is composed out of two steps. First, it attempts to connect start and goal pose geometrically. If this fails, a 3D graph search on a state lattice is initiated.

For the performance evaluation, a simulation environment was prepared. As the calculation time of the graph-search highly depends on situations, some simulation scenarios (lateral parking scenario) with various start-poses (different positions and different headings) have been defined (see Figure 11-Left). Over a large variety of start-goal pose combinations solution computation time is averaged and the trajectory quality is analyzed. The simulation is executed on a computer with a 1.6 GHz CPU and 8 GB of RAM.

In Scenario A of Figure 11 there is no case where the geometric path generation provides collision-free trajectories. In all cases, trajectories are found by the graph-search, however resulting in an average calculation time of less than 1 s on a state-space resolution of 0.2 m in position and 15° in heading. For Scenario B, in 14% of the cases the geometric path generation provides collision-free trajectories. The calculation time for the geometric path generation is less than 5 ms. The averaged calculation time for the graph-search in all other cases is comparable to Scenario A.

Regarding the trajectory quality, the number of driving direction changes have been analyzed and compared between various settings for the penalty costs (see Figure 11-Right). Further criteria to classify a trajectory as “not intuitive” e.g. based on the locations of driving direction changes have not been defined yet.

In the light of these results, we intend on modifying the current algorithm as follows: one, further speed up graph search by optimizing memory access. Two, implement a situation-dependent variable state space resolution. Three, derive a measure to select penalty costs resulting in intuitive forms of parking maneuvers.

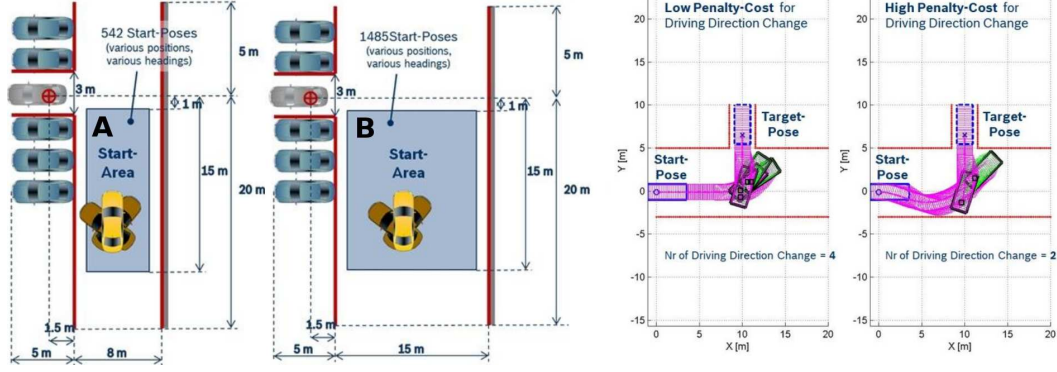


Figure 11: Experiments. *Left*: experimental setup consisting of a wide and a narrow starting area. *Right*: Influence of driving-direction penalty cost on resulting parking maneuvers.

4 Auxiliary Modules

This section describes related auxiliary modules that facilitate development and testing of WP 4 core functionality.

4.1 Simulation Environment V-Rep (ETHZ-ASL)

In order to safely test motion planning strategies and evaluate the performance of the V-Charge navigation system, a simulation environment based on the open-source simulator *Virtual Robot Experimentation Platform* (V-Rep) by Coppelia Robotics was built. This required the integration of an interface to the V-Charge middle-ware RTI *Connex DDS* (employed for message passing on the physical cars). All relevant sensor data returned by the V-Charge platform including wheel tick information, ultrasonic sensor data, localization ground truth as well as a camera-based 2.5D stixel map and binary occupancy grid information have been implemented.

We employ the simulator in two modes of operation. *Full simulation* allows for rapid prototyping and initial sanity testing of new control or motion planning functionality. *Hardware in the loop* operation furthermore allows to employ the simulator as an interactive visualizer. In this mode of operation, the physical car operates in the real world while *also* reacting to artificial objects placed in the virtual scene. This gives us the ability to safely test mature parts of our software under close to real-world conditions without endangering humans or the platform itself. Figure 12 displays a scene created with V-Rep containing static parked cars as well as moving pedestrians.

5 Conclusion

In this deliverable we gave an overview on scientific and implementation progress related to WP 4 and the appended publications provide further detailed descriptions on important aspects. Furthermore, all of the modules described have been integrated on the car and extensively tested—except for the motion prediction and interaction framework, which is available in C++ and has been tested in simulation only so far. In conjunction with the publications attached to this deliverable we believe WP 4 to be well on track with its scientific, engineering, and integration goals.

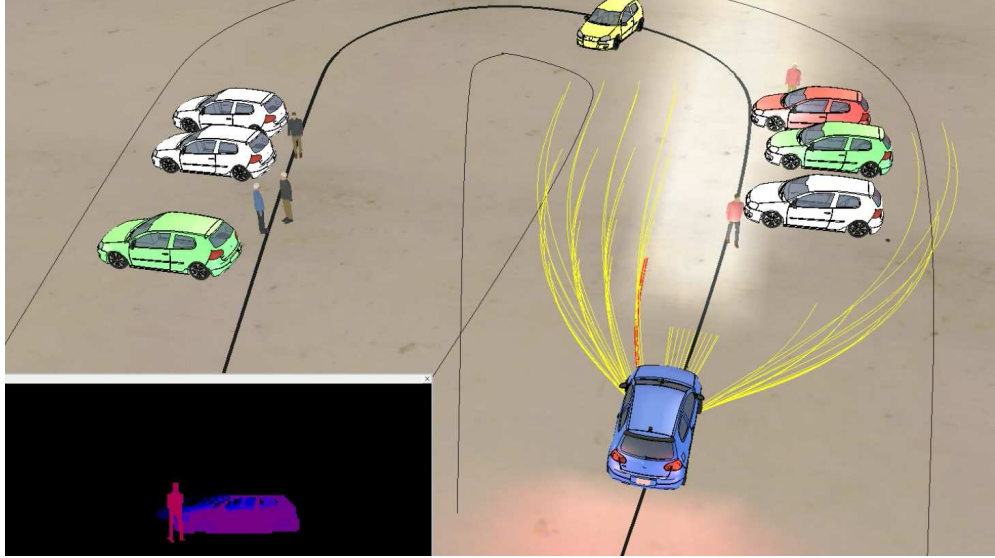


Figure 12: Scene created within Coppelias Robotics’ V-Rep simulator. In this scene the ego vehicle navigates amongst both static, parked cars as well as moving pedestrians. A binary occupancy grid (not displayed) is created from a vision sensor’s depth buffer, shown in the lower left corner. Yellow and red lines indicate the local motion planner’s explored candidate and selected trajectories, respectively.

A Appendices

The following publications describe the local motion planning functionality and the motion prediction framework in detail. They are appended to this deliverable.

	Contribution	Institution	Link
1	<i>A Sampling-Based Partial Motion Planning Framework for System-Compliant Navigation along a Reference Path</i> , U. Schwesinger, M. Ruffi, P. Furgale, R. Siegwart. Proc. of the IEEE Intelligent Vehicles Symposium, 2013	ETHZ-ASL	Section 3.4
2	<i>Reciprocal Collision Avoidance with Motion Continuity Constraints</i> , M. Ruffi, J. Alonso-Mora, R. Siegwart. IEEE Transactions on Robotics, 2013	ETHZ-ASL	Section 3.5
3	<i>Parking Spot Task Processor, Year 2 Progress Report</i> , H. Mielenz, K. Mimura. Technical Report, 2013	RBG	Section 3.6

References

- [DuToit, 2010] DuToit, N. (2010). Robotic Motion Planning in Dynamic, Cluttered, Uncertain Environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.
- [Knepper and Kelly, 2006] Knepper, R. A. and Kelly, A. (2006). High Performance State Lattice Planning Using Heuristic Look-Up Tables. In *Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems (IROS)*.

- [Kushleyev and Likhachev, 2009] Kushleyev, A. and Likhachev, M. (2009). Time-bounded Lattice for Efficient Planning in Dynamic Environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.
- [Kuwata et al., 2009] Kuwata, Y., Karaman, S., Teo, J., Frazzoli, E., How, J. P., and Fiore, G. (2009). Real-Time Motion Planning With Applications to Autonomous Urban Driving. *Control Systems Technology, IEEE Transactions on*, 17:1105–1118.
- [Petti and Fraichard, 2005] Petti, S. and Fraichard, T. (2005). Partial Motion Planning Framework for Reactive Planning Within Dynamic Environments. In *Proceedings of the IFAC/AAAI International Conference on Informatics in Control, Automation and Robotics*.
- [Pivtoraiko et al., 2009] Pivtoraiko, M., Knepper, R. A., and Kelly, A. (2009). Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(1):308–333.
- [Rufli et al., 2013] Rufli, M., Alonso-Mora, J., and Siegwart, R. (2013). Reciprocal collision avoidance with motion continuity constraints. *IEEE Transactions on Robotics (T-RO)*.
- [Samson and Ait-Abderrahim, 1991] Samson, C. and Ait-Abderrahim, K. (1991). Feedback control of a nonholonomic wheeled cart in cartesian space. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 1136 –1141 vol.2.
- [Schwesinger et al., 2013] Schwesinger, U., Rufli, M., Furgale, P., and Siegwart, R. (2013). A sampling-based partial motion planning framework for system-compliant navigation along a reference path. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IVS)*.
- [Trautman and Krause, 2010] Trautman, P. and Krause, A. (2010). Unfreezing the Robot: Navigation in Dense, Interacting Crowds. In *Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems (IROS)*.
- [van den Berg et al., 2008] van den Berg, J., Lin, M. C., and Manocha, D. (2008). Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

A Sampling-Based Partial Motion Planning Framework for System-Compliant Navigation along a Reference Path

Ulrich Schwesinger¹, Martin Rufli¹, Paul Furgale¹ and Roland Siegwart¹

Abstract—In this paper a generic framework for sampling-based partial motion planning along a reference path is presented. The sampling mechanism builds on the specification of a vehicle model and a control law, both of which are freely selectable. Via a closed-loop forward simulation, the vehicle model is regulated onto a carefully chosen set of terminal states aligned with the reference path, generating system-compliant sample trajectories in accordance with the specified system and environmental constraints. The consideration of arbitrary state and input limits make this framework appealing to nonholonomic systems. The rich trajectory set is evaluated in an online sampling-based planning framework, targeting real-time motion planning in dynamic environments.

In an example application, a Volkswagen Golf is modeled via a kinodynamic single-track system that is further constrained by steering angle/rate and velocity/acceleration limits. Control is implemented via state-feedback onto piecewise C^0 -continuous reference paths. Experiments demonstrate the planner’s applicability to online operation, its ability to cope with discontinuous reference paths as well as its capability to navigate in a realistic traffic scenario.

I. INTRODUCTION

While autonomous mobile robots have been developed since the 1960s, research on autonomous cars only intensified in the 1990s—particularly at Carnegie Mellon University (Navlab project [1], 4500 km of autonomous driving) and the University of Parma (ARGO Project [2], 2000 km of autonomous driving). At that time, operation was restricted to lane following and platooning, however. Later, the various DARPA challenges, culminating in the 2007 Urban Challenge, bundled efforts that led to systems approaching fully autonomous operation in urban-like scenarios, including interaction with other traffic participants and adherence to traffic regulations.

One of the central building blocks required for autonomous operation—the navigation framework—remains inherently difficult to approach in a unified algorithm. As a consequence, hierarchical frameworks consisting of a global planner (commonly, graph search) and a system-compliant local planner have found general acceptance. To ensure safety in dynamic environments, the local planner must ensure that returned paths are actually dynamically feasible—that they are within the dynamic capabilities of the robotic platform. Hence, the boundary between planning and traditional control becomes somewhat blurred. In this paper, we

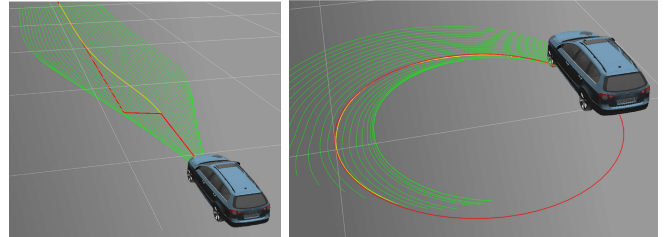


Fig. 1: *Left*: Trajectories with a fixed longitudinal speed profile (green) generated with respect to a C^0 -discontinuous reference path (red) and the minimum cost trajectory (yellow). *Right*: Trajectories generated for a reference path with curvature close to the vehicle’s maximum steering angle.

exploit this connection to develop a partial motion planning framework that utilizes the system model and controller to ensure that the resulting plans are dynamically feasible by construction.

A. Related Work

Related work on *local* motion planning & motion control is commonly separated into approaches that compute solution curves directly, and such that require a reference path (or trajectory) for guidance. In the former case, classic collision avoidance methods [3], [4] have gradually been replaced by state lattice graphs [5], [6] and Rapidly-exploring Random Tree (RRT) [7] methods. Nonetheless, these methods remain restricted to approximately four dimensional state spaces if online operation is desired—thereby limiting the fidelity of the agent models they can faithfully reproduce. Consequently, they typically find application in situations where no external structure can be extracted for guidance, such as off-road or on large-scale parking lots.

If a reference path (or trajectory) is available for guidance, higher-dimensional state-space representations become approachable. Road and lane information in particular enact narrow constraints on heading and curvature values. A common approach is thus to align the endpoints of local trajectory samples with the reference path [8]. This technique simultaneously reduces the search complexity and overcomes the danger of entering unsafe end states. Most of these approaches are based on geometric primitives instead of actual vehicle models, separating the computation of velocity profiles from the geometric construction of the path [9], [10], [11], [12]. This separation may result in conflicts with non-holonomic platform constraints—especially at low speeds. Consequently, trajectories need to be validated in a post-

*The research leading to these results has received funding from the European Union Seventh Framework Programme FP7/2007-2013, Challenge 2, Cognitive Systems, Interaction, Robotics, under grant agreement No 269916, V-Charge (<http://www.v-charge.eu>).

¹Autonomous Systems Lab, ETH Zurich, Tannenstrasse 3, 8092 Zurich
Corresponding author: sculrich@mavt.ethz.ch

processing step—which may in turn lead to the pruning of a considerable amount of candidate motions [8]. In contrast, the local on-road planner described in Urmson et al. [13] generates motion samples by solving two-point boundary value problems subject to the dynamics of a detailed vehicle model directly [14]. Unfortunately, this method is prone to convergence issues related to the non-invertibility of non-holonomic vehicle models.

We thus observe that existing approaches face difficulties in capturing system constraints characteristic for autonomous cars (including steering angle, steering rate, and acceleration limits, as well as reaction delays) explicitly. Furthermore, non-holonomic constraints cannot be integrated—nor inverted—in closed form.

B. Contributions

Our approach presented in this paper fully addresses these important real-world platform and environmental constraints without having to revert to model inversion techniques. It proposes a *generic* sampling-based framework requiring only (i) a reference path verifying piece-wise $C^{\geq 0}$ continuity, (ii) a system description suitable for numerical forward propagation, and (iii) obstacle information expressed in vehicle frame. Its output consists of a system-compliant command sequence, which may be employed for feed-forward control of the platform directly.

The remainder of this paper is organized as follows: in Section II we introduce the overall structure of the proposed motion planning framework. Section III then describes a sample implementation for a Volkswagen Golf. Section IV reports on experiments conducted in simulation as well as during physical runs on a full-scale test vehicle. Finally, Section V concludes this paper and sketches further work.

II. FRAMEWORK

This section introduces the main contribution of this paper, consisting of a novel framework for partial motion planning along reference paths. The proposed planning framework consists of a graph search routine, a system model and a control law. Graph search operates on a tree that is constructed online. Nodes, n , represent particular vehicle state/time pairs, $(\mathbf{x}, t)$, while edges, e , are constructed by system-compliant trajectory segments. An outline of a single planning cycle is portrayed in Algorithm 1.

The search routine operates online at a fixed rate in accordance with environmental requirements. During initialization, any remaining nodes in N are removed. Then, the most recent system state, $\mathbf{x}_0$, obtained from the localization module is inserted at the root of the tree at time t_0 . To account for the planning time, T_{cycle} , during which the planner is busy, the root vertex is expanded according to the initial segment of the solution found during the previous search iteration (if available). Actual search thus begins from $(\mathbf{x}_1, t_0 + T_{\text{cycle}})$ using a user-supplied graph search method which needs to provide *push* and *pop* operations to insert and retrieve items from the search queue, Q , respectively.

Algorithm 1 Planning cycle

```

1:  $N \leftarrow \emptyset, E \leftarrow \emptyset$ 
2:  $n \leftarrow (\mathbf{x}_0, t_0), N \leftarrow N \cup \{n\}$ 
3:  $(n^*, e^*) \leftarrow \text{simulate}(n, \mathbf{f}, \mathbf{g}, T_{\text{cycle}}, \mathbf{u}_0)$ 
4:  $N \leftarrow N \cup \{n^*\}, E \leftarrow E \cup \{e^*\}$ 
5:  $Q.\text{push}(n^*)$ 
6: for  $l = 1 \rightarrow \text{desired tree depth}$  do
7:   for all nodes at tree level  $l$  do
8:      $n \leftarrow Q.\text{pop}()$ 
9:     for  $j = 1 \rightarrow |M|$  do
10:       $(d_{\text{ref}}, v_{\text{ref}}) \leftarrow \text{drawSample}(M)$ 
11:       $(n^*, e^*) \leftarrow \text{expand}(n, \mathbf{f}, \mathbf{g}, d_{\text{ref}}, v_{\text{ref}}, T_{\text{sim}})$ 
12:       $N \leftarrow N \cup \{n^*\}, E \leftarrow E \cup \{e^*\}$ 
13:       $Q.\text{push}(n^*)$ 
14:    end for
15:  end for
16: end for
17:  $\text{computeCosts}(E, N)$ 
18:  $\hat{n} \leftarrow \text{minimum cost node at last tree level}$ 
19:  $\hat{e} \leftarrow \text{edge in tree level 1 leading to } \hat{n}$ 
20: return  $\hat{e}$ 

```

Within a loop, pop operations are then triggered one at a time leading to node expansion. Successor edges are constructed by (numerical) forward propagation of a general user supplied vehicle model, $\mathbf{f}$, with

$$\mathbf{x}(t + \Delta t) = \mathbf{f}([\mathbf{x}(t), \dot{\mathbf{x}}(t), \dots], \mathbf{u}(t), t, \Delta t). \quad (1)$$

Uniquely, this allows to incorporate arbitrary system descriptions—even non-linear and non-invertible ones that are hard to work with in competing approaches. In order to contain the size of the state space when using high-dimensional system representations, however, a control-based approach building on terminal manifolds is enforced. The notion of the terminal manifold [8] refers to a reduced subspace of the state-time space towards which node expansion is directed. In our framework it takes the form of heading and curvature alignment (θ and c) with respect to a user-supplied reference path,

$$\mathbf{z}(s) = \begin{bmatrix} \theta(s) - \theta_{\text{ref}}(s) \\ c(s) - c_{\text{ref}}(s) \end{bmatrix} = \mathbf{0} \quad \forall s > s_0, \quad (2)$$

leaving the desired lateral offset to the reference path, d_{ref} , as well as the desired velocity, v_{ref} , at the curvilinear abscissa, s , as free variables. The alignment is enforced for all states ahead of the current curvilinear abscissa, s_0 . Along the reference path, the terminal manifold, $\mathbf{z}$, is then sampled into a set of terminal states, $M = D \times V$, built from a discrete set of lateral offsets, D , and longitudinal vehicle velocities, V . During node expansion, samples $m = (d_{\text{ref}}, v_{\text{ref}})$, are iteratively drawn from M and used as a target states for closed-loop system control. The function `expand` uses a user-supplied controller, $\mathbf{g}$, with $\mathbf{u}(t) = \mathbf{g}(\mathbf{x}(t), m)$ to regulate the forward-simulated system model $\mathbf{f}$ towards the sample m over a fixed time period, T_{sim} . The newly retrieved state and its incoming trajectory segment are pushed back onto Q as a node/edge-pair, (n^*, e^*) .

Once the tree has been expanded to a desired search depth (or maximum planning time has elapsed), edge and accumulated node costs are computed. Then, the lowest

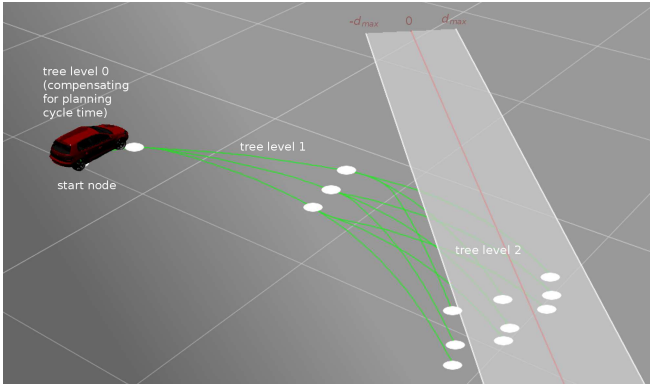


Fig. 2: A tree composed of system-compliant trajectory segments. The edge in tree level zero corresponds to the initial part of the solution found during the previous search iteration. For subsequent tree levels, the lateral terminal manifold is (for illustrative purposes) discretized into a coarse set of three samples $\{-d_{\max}, 0, d_{\max}\}$.

cost node at maximal search depth is identified and back-propagated, which results in the construction of the solution trajectory. Depending on the fidelity of the system model employed during edge construction, inputs corresponding to the solution trajectory may be directly executed on the platform as feed-forward commands.

Figure 2 illustrates a search tree created using a coarse sampling of the terminal manifold. For illustrative purposes, only one velocity profile and three lateral offsets are explored in a tree of depth three.

III. APPLICATION TO AN AUTONOMOUS CAR

In this section, we illustrate the proposed framework of Section II by applying it to the problem of on-road autonomous driving using lane centers as reference. Our experimental platform—both in simulation and real-world experiments—is a VW Golf prepared for the European V-Charge project.

A. System Modeling

The described operating conditions require a modeling approach that is particularly suitable for low-velocity maneuvers, where non-holonomic constraints dominate. Nonetheless, it also needs to maintain accuracy at higher velocities, provided steering and acceleration commands remain within comfortable ranges and operation is limited to surfaces with sufficient traction (i.e., to paved roads).

The kinodynamic single-track model fulfills all of these conditions. At the same time it remains suitably efficient to implement and forward simulate. It combines 2D position, (x/y) , heading, θ , steering angle, ϕ , and longitudinal speed,

v into its state vector, $\mathbf{x}$. The dynamics are then captured by

$$\underbrace{\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \\ \dot{v} \end{bmatrix}}_{\dot{\mathbf{x}}} = \begin{bmatrix} \cos(\theta) \cdot v \\ \sin(\theta) \cdot v \\ \frac{v}{L} \tan(\phi) \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}}_{\mathbf{u}} \quad (3)$$

This translates into a system model as specified in equation 1 via $\mathbf{f} = \mathbf{x}(t_0) + \int_{t_0}^{t+\Delta t} \dot{\mathbf{x}}(t) dt$. The inputs take the form of steering speed, u_1 , and longitudinal acceleration, u_2 , stacked into the control vector, $\mathbf{u}$. L represents the inter-axle distance of the vehicle. In addition, state and actuator constraints are considered during numerical integration. We limit steering angle, $\phi \leq \phi_{\max}$, steering speed, $u_1 \leq \dot{\phi}_{\max}$, longitudinal speed, $v \leq v_{\max}$, longitudinal acceleration, $u_2 \leq \dot{v}_{\max}$, and longitudinal deceleration, $u_2 \geq \dot{v}_{\min}$. Numerical values used in our experiments derived from the manufacturer's specifications are given in Table I. Retrieving a more involved system model via an identification step is left as future work.

Parameter	Value
maximum steering angle, $\phi_{\max}$	0.64 rad
maximum steering speed, $\dot{\phi}_{\max}$	0.57 rad/s
longitudinal acceleration, $\dot{v}_{\max}$	1.0 m/s ²
longitudinal deceleration, $\dot{v}_{\min}$	-1.5 m/s ²
inter-axle distance, L	2.578 m

TABLE I: Vehicle parameters for the VW Golf platform. Acceleration limits selected to ensure comfort.

B. Controller Design

Control of the single-track vehicle model given in (3) onto a set of reference trajectories following the road (and thus the generation of a set of locally expressive trajectory segments) is achieved via the nonlinear state-feedback controller described in [15]. The controller is designed to follow a moving reference cart, where center of the rear vehicle axle serves as reference position. The error states for the lateral control are given by the heading difference, $\Delta\theta$, and the lateral distance, Δd , to the reference cart in the ego-vehicle frame, both depicted in Figure 3. The reference cart is placed with a speed-dependant look-ahead along the reference path. The lateral control output for the turning rate ω then is

$$\omega = \omega_f + k_1 v_{ref} \frac{\sin(\Delta\theta)}{\Delta\theta} \Delta d - k_2 \Delta\theta \quad (4)$$

with an optional feed-forward term for the turning rate, $\omega_f = v_{ref} c_{ref}$, based on the curvature of the reference path.

While this control law guarantees the regulation of the error states to zero eventually for an unconstrained system, the introduction of actuator constraints may negate these guarantees. In practice we have been successful in scheduling the lateral control gains k_1 and k_2 for specific longitudinal

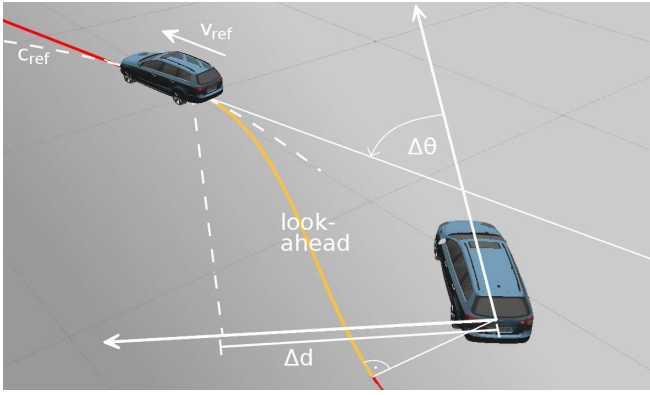


Fig. 3: Control variables for lateral control, the nonlinear state feedback regulates the error states $\Delta\theta$ and Δd to zero.

speed ranges. Note that a proof of stability is in this case not required, as the controller only serves as a generator of a locally expressive trajectory set, which is then assessed for suitability by the optimization routine, i.e., the cost function.

C. Cost Function

The design of the cost function should thus both penalize divergence from the reference path as well as rewarding progress along it. In general, these two criteria are conflicting in situations where the vehicle is not well aligned with the reference path. Due to the nonholonomic constraints acting on the platform, the vehicle then needs to deviate from the reference path in order to make progress along it. To generate a behavior that attempts to fulfill both close and quick path following, we linearly combine a lateral cost J_d with a longitudinal cost J_s according to Equation (7).

$$J_d = \frac{1}{d_{max}c_f} \int_0^{c_f} d(c) dc \quad (5)$$

$$J_s = \frac{c_f}{v_{max}T_{sim}} \quad (6)$$

$$J = kJ_d + (1 - k) J_s \quad (7)$$

J_d integrates the lateral distance to the reference path over the curvilinear abscissa c , where c_f represents the curvilinear abscissa of the reference path associated with the final state of the simulated trajectory. The integral penalizes a divergence of the platform from the reference path along the simulated trajectory, which counteracts shortcutting in sharp turns. The longitudinal cost term, on the other hand, is comprised of the final curvilinear abscissa c_f , normalized by the largest feasible travel distance $v_{max}T_{sim}$. Hence, both cost terms are normalized to lie within the range $[0, 1]$, and an intuitive combination into a single scalar cost term J becomes feasible. Via the factor k , preference between close versus rapid path following can be expressed.

Trajectories in collision with an obstacle at any single time during simulation are assigned infinite cost.

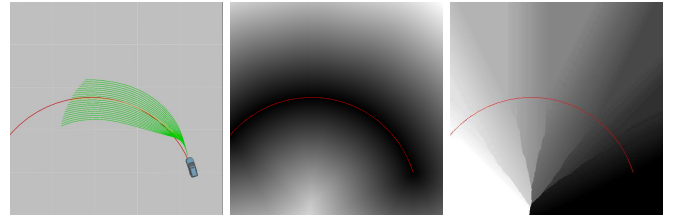


Fig. 4: *Left*: Reference path (red) and forward simulated trajectories (green) that have been obtained by controlling the vehicle model onto the terminal manifold. *Center*: Path distance transform of the reference path. Lighter colors refer to larger distances. *Right*: Voronoi labeling. Colors refer to the curvilinear abscissa of the associated reference path item.

D. Implementation Details

This section presents implementation details for aspects of our sample application.

a) Collision Checking: The static map is represented as an occupancy grid. To perform fast collision checks, we compute a one-time dilatation on the grid at the start of the planning cycle with the distance transform described in [16]. The rectangular shape of the car is approximated by a set of discs. The dilatation map is thresholded by the disk radius which allows for a single look-up collision check per disk. Collision checking with dynamic obstacles is achieved in a similar manner. Predicted positions and timestamps of obstacles are stored in an occupancy grid. Using a one-time dilatation we obtain a map that, at a given position, allows for fast look-ups of timestamps associated with that position's spatially closest dynamic obstacles. Note that this method is limited to situations where predicted obstacle paths do not overlap. For spatially colliding objects the time-gap is inspected.

b) Path Distance Transform: Our proposed controller requires the orthogonal projection of the current position onto the reference path for all visited cells during the forward simulation of the system. For a fast (approximate) computation of said projection, we calculate in a pre-computation step the Voronoi decomposition of the discretized reference path with the method described in [17]. Each cell in the map is associated with the curvilinear abscissa of the nearest reference path item. The path distances themselves are needed to calculate the lateral cost term J_d . A sample of the output of this step is displayed in figure 4.

c) Parallelization: The map dilatation of the occupancy grid for fast collision checking and path distance transform including Voronoi decomposition are operations that have to be performed in a pre-processing step, but may run on individual cores in parallel. We found the implementation in *OpenCV* to operate sufficiently fast. In *OpenCV* $\geq 2.4.3$, the function `cv::distanceTransform` provides both distance transform and Voronoi decomposition used for orthogonal projection of any given position on the map onto the reference path. Computation times are given in table II.

The remainder of the total planning time is subsequently allocated to the processes of trajectory generation and col-

lision checking. Both of these operations can be fully parallelized among CPU and GPU cores, thereby offering the potential for substantial gains in search quality if adequate hardware is available. Computation times provided in table II are in terms of a single CPU.

IV. EXPERIMENTAL EVALUATION

Below we present experimental results based on the system and controller implementations described in Section III. We first analyze the effect of discontinuous and strongly curved reference paths on the closed-loop system response. Then, we qualitatively demonstrate the capabilities of our framework in simulated inner-city traffic scenario. Finally, we provide runtime results.

A. Sensitivity to Reference Path Shape

The reference path represents a key aspect of our framework in that it defines the target manifold of the sampling strategy. It is thus important that its shape does not negatively affect planning results. In this regard, figure 1 indicates that motion segments are not even unduly affected by path discontinuities and curvatures close to vehicle limits. In fact, it is the use of a closed-loop control strategy, which allows to obtain system-compliant trajectories despite reference path discontinuities. This property serves well in practice where global localization jumps may result in similar effects.

B. Navigation in Simulated Traffic Scenario

In a simulated inner-city traffic scenario, we demonstrate our planning framework’s capacity for complex decision-making—including lane and intersection handling, static and dynamic obstacle avoidance, as well as platooning¹. Importantly, the handling of these situations *emerges* from the interplay between the framework (in particular state-time space collision checking) and the environment and thus reduces the need for a separate situational awareness module.

Figure 5 shows an orthographic image of Zurich, Hegibachplatz together with an artificially created reference path for the ego vehicle and three dynamic obstacle tracks. Each dynamic obstacle track was populated with several circular dynamic objects of radius 1 m with a constant time gap of 4 s between them. Obstacle predictions for the planner were taken from the known obstacle tracks. The reference travel speeds for the ego vehicle and the dynamic obstacles on tracks 1 and 2 were set to 8.33 m/s = 30 km/h, while those on track 3 were set to 2 m/s to cause a congestion.

We applied a binary mask to the static environment to mask impassable areas outside of drivable roads. In this regard note, that the tram displayed in the orthographic image was not considered in the scenario. The reference path for the ego vehicle was created based on a hand-drawn reference path in the image. Heading and curvature information was extracted by pixel-based finite differences, post-processed via a moving average filter for smoothing purposes. Despite the smoothing step, heading as well as steering angle information

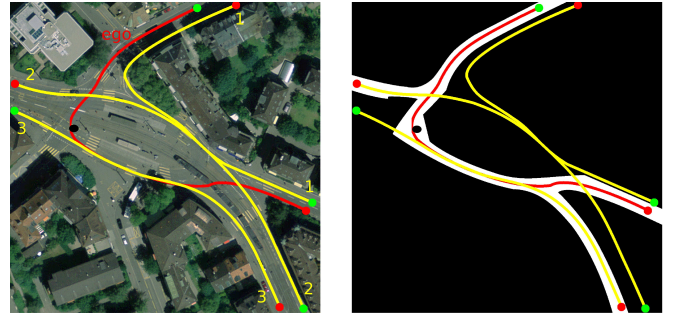


Fig. 5: Simulated run at Zurich, Hegibachplatz. *Left*: Orthographic image of Zurich, Hegibachplatz. Reference paths for the ego vehicle (red) and three dynamic obstacle tracks (yellow) are displayed, with their start (green)/end points (red) indicated. *Right*: Binary occupied(black)/free(white) space mask applied to the image, representing road information.

is rather noisy, which enables the demonstration of discontinuous reference information handling within our framework. After every planning cycle, the vehicle in simulation executes the initial parts of the solution $\mathbf{u} = [\dot{\phi}, \dot{v}]$ as feed-forward commands. In figure 6 four snapshots of the simulated run are displayed, capturing four distinct situations that challenge the planner’s capabilities: *Intersection handling*, *static obstacle avoidance*, *platooning* and *pure lane following*. In the initial part of the run ($t = [0\text{ s}, 10\text{ s}]$), the ego vehicle faces oncoming vehicles on track 1. The knowledge that their future movement is constrained to the oncoming lane allows the ego vehicle to avoid improper evasive maneuvers. After 10 s, while approaching the first intersection, the vehicle needs to yield to dynamic obstacles on track 2. The ego vehicle slows down to let the obstacles pass, as time does not permit pulling out in front of them. At $t = 18\text{ s}$, the vehicle encounters a static obstacle (black ellipse) in the middle of the intersection and steers to the right to pass by. This illustrates that the reference path is readily departed if circumstances call for it. Thereafter, the vehicle faces a convoy of four slow traveling obstacles with 2 m/s. It decelerates and follows the convoy with a constant time gap of 3 s as space does not permit an overtaking maneuver. As soon as the road is clear, the ego vehicle accelerates again. In order to come to a standstill at the end of the reference path the ego vehicle brakes at the end of the run. All these maneuvers emerged from the same cost function without switching between different behavior or parameter sets.

C. Framework Runtime Results

A key aspect of every online planning algorithm is its cycle runtime. Table II displays the execution times of the main computation steps of our algorithm on a single core of an Intel Core i7@2.67 GHz, 4 MiB cache. Using this setup, a set of approximately 3500 trajectories can be evaluated (based on 100 samples per trajectory) during a planning cycle of 200 ms. This set is large enough to retain adequate motion diversity as well as sampling density up to a tree depth of two. Parallelization and GPU implementations would further

¹A video of the run can be found at http://www.asl.ethz.ch/people/sculrich/hegibachplatz_iv13.avi.

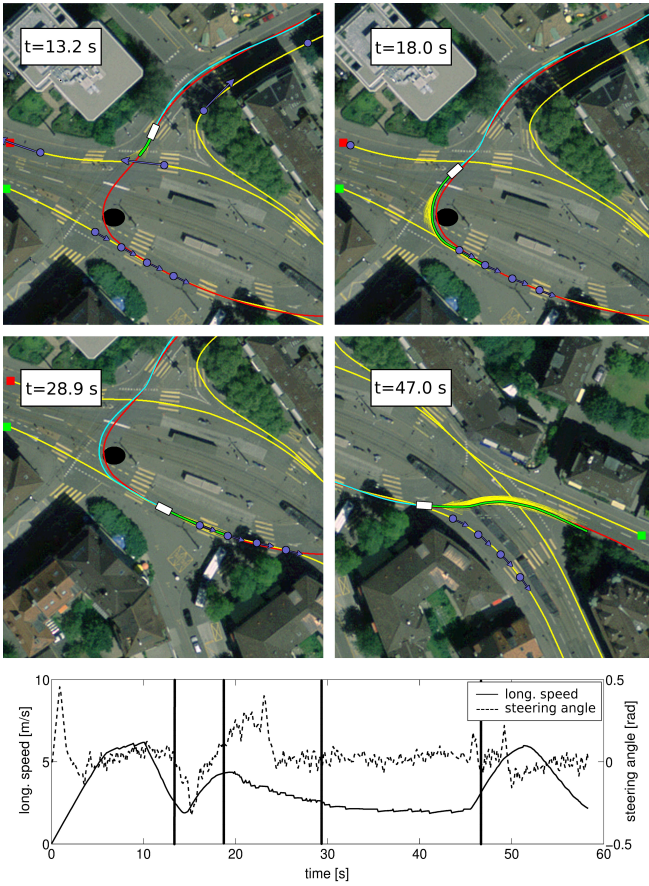


Fig. 6: *Top*: Snapshots of the run. The vehicle (odometry solution in cyan) successfully avoids dynamic obstacles (blue circles with velocity vectors) as well as static (black) ones while following the reference path (red line). From a set of explored trajectories (yellow lines), the best partial motion plan (green line) is chosen. *Bottom*: The graph shows longitudinal speed as well as steering angle of the ego vehicle. Vertical lines refer to the timestamps of the snapshots.

speed up the motion generation and collision checking steps.

operation	computation time
obstacle map dilatation	6 ms
path distance transform with Voronoi decomposition	6 ms
trajectory generation	43 ms
collision checking	11 ms

TABLE II: Computation times for the algorithm's main operations. The obstacle map contains 500×500 cells at 0.1m resolution. Trajectory generation and collision check times are expressed per 1000 trajectories and 100 samples.

V. CONCLUSION

We presented a sampling-based framework for online motion planning. Through a method capable of dealing

with arbitrary motion models and system constraints, the framework is widely applicable. Motion samples aligned with the reference path are generated via forward simulation of a vehicle model. A controller regulates the vehicle towards samples, representing a desired lateral offset and reference speed. The method is able to generate motions for discontinuous reference paths, where previous methods often failed. Due to the difficulty in controlling combustion engine cars at low speed, future research will focus on the identification of a system model for the real platform, capturing a probabilistic input-output map. Incorporating a probabilistic treatment of the ego motion uncertainty is intended.

REFERENCES

- [1] D. Pomerleau and T. Jochem, "Rapidly adapting machine vision for automated vehicle steering," *IEEE Expert*, vol. 11, pp. 19–27, apr 1996.
- [2] A. Broggi et al., "The argo autonomous vehicles vision and control systems," *International Journal of Intelligent Control and Systems*, pp. 409–441, 1999.
- [3] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *Robotics and Automation, IEEE Transactions on*, vol. 7, no. 3, pp. 278–288, 1991.
- [4] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *Robotics Automation Magazine, IEEE*, vol. 4, pp. 23–33, Mar. 1997.
- [5] M. Pivtoraiko and A. Kelly, "Constrained Motion Planning in Discrete State Spaces," in *Field and Service Robotics*, pp. 269–280, 2005.
- [6] M. Rufli and R. Siegwart, "On the design of deformable input- / state-lattice graphs," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 3071–3077, May 2010.
- [7] S. M. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," Tech. Rep. 98-11, Dept. of Computer Science, Iowa State University, 1998.
- [8] M. Werling et al., "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *The International Journal of Robotics Research*, vol. 31, pp. 346–359, Dec. 2011.
- [9] A. Bacha et al., "Odin: Team VictorTango's entry in the DARPA Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 467–492, 2008.
- [10] M. Montemerlo et al., "Junior: The Stanford entry in the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.
- [11] J. Bohren et al., "Little Ben: The Ben Franklin Racing Team's entry in the 2007 DARPA Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 598–614, 2008.
- [12] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," *Intelligent Robots and Systems (IROS)*, 2009.
- [13] C. Urmson et al., "Autonomous driving in urban environments: Boss and the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.
- [14] T. M. Howard and A. Kelly, "Optimal Rough Terrain Trajectory Generation for Wheeled Mobile Robots," *The International Journal of Robotics Research*, vol. 26, no. 2, pp. 141–166, 2007.
- [15] C. Samson and K. Ait-Abderrahim, "Feedback control of a nonholonomic wheeled cart in cartesian space," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pp. 1136–1141 vol.2, apr 1991.
- [16] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," tech. rep., Cornell Computing and Information Science, 2004.
- [17] G. Borgefors, "Distance transformations in digital images," *Comput. Vision Graph. Image Process.*, vol. 34, pp. 344–371, June 1986.

Reciprocal Collision Avoidance with Motion Continuity Constraints

Martin Rufli, *Member, IEEE*, Javier Alonso-Mora, *Student Member, IEEE*,
and Roland Siegwart, *Fellow, IEEE*

Abstract—This article addresses decentralized motion planning among a homogeneous set of feedback-controlled, *decision-making* agents. It introduces the *Continuous Control Obstacle* (C^n -CO), which describes the set of C^n -continuous control sequences (and thus trajectories) that lead to a collision between interacting agents. By selecting a feasible trajectory from C^n -CO's complement, collision-free motion is obtained. The approach represents an extension to the *reciprocal Velocity Obstacle* (RVO, ORCA) collision avoidance methods so that trajectory segments verify C^n continuity rather than piecewise linearity. This allows the large class of robots capable of tracking C^n -continuous trajectories to employ it for partial motion planning directly—rather than as a mere tool for collision checking.

This article further establishes that both the original Velocity Obstacle (VO) method and several of its recently developed reciprocal extensions (which treat specific robot physiologies only) correspond to particular instances of C^n -CO. In addition to the described extension in trajectory continuity, C^n -CO thus represents a *unification* of existing reciprocal VO theory.

Finally, the presented method is validated in simulation—and a parameter study reveals under which environmental and control conditions C^n -CO with $n > 0$ admits significantly improved navigation performance compared to inflated approaches based on ORCA.

Index Terms—Autonomous Agents, Path Planning for Multiple Mobile Robot Systems, Reciprocal Collision Avoidance

I. INTRODUCTION

SUCCESSFUL robot operation builds on at least three interconnected competences, namely localization, mapping and motion planning/control. The latter is concerned with computing a (lowest cost) path or trajectory between two configurations embedded in a cost field—while taking into account motion constraints, static obstacles, and dynamic objects. The important case where dynamic objects are *decision-making agents* forms the topic of this article.

A. Related Work

Specific problems of the motion planning problem have been studied since the late seventeenth century [1], but progress since has proven slow and difficult. To this day, only a few local (convex) problems concerned with stability and convergence—spanning the modern field of control theory [2]—are understood analytically. It is thus not surprising that the large remaining space of optimization problems has been attacked by approaches admitting various forms of approximations. In this regard, advances in deterministic graph

search [3, 4], graph representation [5, 6], and randomized sampling-based methods [7] have enabled (approximately optimal) solution strategies on a global scale including obstacles. The resulting fundamental separation between (global) planning and motion control has had a tremendous influence on mobile robotic navigation and led to systems fitted with multi-tiered navigation frameworks that often lacked coherence between the individual modules [8]. The recent development of asymptotically optimal sampling-based planning algorithms [9], and optimal tree-based approaches [10] has considerably softened this separation, and for the first time enabled truly unified *system-compliant* online motion planning with optimality guarantees. Consequently, robotic agents increasingly operate in challenging dynamic environments with applications as diverse as warehouse delivery [11], service robotics [12] and entertainment [13]. Nonetheless, if this operation takes place among other *decision-making* agents, system-compliant planning alone does not seem adequate to ensure performant navigation. Rather, it becomes important that the individual planning strategies are aware of (and take into account) that other agents are also engaging in a similar activity. Yet many prevalent navigation methods remain centralized and hence do not scale well [14, 15], or are limited to open-loop motion predictions instead of considering (collaborative) interaction directly [16, 17].

The Reciprocal Velocity Obstacle (RVO) method [18], by contrast, models robot interaction both in a decentralized manner and pairwise optimally. It restricts robots' action capabilities to a set of constant velocities. Under the assumption that other agents also continue their present motion along a straight line trajectory, future collisions may be estimated as a function of *relative velocity* alone. Although RVO was originally developed as a collision checking method for holonomic (i.e., omni-directionally actuable) robots and results in C^0 -continuous trajectories, its success in multi-robot applications paved the road towards a multitude of extensions to and revisions of the basic framework: the Optimal Reciprocal Collision Avoidance (ORCA) method [19] prevents reciprocal dances and casts the problem into a linear programming framework which can be solved efficiently. ORCA-DD [20] and NH-ORCA [21] are extensions to robots of type unicycle which artificially inflate the agents' diameter in compensation for the more constrained kinematics' inability to maneuver C^0 -continuous paths. However, these methods still subject robots to sudden changes in speed. The Acceleration Velocity Obstacle (AVO) method [22] extends the continuity of the RVO solution by one order (to a C^1 -continuous trajectory)

The authors are with the Institute of Robotics and Intelligent Systems, ETH Zurich, 8092 Zurich, Switzerland e-mail: firstname.lastname@mavt.ethz.ch. Javier Alonso-Mora is furthermore with Disney Research Zurich, Switzerland.

and thus obtains a smoother response that can be followed by some dynamically constrained vehicles. Nevertheless, for several practically relevant robot physiologies continuity in actuators can still not be enforced. The B-ORCA method [23], on the other hand, allows for arbitrary motion continuity, but requires an inflation of the robot shape similar to [21].

Thus, despite the large body of related contributions, we note that most of the extensions are limited towards a specific vehicle model or a specific order of solution continuity. In this article, by contrast, we derive a unifying theory for reciprocal collision avoidance based on the concept of *control obstacles*.

B. Contributions

This article develops three main contributions. First, we introduce the *Order n Continuous Control Obstacle* (C^n -CO), which describes the set of feasible feedback control trajectories that lead to a collision between interacting robots. The complement of this set forms the basis for safe motion planning. The method represents an extension to the *reciprocal Velocity Obstacle* (VO) collision avoidance method, such that trajectory segments verify C^n continuity rather than piecewise linearity. This property renders the presented method suitable as the local motion planning component for the large class of practically relevant robot physiologies admitting a linear or feedback-linearizable model representation.

Second, we establish that both the original VO method and several of its recently developed reciprocal extensions (which treat specific robot physiologies only) correspond to particular instances of C^n -CO. In addition to the described extension in trajectory continuity, C^n -CO thus represents a *unification* of existing reciprocal VO theory.

Third, we conduct an experimental evaluation of C^n -CO on a set of standard test scenarios. Additionally, we provide the results of a thorough parameter study that reveals under which environmental and control conditions the presented method with $n > 0$ is preferable over previous approaches that take the vehicle's kinematics into account via a radius enlargement of the original (zero-order) ORCA framework (i.e., [20, 21, 23]).

C. Limitations

While the C^n -CO method significantly extends the applicability of the relative velocity paradigm (as outlined above), it also inherits some of VO's limitations. In particular, C^n -CO requires agents to conform to a circular shape. For non-circular objects, collision avoidance is only guaranteed if the out-circle of their shape is selected instead. Furthermore, all agents are expected to employ the same collision avoidance strategy including *identical control parameters*. For heterogeneous groups of robots a combination between C^n -CO and the method presented in [23] may be employed instead.

Finally, this article describes a method for *local* navigation. As such, it requires a global motion planner to guarantee convergence among large static obstacles. One possible implementation is presented in Section VI resulting in an interactive partial motion planning framework.

D. Organization

The remainder of this article is structured as follows: in Section II we review the relative velocity paradigm that the Control Obstacle builds upon, extends and unifies. We then develop the C^n -CO method for *linear* robot models in *normal form* in Section III. In Section IV we describe the adaptations necessary to enable treatment of nonlinear systems via the well-known method of feedback linearization. Section V describes how C^n -CO may be seamlessly integrated into global graph search. The presented methods are then experimentally validated on a set of standardized test scenarios (Section VI). Finally, in Section VII we discuss the obtained results and conclude the article in Section VIII.

II. BACKGROUND

The development of the C^n -CO method builds directly on VO theory, and thus the relative velocity paradigm. In this section we outline the fundamentals of VO [24] and its reciprocal extension, ORCA [19]. We employ the formulation of [22] and set scalars in lower case italics (x), vectors in lower case bold ($\mathbf{x}$), and sets in uppercase italics (X).

A. Velocity Obstacle

The original Velocity Obstacle method [24] considers a single decision-making agent A (of radius r_A), located at position $\mathbf{p}_A$ with velocity $\mathbf{v}_A$, that moves among known objects B. Agent A's velocity obstacle induced by object B can then be introduced as a useful extension of the work-space obstacle concept.

Definition 1 (Velocity Obstacle). *The velocity obstacle VO_{AB}^τ induced for agent A by agent (or object) B for a time horizon τ is the set of all relative velocities $\mathbf{v}_{AB} = \mathbf{v}_A - \mathbf{v}_B$ of A with respect to B that will result in a collision between A and B before time τ .*

The VO method thus assumes that direct command of velocities is feasible, and further that other objects and agents continue moving along a linear extrapolation of their current motion. Hence it constitutes a predictive first-order method. Under these assumptions, the colliding configurations at time t are given by

$$\|\mathbf{p}_{AB,0} + \mathbf{v}_{AB}t\|_2 < r_{AB},$$

where $\mathbf{p}_{AB,0} = \mathbf{p}_A(0) - \mathbf{p}_B(0)$ and $r_{AB} = r_A + r_B$. The composite velocity obstacle VO_{AB}^τ is obtained as the union of velocities leading to collision

$$VO_{AB}^\tau = \bigcup_{0 < t \leq \tau} D\left(-\frac{\mathbf{p}_{AB,0}}{t}, \frac{r_{AB}}{t}\right),$$

where $D(\mathbf{p}, r)$ is the disk of radius r centered at $\mathbf{p}$ [22]. The VO of A induced by B is illustrated in Figure 1.

Collision avoidance is then accomplished as follows: agent A selects a new velocity $\mathbf{v}_{A,\infty} \notin VO_{AB}^\tau \oplus \mathbf{v}_B$ (with $\oplus$ the Minkowski sum operator) to remain safe for at least τ seconds under the stated assumptions. For local navigation $\mathbf{v}_{A,\infty}$ is usually selected so that it minimizes the Euclidean distance to

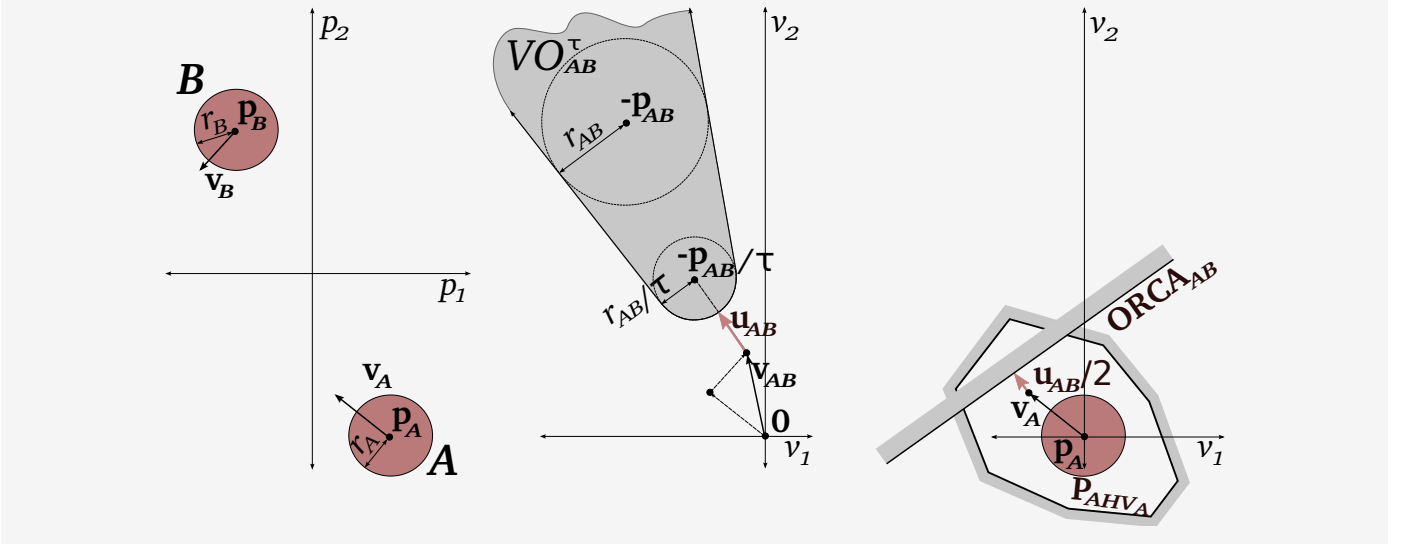


Fig. 1. The Velocity Obstacles (VO) and Optimal Reciprocal Velocity Obstacles (ORCA) methods. *Left*: Two agents A (located at $\mathbf{p}_A$) and B (at $\mathbf{p}_B$) interact using a VO-based method. In the VO formulation A assumes that B continues moving along its current velocity, hence $\mathbf{v}_{B,0} = \mathbf{v}_{B,\infty}$. In the ORCA approach A assumes that B also takes part in collision avoidance. *Center*: The velocity obstacle VO_{AB}^τ induced by B is composed of all relative velocity choices $\mathbf{v}_{AB,\infty} = \mathbf{v}_{A,\infty} - \mathbf{v}_{B,\infty}$ that lead to a collision before time τ . In the VO formulation A selects a new velocity $\mathbf{v}_{A,\infty}$ outside of $VO_{AB}^\tau \oplus \mathbf{v}_B$ to remain safe. In the ORCA approach a vector $\mathbf{u}_{AB}$ is constructed originating at $\mathbf{v}_{AB}$ and ending at the point on the VO boundary closest to $\mathbf{v}_{AB}$. *Right (ORCA approach only)*: each agent is subjected to one ORCA half-plane constraint for each interacting agent. Under the assumption that interaction proceeds fairly, $ORCA_{AB}$ is characterized by the point-direction pair $(\mathbf{v}_A + \frac{\mathbf{u}_{AB}}{2}, R(\frac{\pi}{2})\mathbf{u}_{AB})$, where $R(\cdot)$ denotes the 2D rotation matrix. In the case where robots are kinematically constrained, additional constraints that limit trajectory tracking error (such as P_{AHV_A} in the NH-ORCA approach [21]) may be specified. A then selects a velocity $\mathbf{v}_{A,\infty}$ that fulfills all constraints and minimizes the Euclidean distance to a preferred velocity $\mathbf{v}_A^{\text{pref}}$ linked to the agent's motion target.

$\mathbf{v}_A^{\text{pref}}$, which in turn points in direction of the next (sub-)goal extracted from a global waypoint-based plan. $\mathbf{v}_{A,\infty}$ is then applied for a short time period until the control loop begins anew. In practice the constant velocity assumption rarely holds but is addressed by having a fast re-planning loop.

B. Reciprocal Velocity Obstacle

In presence of decision-making agents, rather than passive dynamic objects, the assumptions underlying the VO approach become inaccurate and may lead to strong oscillations in the resulting trajectories. The recently introduced Reciprocal Velocity Obstacle (RVO) [18] and Optimal Reciprocal Collision Avoidance (ORCA) [19] methods eliminate these oscillations. They allow to compute a decentralized steady-state solution directly by *implicitly* requiring that the collision avoidance effort is shared between interacting agents. We refer to such methods as predictive-interactive first order approaches. The RVO method proposes a shifted formulation of that presented for the VO approach: by shifting the collision cone by $VO_{AB}^\tau \oplus \frac{\mathbf{v}_A + \mathbf{v}_B}{2}$ instead of $VO_{AB}^\tau \oplus \mathbf{v}_B$, interacting agents partake at least half in any resulting avoidance maneuver. Unfortunately the agents may chose to avoid each other on opposing sides, which may result in so-called reciprocal dances (not unlike to the ones observed in human crowds) [19]. ORCA, on the other hand, constructs a reciprocally collision-free velocity by solving an optimization instance involving linear constraints in velocity space, where each constraint is computed from the Velocity Obstacle induced by a particular neighboring agent. Additional constraints can be enforced to account for actuator limits [21, 23]. Figure 1 illustrates and describes the

methods. Due to the use of linear constraints, ORCA prevents oscillations altogether and ensures that contradictory decisions cannot be taken in absence of noise. We refer the reader to the literature for a more thorough treatment of this topic.

III. C^n -CO: HIGHER ORDER CONTINUOUS VELOCITY OBSTACLES FOR LINEAR SYSTEMS IN NORMAL FORM

A. Overview

We now proceed with developing the main contribution of this article. We construct the control obstacle of arbitrary continuity, C^n -CO, which represents the set of all colliding *relative* control sequences between two interacting agents that perform feedback control onto straight line reference trajectories (see Equation (1)). In contrast to other methods that have been developed for input or control obstacles [25, 26], our approach admits a formulation in relative velocity space that is suitable for *reciprocal* collision avoidance. In particular, $\mathbf{v}_{AB,\infty}$ appears linearly and thus separably in the resulting formulation, analogous to reciprocal VO. Our restriction to straight line reference segments is a consequence of this. Nonetheless, the transients locally ensure an expressive path set suitable for local motion planning. This property is exploited in Section V to integrate C^n -CO with global graph search.

In this section we develop the method for linear robot representations in *normal form* (i.e., a chain of integrators, see System 1), such as holonomic robots with acceleration control. In Section III-B we derive a state feedback controller that lets the agent converge onto straight line reference trajectories. By using this formulation the trajectories of two agents can be combined in relative velocity space from which a control

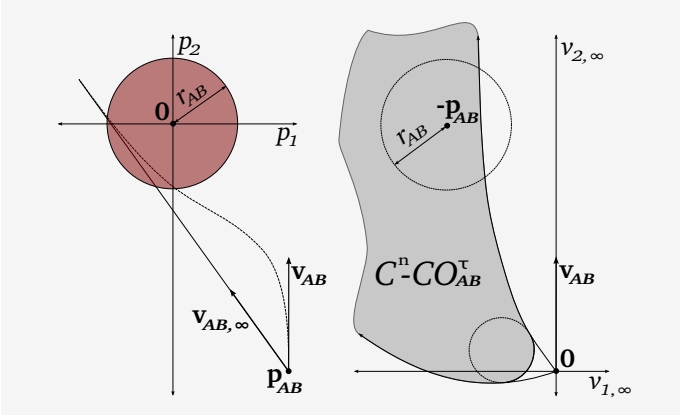


Fig. 2. The Control Obstacle with Continuity Constraints (C^n -CO). Left: Two agents A (located at $\mathbf{p}_A$) and B (at $\mathbf{p}_B$) move along continuous trajectories using the C^n -CO approach. Their relative position $\mathbf{p}_{AB}(t)$ can be expressed as a function of $\mathbf{v}_{AB,\infty}$. Right: The continuous control obstacle C^n-CO_{AB} induced for A by B is then composed of all relative velocities $\mathbf{v}_{AB,\infty} = \mathbf{v}_{A,\infty} - \mathbf{v}_{B,\infty}$ that lead to a collision before time τ .

obstacle may be computed. This obstacle specifies the set of control sequences leading to a collision (Section III-C). Finally, in Section III-D we describe how to transform the relative control obstacle into absolute coordinates while achieving the desired reciprocity in collision avoidance in absence of explicit communication.

B. Definitions

System 1 introduces the proposed agent description.

System 1 (Chain of Integrators).

$$\mathbf{p}^{(n+1)} = \mathbf{u}$$

Let $\mathbf{u} \in \mathbb{R}^m$ be piecewise continuous and $(\cdot)^{(n+1)}$ represent the derivative of order $n+1$ with respect to time. Then the resulting solution $\mathbf{p}(t)$ is C^n -differentiable. For $n=0$, a system representation directly compatible with the original ORCA approach is obtained. In the case where the input signal $\mathbf{u}(t)$ is such that System 1 is controlled onto Equation (1), we furthermore derive below that $\mathbf{p}(t)$ remains compatible with the relative velocity paradigm. This motivates the definition of a more general obstacle.

Definition 2 (C^n -Control Obstacle). *The Continuous Control Obstacle of order n induced for agent A by agent B for a time horizon τ ($C^n-CO_{AB}^\tau$) is the set of all relative desired velocities at $t \rightarrow \infty$: $\mathbf{v}_{AB,\infty} = \mathbf{v}_{A,\infty} - \mathbf{v}_{B,\infty}$ of A with respect to B that result in a collision before time τ if both agents are modeled according to System 1 and controlled via state feedback onto the piecewise linear trajectory given by Equation (1).*

C. Control trajectory for C^n -CO

The linear trajectory segments resulting from the original ORCA approach represent primitives for which reciprocal

collision avoidance for $t \rightarrow \infty$ can be specified. In the C^n -CO approach we retain these segments as reference trajectories

$$\mathbf{p}_{\text{ref}}(t) = \mathbf{p}_{\text{ref},0} + \mathbf{v}_\infty t, \quad t \in [0, \infty), \quad (1)$$

where $\mathbf{v}_\infty$ represents the ideal (or desired) velocity vector at $t \rightarrow \infty$. The kinematic model of System 1 with $n=0$ is able to precisely follow such reference motions $\mathbf{p}_{\text{ref}}(t)$. Conversely, in the case of $n > 0$ it cannot precisely track Equation (1) for almost all initial conditions. However, for $n > 0$ we may formulate a state feedback control law

$$\begin{aligned} \mathbf{p}^{(n+1)} &= \mathbf{u}(t) \\ \mathbf{u}(t) &= c_n(\mathbf{p}_{\text{ref}}^{(n)} - \mathbf{p}^{(n)}) + \dots + \\ &\quad c_0(\mathbf{p}_{\text{ref}} - \mathbf{p}), \end{aligned}$$

where c_i are control parameters and $\mathbf{p}_{\text{ref}}$ is specified according to Equation (1) with $\mathbf{p}_{\text{ref}}^{(k)} \equiv 0, \forall k > 1$. The control input can then be rewritten as

$$\begin{aligned} \mathbf{u}(t) &= -c_n \mathbf{p}^{(n)} - \dots - c_2 \ddot{\mathbf{p}} \\ &\quad + c_1(\mathbf{v}_\infty - \dot{\mathbf{p}}) + c_0(\mathbf{p}_{\text{ref},0} + \mathbf{v}_\infty t - \mathbf{p}). \end{aligned} \quad (2)$$

We will now show that the resulting transients onto Equation (1) also admit a reciprocal formulation: System 1 (controlled via the feedback law (2)) represents a non-homogeneous linear ordinary differential equation (ODE) whose solutions have been studied extensively [27]. Such solutions consist of a complementary $\mathbf{p}^c(t)$ and a particular $\mathbf{p}^p(t)$ part with $\mathbf{p}(t) = \mathbf{p}^c(t) + \mathbf{p}^p(t)$. The particular solution corresponds to the reference trajectory of Equation (1) and is independent of the order of System 1

$$\mathbf{p}^p(t) = \mathbf{p}_{\text{ref},0} + \mathbf{v}_\infty t, \quad t \in [0, \infty).$$

The complementary solution is obtained by solving the associated characteristic polynomial

$$\mathbf{q}^{n+1} + c_n \mathbf{q}^n + \dots + c_1 \mathbf{q} + c_0 \mathbf{1} = \mathbf{0}$$

and accounts for the added continuity requirements of the C^n -CO approach. For C^0 -CO (i.e., ORCA), the complementary solution vanishes. In the general case, for the control law to have a vanishing steady-state error, the boundary conditions $\mathbf{p}^c(0) = \lim_{t \rightarrow \infty} \mathbf{p}^c(t) = \mathbf{0}$ are imposed.

Example 1. *We illustrate the solution process for System 1 with continuity in velocity ($n=1$). Intuitively, System 1 with $n=1$ can be thought of as a point mass m attached to a moving reference position $\mathbf{p}_{\text{ref}}$ via a linear spring with spring constant k and a damper with viscous damping coefficient d . The associated characteristic polynomial is*

$$\begin{aligned} \mathbf{0} &= \mathbf{q}^2 + c_1 \mathbf{q} + c_0 \mathbf{1} \\ &= \mathbf{q}^2 + d/m \mathbf{q} + k/m \mathbf{1}. \end{aligned}$$

In this case the system's complementary solution admits three stable regimes, namely under-damped, over-damped and critically damped control. The resulting regime is determined through the control parameters c_i , or, equivalently, k and d . In absence of other constraints the critically damped solution ($d = 2\sqrt{mk}$) represents a natural choice as it provides the fastest non-oscillating decay to zero. After insertion of the

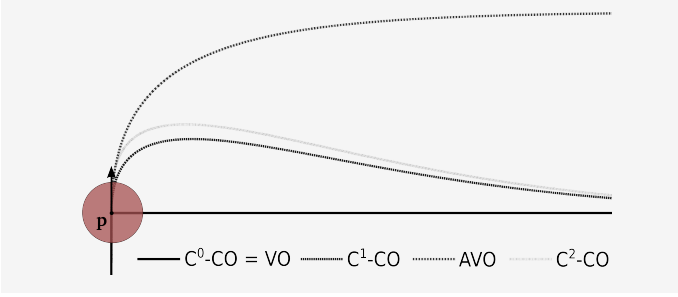


Fig. 3. Trajectory continuity across re-planning steps. Assume that robot A is located at $\mathbf{p}_{A,0} = \mathbf{p}$ and faces upwards with an initial velocity of $\mathbf{v}_{A,0} = 1 \text{ m/s}$, while a re-planning event due to a new goal located to the right is triggered. Using the ORCA = C^0 -CO approach, $\mathbf{v}_\infty$ is attained immediately. C^n -CO-controlled agents with $n > 0$ approach the new reference trajectory more slowly, and thereby ensure continuity of appropriate order. AVO [22] guarantees C^1 continuity but retains a steady-state tracking error to $\mathbf{p}_{\text{ref}}$.

boundary conditions and with $\lambda_0 = \sqrt{\frac{k}{m}}$, the complementary solution takes the form

$$\mathbf{p}^c(t) = e^{-t\lambda_0} \left[\mathbf{p}_0 - \mathbf{p}_{\text{ref},0} + t(\mathbf{v}_0 - \mathbf{v}_\infty + \lambda_0(\mathbf{p}_0 - \mathbf{p}_{\text{ref},0})) \right].$$

The combined solution for $n=1$ is then given by

$$\mathbf{p}^{C^1}(t) = \mathbf{p}^p(t) + \mathbf{p}^c(t). \quad (3)$$

We note that in Equation (3) the desired velocity $\mathbf{v}_\infty$ appears in the linear and additive form required by the relative velocity paradigm if the parameter λ_0 is selected identically for all agents. Example 1 thus demonstrates that there exist more complex than linear motions which are compatible with the reciprocal VO family of approaches. Similar results can be computed for the under- and over-damped cases. However, then two distinct eigenvalues are obtained.

The combined solution in Equation (3) also unbosoms the similarity to the AVO approach [22] which generates an input force proportional to the difference between an agent's instantaneous velocity $\mathbf{v}(t)$ and the reference velocity $\mathbf{v}_\infty$.

$$\mathbf{p}^{\text{AVO}}(t) = \mathbf{p}_0 + t\mathbf{v}_\infty + 1/\lambda_0(e^{-t\lambda_0} - 1)(\mathbf{v}_\infty - \mathbf{v}_0) \quad (4)$$

A comparison between Equations (3) and (4) affirms that AVO is indeed a special case of C^1 -CO with spring constant $c_0 = 0$. This choice is responsible for AVO's observed steady-state tracking error at $t \rightarrow \infty$, as depicted in Figure 3.

It can be furthermore shown that all solutions obtained by controlling System 1 of arbitrary order onto the reference trajectory (Equation (1)) via state-feedback are in accordance with the relative velocity paradigm. In particular the important case of $n=2$ may be derived as

$$\mathbf{p}^{C^2}(t) = \mathbf{p}^{C^1}(t) + F e^{-t\lambda_1} + ((\lambda_1 - \lambda_0)Ft - F) e^{-t\lambda_0}, \quad (5)$$

with

$$F = \frac{\mathbf{a}_0 + 2\lambda_0(\mathbf{v}_0 - \mathbf{v}_\infty) + \lambda_0^2(\mathbf{p}_0 - \mathbf{p}_{\text{ref},0})}{(\lambda_0 - \lambda_1)^2},$$

and λ_1 a design parameter related to the decay of the initial acceleration $\mathbf{a}_0$. Notice that $\mathbf{p}^{C^1}(t)$ forms an additive part of

the solution. The derivation is analogous to Example 1.

Figure 3 depicts $\mathbf{p}^{C^n}(t)$ for $n \in \{0, 1, 2\}$ and identical $\lambda_{(\cdot)}$. In our present implementation we select $\mathbf{p}_{\text{ref},0} = \mathbf{p}_0$ (i.e., the reference trajectory passes through the robot's current position), which substantially simplifies both Equation (3) and Equation (5) and allows for a simple way to infer other agents' reference position in absence of communication. In principle $\mathbf{p}_{\text{ref},0}$ could be used as an additional degree of freedom for the placement of reference trajectories, however. This would also allow to regenerate identical solution curves across re-planning steps.

D. Derivation of C^n -CO

Given the general solution of System 1, we now construct C^n -CO $_{AB}^\tau$ for two interacting agents A and B that each employ state feedback control onto their reference trajectory $\mathbf{p}_{\text{ref}}(t)$ following the procedure outlined in Section III-C. C^n -CO $_{AB}^\tau$ represents the set of relative reference velocities $\mathbf{v}_{AB,\infty}$ which result in a collision at time $t \leq \tau$. Thus C^n -CO $_{AB}^\tau = \{\mathbf{v}_{AB,\infty} \mid \|\mathbf{p}_{AB}(t)\|_2 \leq r_{AB}, t \leq \tau\}$ —see Figure 2. For notational simplicity we explain the solution process at the example of $n=1$.

Example 2 (Construction of C^1 -CO $_{AB}^\tau$). A collision between agents A and B occurs if their relative position becomes smaller than their combined radius, i.e., $\|\mathbf{p}_{AB}(t)\| \leq r_{AB}$. The agents' relative position follows from Equation (3) if λ_0 is selected identically for both of them:

$$\mathbf{p}_{AB}(t) = \mathbf{p}_{AB,0} + t\mathbf{v}_{AB,\infty} + e^{-t\lambda_0}t(\mathbf{v}_{AB,0} - \mathbf{v}_{AB,\infty})$$

The relative velocity $\mathbf{v}_{AB,\infty}$ may be separated by dividing both sides by $t(1 + e^{-t\lambda_0})$. We obtain

$$\|\mathbf{v}_{AB,\infty} + \frac{\mathbf{p}_{AB,0} + te^{-t\lambda_0}\mathbf{v}_{AB,0}}{t(1 + e^{-t\lambda_0})}\| \leq \frac{r_{AB}}{t(1 + e^{-t\lambda_0})}.$$

Finally, the continuous velocity obstacle C^1 -CO $_{AB}^\tau$ is given by the union of disks D centered at $\mathbf{p}_\odot(t)$ with radius $r_\odot(t)$.

$$C^1\text{-CO}_{AB}^\tau = \bigcup_{0 < t \leq \tau} D(\mathbf{p}_\odot(t), r_\odot(t))$$

$$D(\mathbf{p}_\odot(t), r_\odot(t)) = D\left(-\frac{\mathbf{p}_{AB,0} + te^{-t\lambda_0}\mathbf{v}_{AB,0}}{t(1 + e^{-t\lambda_0})}, \frac{r_{AB}}{t(1 + e^{-t\lambda_0})}\right)$$

Similar results hold for C^n -CO $_{AB}^\tau$ with higher-order continuity, as $\mathbf{v}_{AB,\infty}$ always appears linearly and thus separably in the solution of $\mathbf{p}_{AB}(t)$. The left and right boundary segments $\mathbf{b}^\pm(t)$ of C^n -CO $_{AB}^\tau$ can be developed in closed form. They are given by

$$\mathbf{b}^\pm(t) = \mathbf{p}_\odot(t) - \frac{r_\odot(t)}{\dot{r}_\odot(t)}\dot{\mathbf{p}}_\odot(t) + \mathbf{q}^\pm\left(\frac{r_\odot(t)}{\dot{r}_\odot(t)}\dot{\mathbf{p}}_\odot(t), r_\odot(t)\right),$$

with

$$\mathbf{q}^\pm(\mathbf{p}, r) = \begin{bmatrix} s & \mp r \\ \pm r & s \end{bmatrix} \frac{\mathbf{p}}{\|\mathbf{p}\|^2} \\ s = \sqrt{\|\mathbf{p}\|^2 - r^2}.$$

These Equations were first derived in [22] for AVO.

E. C^n -CO in Reciprocal Collision Avoidance

Similar to the ORCA algorithm we may require that each agent partially engages in collision avoidance based on a linear partitioning scheme to obtain reciprocity in navigation. But contrary to VO, the boundary segments of C^n -CO with $n > 0$ are curved, which would necessitate to consider its convex hull to properly designate the vector $\mathbf{u}_{AB}$ (introduced in Figure 1). If the convex hull was applied on the whole control obstacle, large parts of the feasible relative velocity space may be wasted. Fortunately, interacting agents are typically subject to input and state constraints. This allows us to restrict our attention to the reachable subset (consult Example 3) of the relative velocity space and apply the convex hull on that only. Let us thus develop the shape of the reachable set V_{AB} in relative velocity space before proceeding with the computation of fair partitions. V_{AB} amounts to the Minkowski sum of the reachable sets in absolute velocity space, namely $V_{AB} = V_A \oplus -V_B$. Their shape in turn depends on the particular constraints and agent model considered. Constraints arising from generic (non-) linear models are discussed in Section IV. For decoupled linear systems, however, state constraints act along each dimension of the physical space independently, resulting in simple reachable shapes for V_A and V_B . Let us illustrate this with an example.

Example 3 (Set of Reachable Velocities $V_{A,\infty}$ using C^1 -CO). From Equation (3) and its derivatives the time instances t^{ext} at which extremal velocities $\mathbf{v}^{\text{ext}}$ and accelerations $\mathbf{a}^{\text{ext}}$ occur along control trajectories are given by

$$\begin{aligned} \mathbf{v}^{\text{ext}} &= \mathbf{v}(2/\lambda_0) = \mathbf{v}_\infty - (\mathbf{v}_0 - \mathbf{v}_\infty)e^{-2} \\ \mathbf{a}^{\text{ext}} &= \mathbf{a}(3/\lambda_0) = (\mathbf{v}_0 - \mathbf{v}_\infty)e^{-3}\lambda_0. \end{aligned}$$

By observing the values both at these points and at the boundaries, the set of terminal velocities $\mathbf{v}_\infty$ compliant with velocity and acceleration constraints is obtained. For decoupled linear system models the resulting set is square. In order to guarantee equal responsiveness in all directions, we restrict $V_{AB,\infty}$ to its in-circle, however.

The procedure for $n > 1$ is identical to the one in Example 3, but the number of extremal points that need to be verified grows with n . In practice, it is thus often more appropriate to sample a discrete set of trajectories $\mathbf{p}(t)$ and selectively discard control sequences which lead to a violation of a system constraint. By doing so, arbitrary constraints may be considered.

After $V_{AB,\infty}$ (i.e., the set of reachable relative velocities) has been computed, a collision-free velocity for each agent can be obtained analogously to [22], as depicted in Figure 4: a half-plane H_{AB} is placed tangentially onto the convex hull of $V_{AB,\infty} \cap C^n\text{-CO}_{AB}^\tau$, so that the minimal *signed* distance to $\mathbf{v}_{AB}$ is maximized. The vector connecting the convex Hull with $\mathbf{v}_{AB}$ is again denoted by $\mathbf{u}_{AB}$. In each interacting agent A's absolute velocity space, linear half-plane constraints are then applied at $\mathbf{v}_A + \frac{\mathbf{u}_{AB}}{2}$, in direction of $R(\frac{\pi}{2})\mathbf{u}_{AB}$, where $R(\cdot)$ denotes the 2D rotation matrix. For the duration of a control cycle, each agent then selects the feasible velocity $\mathbf{v}_{(\cdot),\infty}$ that minimizes the Euclidean distance to its preferred velocity.

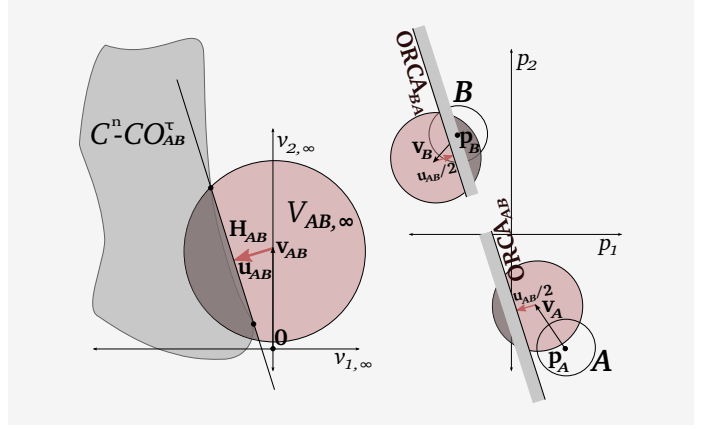


Fig. 4. Reciprocal collision avoidance using C^n -CO (described for agent A). *Left:* The set of reachable relative velocities $V_{AB,\infty}$ is colored in red, $C^n\text{-CO}_{AB}^\tau$ in light gray, and the convex hull of their intersection $V_{AB,\infty} \cap C^n\text{-CO}_{AB}^\tau$ in dark gray. $\mathbf{u}_{AB}$ is computed by intersecting $V_{AB,\infty}$ with a half-plane H_{AB} . The half-plane H_{AB} is constructed via tangential placement onto the convex hull of $V_{AB,\infty} \cap C^n\text{-CO}_{AB}^\tau$ through the point on the convex hull that maximizes the *signed* minimal distance $d_{\min}$ between $\mathbf{v}_{AB}$ and H_{AB} . $\mathbf{u}_{AB}$ is a vector perpendicular to H_{AB} with $\|\mathbf{u}_{AB}\|_2 = d_{\min}$. *Right:* $\mathbf{u}_{AB}$ is used to construct linear half-plane constraints in each agent's absolute velocity space analogous to the ORCA approach: under the assumption that interaction proceeds *fairly*, ORCA_{AB} is characterized by the point-direction pair $(\mathbf{v}_A + \frac{\mathbf{u}_{AB}}{2}, R(\frac{\pi}{2})\mathbf{u}_{AB})$, where $R(\cdot)$ denotes the 2D rotation matrix. A then selects the velocity $\mathbf{v}_{A,\infty}$ in accordance with all constraints, that minimizes the Euclidean distance to its preferred velocity $\mathbf{v}_A^{\text{pref}}$. This procedure is repeated at control frequency.

Let us conclude this section with the following remarks: the C^n -CO approach allows to apply the reciprocal VO concept *in closed form* to the class of robot models equivalent to System 1. And while such models rarely depend on solution continuity beyond C^1 , more general (non-)linear systems often do. In Section IV we extend C^n -CO's applicability to a large class of (non-)linear robotic systems which can be brought into an equivalent (higher-order) representation in normal form via feedback linearization. This ensures the practical relevance of C^n -CO with $n > 1$.

IV. C^n -CO APPLIED TO FEEDBACK LINEARIZABLE SYSTEMS

In the previous section we described a reciprocal collision avoidance framework for linear integrators of arbitrary order and thus continuity. In this section the framework is extended via feedback linearization to a broader range of both linear and non-linear agent models. The procedure is as follows: if found to be conformant, the system under consideration is first feedback linearized (Section IV-A). Second, a collision-free reference trajectory of sufficient continuity (Sections IV-C & IV-D) is computed following the method of Section III for the equivalent system in linear normal form. Third, input and state constraints stemming from the non-linear system may be added to the optimization problem as described in Section IV-E. The ensuing optimization results in the selection of $\mathbf{v}_\infty$. Finally, the non-linear system is controlled onto the selected high-order reference trajectory given by Equation (1) using the input transformation obtained from the feedback linearization step.

A. Feedback Linearization

Feedback linearization is a well-known control technique which allows to modify a controlled system's inputs so that its input-output (and in some cases input-state) behavior assumes the linear normal form of System 1. Such systems can thus be controlled via well-understood linear control theory. Conversely, the C^n -CO framework—which we developed for linear systems above—can be naturally extended to feedback linearizable systems. The technique is applicable to systems of the form

System 2 (General Feedback Linearizable System).

$$\begin{aligned}\dot{\mathbf{x}} &= f(\mathbf{x}) + g(\mathbf{x})\mathbf{v} \\ \mathbf{y} &= h(\mathbf{x})\end{aligned}$$

with $\mathbf{x}$ the state vector and $\mathbf{y}$ the output of the system. The aim is then to derive an appropriate control input transformation

$$\mathbf{v} = a(\mathbf{x}) + b(\mathbf{x})\mathbf{u},$$

that creates a linear mapping between $\mathbf{u}$ and $\mathbf{y}$ [28]. We distinguish between static and dynamic feedback linearization. Static feedback linearization applies to systems that are fully controllable about their reference point already. Examples include generic linear systems. In that case, the resulting equivalent system in normal form is of the same order as the original system. If, on the other hand, the reference point is not fully controllable (such as $\mathbf{p}$ in Systems 4 & 5), the state-space needs to be augmented with dynamic compensators, resulting in a higher-order equivalent system in normal form [29].

It turns out that many robotic system models of practical relevance are feedback linearizable but we will in this article restrict our attention to general linear systems, kinematic unicycles and bicycle models for two main reasons. On one hand, they are interesting in view of their practical relevance—omni-drives, diff-drive robots, and cars may all be modeled this way. On the other hand, this choice allows us to show that C^n -CO subsumes both ORCA-DD [20], developed for unicycles, and LQR-obstacles [30, 31], which has (independently from our work) been developed for general linear systems.

B. Selected Agent Models

The most general class of time-invariant linear systems may be expressed as follows.

System 3 (General Linear System).

$$\dot{\mathbf{x}} = F\mathbf{x} + G\mathbf{v},$$

where F is an $(n+1) \times (n+1)$ matrix and G is of size $(n+1) \times m$, with m the number of inputs.

Kinematic models of unicycles and bicycles, on the other hand, belong to the class of non-holonomic, and thus non-linear systems. Such systems have in common that they are subject to at least one non-holonomic constraint. The kinematic model of a unicycle is given by System 4

System 4 (Unicycle).

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{p}} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} \cos \delta \\ \sin \delta \\ 0 \end{bmatrix} v_1 + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} v_2,$$

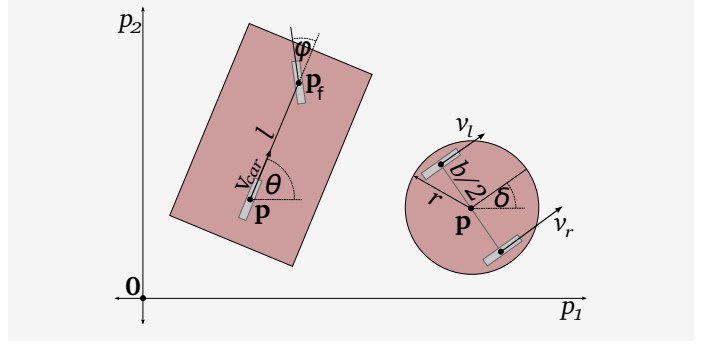


Fig. 5. Non-holonomic system models. *Left*: Bicycle model of a car. It is steered via throttle (v_{car}) and steering wheel velocity ($\dot{\phi}$). The reference point is usually selected at the center of the rear-wheel axle, but another common choice includes the center of the front-wheel axle ($\mathbf{p}_f$). In the latter case v_{car} is applied at the front wheels as well. *Right*: Unicycle model of a differential-drive robot that is steered via left and right wheel motors. Forward and rotational velocity result in $v_{\text{dd}} = (v_r + v_l)/2$, $\omega_{\text{dd}} = (v_r - v_l)/b$.

with $\mathbf{v} = (v_1, v_2)^T$, the forward and rotational speed of the platform, respectively. The relevant variables are introduced in Figure 5.

In contrast to the unicycle, bicycles are commonly commanded via the throttle $v_1 = \|\dot{\mathbf{p}}\|$ and the front wheel's steering velocity $v_2 = \dot{\phi}$. The kinematic model of a bicycle with rear-wheel drive is represented by

System 5 (Rear-wheel driven Bicycle).

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{p}} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos \theta \\ \sin \theta \\ \tan \phi/l \\ 0 \end{bmatrix} v_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v_2.$$

A structural similarity between Systems 4 and 5 becomes apparent by moving the bicycle's reference point to the front axle center [29]. The resulting system, a bicycle with front-wheel drive, may be thought of as a unicycle with a trailer attached [29], hence $\delta = \theta + \phi$. The derivation of the above robot models are described in [29, 6].

C. Linearization via Static State Feedback

We noted that static feedback linearization is applicable to systems whose reference point is fully controllable. To this end, let us first consider general linear systems (characterized by System 3). Such models are said to be fully controllable iff $[F, FG, \dots, F^{n+1}G]$ has full rank. Under the assumption of full controllability, System 3 then admits feedback linearization into System 1 thanks to its conformance with System 2. Interestingly, the LQR Obstacle method [30] steers precisely such general linear systems onto straight line reference trajectories using state feedback control. Hence, C^n -CO completely subsumes the LQR Obstacle method on the basis of System 2 and feedback linearization.

We now proceed with the non-linear systems introduced in Section IV-B. For Systems 4 and 5 the reference point $\mathbf{p}$ is not fully controllable. Following [29], any reference point $\hat{\mathbf{p}}$ to the front of the vehicle axles is fully controllable, however, and linearization via static feedback becomes feasible. For

symmetry reasons, the new reference point $\hat{\mathbf{p}}$ is usually chosen as follows (depicted in Figure 6):

$$\hat{\mathbf{p}} = \mathbf{p} + [\cos \delta, \sin \delta]^T \Delta,$$

where $\Delta > 0$ defines the maneuverability of the vehicle. For the robot of System 4 the following relations hold

$$\begin{aligned} \dot{\hat{\mathbf{p}}} &= \mathbf{u} \\ v_1 &= \mathbf{u} \cdot [\cos \delta, \sin \delta]^T \\ v_2 &= \mathbf{u} \cdot [-\sin \delta, \cos \delta]^T / \Delta. \end{aligned}$$

System 4 is then controllable about $\hat{\mathbf{p}}$ with holonomic velocity inputs given by $\mathbf{u}$ and C^n -CO may be applied at this point. Due to the internal dynamics of the new system and the circular shape requirement of VO-based methods, an increased radius given by $R = r + \Delta$ is required. This leads to the situation depicted in Figure 6–*Left* and for C^0 -CO is equivalent to the ORCA-DD approach presented in [20]. Note that the parameter Δ is variable so that it may be decreased in dense situations—at the expense of lateral maneuverability.

As established in Section IV-B, a front-wheel driven bicycle is mathematically equivalent to a unicycle (System 4) pulling a trailer with relative orientation φ . Analogue to the case above, such a System is then controllable about $\hat{\mathbf{p}}$ and C^n -CO can be applied at this point. Nevertheless, in this case, the increased radius R centered at $\hat{\mathbf{p}}$ may become prohibitively large as depicted in Figure 6–*Right*. Note that this method likewise extends to vehicles with trailers—at the cost of the corresponding enlargement. By contrast, dynamic feedback linearization does not require artificial inflation of the vehicle radius, as even partially controllable reference points $\mathbf{p}$ can be employed in that case.

D. Linearization via Dynamic State Feedback

Dynamic feedback linearization overcomes the non-controllability of $\mathbf{p}$ via so-called dynamic extension. Here we apply it again to Systems 4 & 5. The degree of the extension is dependent on the number of states of the original system.

For the kinematic unicycle, its three states need to be appended by a single dynamic compensator only. The resulting chain of integrators is of second order, as derived in [29], and hence C^1 -CO applies:

$$\begin{aligned} \ddot{\mathbf{p}} &= \mathbf{u} \\ v_1 &= \|\dot{\mathbf{p}}\| \\ v_2 &= v_1 \frac{\dot{p}_1 \ddot{p}_2 - \dot{p}_2 \ddot{p}_1}{(\dot{p}_1^2 + \dot{p}_2^2)^{3/2}}. \end{aligned}$$

After optimization, the inputs to the original system $(v_1, v_2)^T$ can be deduced from the holonomic solution trajectory (and its derivatives). Recalling System 4, this implies $\delta \in C^0$ and $\dot{\mathbf{p}} \in C^0$. If continuity in the actuators and in the curvature of the unicycle's trajectory is desired (kino-dynamic case), C^2 -CO becomes necessary.

Dynamic feedback linearization thus allows to retain the vehicle reference point $\mathbf{p}$ and avoids artificial inflation of the robot radius r that the shifting of $\mathbf{p}$ would entail. This benefit comes at the price of a higher-order model.

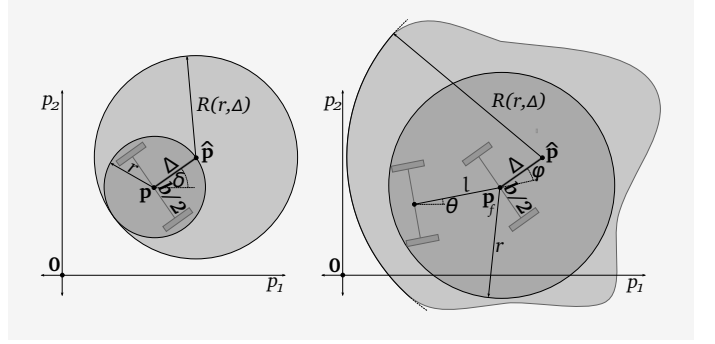


Fig. 6. Consequence of feedback linearization on the effective robot radius. Dynamic feedback linearization allows to retain the reference point $\mathbf{p}$ (and $\mathbf{p}_f$, respectively) allowing to formulate C^n -CO with the true robot radius r (i.e., the out-circle of the robot, colored in dark gray). Static feedback linearization necessitates a redefined reference point $\hat{\mathbf{p}}$ which increases the effective radius to $R(r, \Delta)$, colored in light gray. *Left*: Unicycle. *Right*: Bicycle.

For the rear-wheel driven bicycle (System 5), the same procedure results in the following chain of integrators and input transformations [29]:

$$\begin{aligned} \ddot{\mathbf{p}} &= \mathbf{u} \\ v_1 &= \|\dot{\mathbf{p}}\| \\ v_2 &= l v_1 \frac{(\ddot{p}_2 \dot{p}_1 - \dot{p}_2 \ddot{p}_1) v_1 - 3(\ddot{p}_2 \dot{p}_1 - \dot{p}_2 \ddot{p}_1)(\dot{p}_1 \dot{p}_1 + \dot{p}_2 \dot{p}_2)}{v_1^6 + l^2 (\ddot{p}_2 \dot{p}_1 - \dot{p}_2 \ddot{p}_1)^2}. \end{aligned}$$

This result can be motivated as follows: for the kinematic version of the bicycle the linear and steering velocities must at least verify $v_1, v_2 \in C^{-1}$. Recalling System 5 this implies $\varphi \in C^0$, $\theta \in C^1$ and $\dot{\mathbf{p}} \in C^1$, which necessitates C^2 -CO. If the continuity requirements are increased to continuity in steering rate $\varphi \in C^1$ (kino-dynamic model), C^3 -CO is needed.

We note that the above formulations do not inherently respect system constraints. Handling of actuator and state constraints is described next.

E. Nonlinear Constraint Satisfaction

In Section III-E we established that for decoupled linear systems the set of attainable velocities $V_{AB, \infty}(\mathbf{v}_{AB, 0})$ is a square (or, more conveniently, its in-circle) centered at $\mathbf{p}_0$. For feedback-linearized systems the situation is more complex, as the actuator and state constraints of the underlying system may impose non-linear requirements on the attainable shapes of the solution trajectory $\mathbf{p}(t)$. While in principle the individual sets of constraint-compliant velocities $V_{A, \infty}$ can be computed using agent A's system model, in general the transformed constraints do not admit a closed form solution. Instead, we may (exhaustively) sample from $V_{A, \infty}$ and for each resulting trajectory numerically verify conformance with non-linear system constraints. This operation may be performed offline and stored in a *constraint mask* representing the set of reachable velocities (for discretized initial conditions), as proposed in [23].

F. Reciprocal Collision Avoidance

Reciprocal collision avoidance for feedback linearized systems proceeds similarly to Section III-E, except that the set

$V_{AB,\infty}$ of reachable relative velocities at $t \rightarrow \infty$ is obtained by taking the Minkowski sum of the individual agents' constraint masks computed above. Consequently, the set $V_{AB,\infty}$ will in general no longer be convex. By selecting its *out-circle* $\tilde{V}_{AB,\infty}$ for the purpose of constructing the half-plane H_{AB} only, we ensure that feasible actions are never inhibited. On the other hand, we need to ensure that unreachable velocities will later be discarded again. The half-plane constraint H_{AB} is then computed tangent to the convex hull of $\tilde{V}_{AB,\infty} \cap C^n\text{-CO}_{AB}^\tau$ so that its signed minimal distance $d_{\min}$ to $\mathbf{v}_{AB}$ is maximized. The vector $\mathbf{u}_{AB}$ is formed perpendicular to H_{AB} with $\|\mathbf{u}_{AB}\|_2 = d_{\min}$ and is used to construct ORCA half-plane constraints in each agent's absolute velocity space. Finally, each agent A selects the collision-free velocity within $V_{A,\infty}$ that is closest to its preferred velocity $\mathbf{v}_A^{\text{pref}}$. Due to the potential non-convexity of the set V_A , this optimization may be efficiently solved by a wavefront propagation algorithm that proceeds from $\mathbf{v}_A^{\text{pref}}$ outwards [23, 32].

V. C^n -CO EMPLOYED IN PARTIAL MOTION PLANNING

The practice of employing the Euclidean distance to $\mathbf{v}^{\text{pref}}$ as optimization criterion has rendered relative velocity-based methods inherently local—and thus prone to local minima in the vicinity of concave static obstacles. C^n -CO's direct incorporation of an agent model, on the other hand, facilitates the adoption of a global optimization criterion. The key insight lies in the existence of a bijection between elements of $V_{(\cdot),\infty}$ and trajectory segments $\mathbf{p}_{(\cdot)}(t)$: a sampling in $V_{(\cdot),\infty}$ forms a highly expressive trajectory set locally, that approaches the various straight line reference velocity vectors at $t \rightarrow \infty$ (see Figure 3). By considering these system-compliant segments up to a time horizon τ only (linked to the control parameter $\lambda_{(\cdot)}$), a tree of depth one is obtained. This tree may be searched and extended online using a graph search method compatible with the system models underlying C^n -CO, such as ARAE* [10] or RRT* [33]. During search, sampled edges at depth one that are found to be in violation of fair interaction principles (i.e., segments that do not conform with the imposed linear constraints) are discarded. The same applies to segments (at any search depth) in collision with static obstacles. By running these components in a loop at control frequency, an interactive model-predictive partial motion planning framework is obtained. This is demonstrated in Section VI-C.

VI. EXPERIMENTS

The experimental section is divided into four parts. In the first part we provide experimental evidence supporting the adequacy of key design choices made during the development of Section III. In particular we quantitatively specify the advantage of modeling a curved higher-order collision cone (in accordance with robot dynamics) over the much simpler instance of the original VO approach—even if a linear half-plane constraint is used to partition sets of fair velocities for performance reasons. Tied to this we provide evidence that in absence of (inefficient) centralized optimization, it seems difficult to obtain significantly better results by using a partition scheme different from said fast linear approach. In

the second part we perform simulated multi-agent navigation experiments. Specifically, we assess C^1 -CO against inflated C^0 -CO (representing a generic controller tracking ORCA) for a large set of environmental and agent conditions. In part three we demonstrate that C^n -CO can be seamlessly integrated into a global navigation framework. For parts two and three we have included a supplementary color MPEG-4 file that visualizes typical sample runs. This material is available at <http://ieeexplore.ieee.org>. Finally, in part four the runtime performance and scalability of the presented methods is analyzed.

A. Assessment of Design Choices

The exposition of this article is largely based on the assumption that C^n -CO's collision cone with $n > 0$ is at most as large as the corresponding omni-directionally inflated C^0 -CO. This is quantitatively verified via a comparison between the C^1 -CO method and an ε -inflated C^0 -CO method tracked by a generic controller, where ε accounts for tracking errors. The latter method essentially corresponds to NH-ORCA [21]. However, for better comparability we substitute its controller with the one employed in C^1 -CO.

We furthermore observed that C^n -CO's curved shape complicates the construction of a fair partition scheme. In Section III we opted for a well-known linear partition for computational performance reasons. The potential loss in solution quality stemming from this decision with respect to a non-linear partition scheme thus also requires analysis. The size of the obtained partition $|V|$, or the minimal distance between $\mathbf{v}_{AB,0}$ and the boundary of the partition (i.e., $d_{\min}$) serve as performance criteria.

1) *Linear vs. Non-Linear Constraints:* Let us recall from Figure 4 that the C^n -CO method selects a half-plane H_{AB} such that the *signed* shortest distance $d_{\min}$ between the convex hull of $V_{AB,\infty} \cap C^n\text{-CO}_{AB}^\tau$ and $\mathbf{v}_{AB,0}$ is maximized. This is equivalent to the maximization of the radius $r_{\min}$ of a disk centered at $\mathbf{v}_{AB,0}$. Hence we observe that our method favors allotment of fair velocities close to the current absolute velocities over maximizing the overall size of the set. For higher-order models and the associated—characteristically non-convex—collision cones, partitions more efficient than those arising from linear constraints could be constructed. However, many such (e.g., deconvolution-based) methods are not well posed inside a distributed framework and thus require centralized coordination. By retaining the same optimization criteria as for the ORCA algorithm, for comparison purposes a (computationally expensive) decentralized non-linear iterative wavefront expansion method can be designed to partition curved collision cones instead. This method incrementally expands a wavefront from $\mathbf{v}_A$ and $\mathbf{v}_B$ simultaneously. The covered velocities are then appropriated to the sets of fair velocities $\hat{V}_{A,\infty}$ and $\hat{V}_{B,\infty}$, respectively. Expansion in a particular direction is interrupted if the Minkowski sum of the two growing sets $\hat{V}_{AB,\infty} = \hat{V}_{A,\infty} \oplus -\hat{V}_{B,\infty}$ intersects the nonlinear collision cone $C^n\text{-CO}_{AB}^\tau$. The method terminates once the boundary cannot be expanded any further. In practice, we implemented this approach via a discrete scheme using

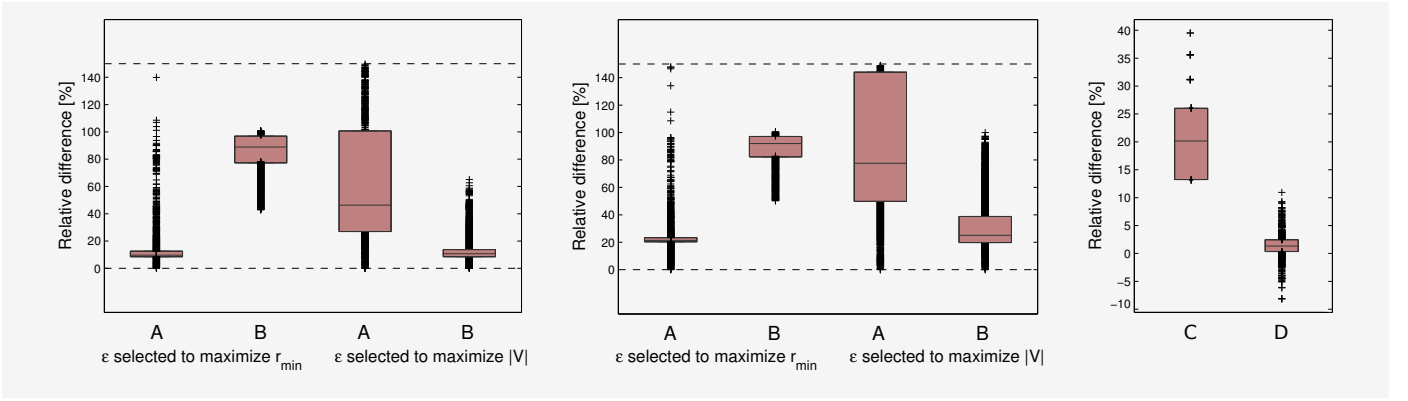


Fig. 7. Assessment of design choices. *Left*: Comparison of $r_{\min}$ and $|\hat{V}_{AB,\infty}|$ between C^1 -CO and ϵC^0 -CO for control parameters $\lambda_0 = 0.50 \frac{1}{s}$. Displayed are the relative values A: $\frac{r_{\min}^{C^1\text{-CO}} - r_{\min}^{\epsilon C^0\text{-CO}}}{r_{\min}^{C^1\text{-CO}}}$ and B: $\frac{|V|_{\min}^{C^1\text{-CO}} - |V|_{\min}^{\epsilon C^0\text{-CO}}}{|V|_{\min}^{C^1\text{-CO}}}$, respectively. *Center*: Same scenario for $\lambda_0 = 0.25 \frac{1}{s}$. *Right*: Comparison between the C^1 -CO method performing a wavefront-based nonlinear partition of $\hat{V}_{AB,\infty}$ and the C^1 -CO method based on a linear partition. Control parameters are selected as $\lambda_0 = 0.25 \frac{1}{s}$. Displayed are the relative values C: $\frac{r_{\min}^{\text{wave}} - r_{\min}^{\text{ORCA}}}{r_{\min}^{\text{wave}}}$ and D: $\frac{|V|_{\min}^{\text{wave}} - |V|_{\min}^{\text{ORCA}}}{|V|_{\min}^{\text{wave}}}$. We observe that the nonlinear method's improvement in minimal radius $r_{\min}$ is insignificant (i.e. below the $0.05 m$ discretization) in almost all cases (27490 out of 28802). Displayed are the significant cases only.

polar coordinates centered around the agents' current velocities with $0.05 \frac{m}{s}$ radial and $\pi/50 \frac{rad}{s}$ tangential discretization.

The experimental setup consists of two identical circular interacting agents A and B of radius $1.0 m$. Agent A is placed at the origin. Successively a discrete set of speed levels $s_A = \{-0.5 : 0.25 : 0.5\} \frac{m}{s}$ is applied in direction $\theta_A = \{\pi/4 : \pi/4 : \pi\}$. Agent B is successively placed on a rectangular grid with coordinates $\mathbf{p}_B = \{(2.5 : 0.5 : 5, 2.5 : 0.5 : 5)\} m$ and subjected to velocities from the same set as that of agent A. For both agents the eigenvalues of the closed loop system are either placed at $\lambda_{01} = 0.5 \frac{1}{s}$ or at $\lambda_{02} = 0.25 \frac{1}{s}$. These choices correspond to maximal acceleration constraints $\mathbf{a}_{\max_1} = 1.0 \frac{m}{s^2}$ and $\mathbf{a}_{\max_2} = 0.5 \frac{m}{s^2}$, respectively, occurring for 90° turns at $\mathbf{v}_{\max} = 1 \frac{m}{s}$. The $\mathbf{a}_{\max}$ -values roughly represent upper limits for road-borne vehicles on dry and wet roads, respectively, after which tires start to slip. We draw statistics over the whole set of configurations.

As expected, the more involved wavefront expansion generally produces fair sets of larger $r_{\min}$ —the largest differences are observed in cases where the collision cone is strongly curved. But in the overall context, the gains are mostly insignificant. In fact, only 1312 out of 28802 samples resulted in an increase in $r_{\min}$ larger than the discretization employed ($0.05 \frac{m}{s}$). Figure 7–*Right* visualizes the distribution of the improvement of $r_{\min}$ over the 1312 significant experiments. In the light of these findings the choice favoring the much faster linear constraints is justified.

2) *Partition size of C^1 -CO vs. ϵC^0 -CO*: Prior to this work, most reciprocal VO-based methods required an inflation of the effective agent radii to compensate for the systems' inability to follow C^0 -continuous trajectories. Hence we assess the advantage of employing C^n -CO (subject to linear constraints, as justified above) for higher-order robot models directly over the simpler specification of an ϵ -inflated linear collision cone.

Let us consider the same experimental setup as in Section VI-A1 composed of two agents A and B. We examine two *maximization* criteria. The first one pertains to the minimal

radius of a disk centered around $\mathbf{v}_{AB,0}$ (as in the ORCA method). The second one is complementary and concerns the size of the fair velocity set $|\hat{V}_{AB,\infty}|$.

For the inflated C^0 -CO approach, the optimal result represents a trade-off between increasing ϵ (which increases the size of the set of trackable velocities) and decreasing inflation (which represses the half-plane boundary constraint). While in theory the inflation value ϵ may be adaptively selected, practical aspects further constrain this choice. In fact, pairwise interactions between multiple agents may produce differing optimal values (according to one of the mentioned criteria). Nonetheless, in our comparison limited to a single interaction, we densely sample over ϵ values and select the one which maximizes the appropriate metric. The results should thus be interpreted as a best possible outcome for the ϵC^0 -CO approach, which may not fully materialize in multi-agent navigation. As Figure 7–*Left* and –*Center* show for $\lambda_0 \in \{0.25, 0.5\} \frac{1}{s}$, however, the higher-order solution is even under these conditions significantly preferable, producing both larger $r_{\min}$ and $|\hat{V}_{AB,\infty}|$ at the same time. On the other hand, the results also suggest that the performance of both approaches converges for agents admitting even faster dynamics, i.e., $\lambda_0 > 0.5 \frac{1}{s}$. This corresponds to agents for which a first-order model is approximately applicable.

Together, the above two results justify our proposed approach in general, and the choice for linear constraints fitted onto the non-linear collision cone in particular.

B. Assessment of Local Multi-Agent Navigation Performance

While the above test cases indicate that an agent employing C^1 -CO is generally allowed more avoidance options than one using inflated C^0 -CO, these theoretical benefits do not necessarily need to materialize in typical planning scenarios. To this end, we compare the C^1 -CO approach versus an inflated C^0 -CO approach on a series of test cases.

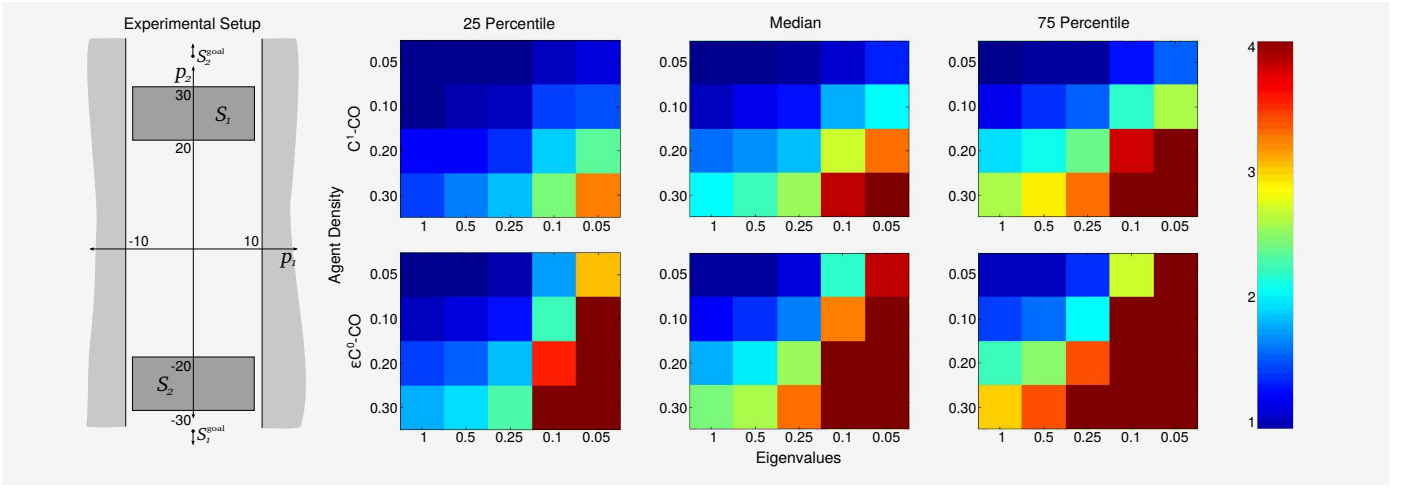


Fig. 8. Local multi-agent navigation. *Left*: Experimental setup consisting of two sets of agents S_1 and S_2 that operate within a bounded scene, as described in Section VI-B2. *Right*: Displayed are the 25, 50, and 75 percentile convergence times for said randomized experiments. For each parameter combination, 20 experiments have been performed. *Right-Top Row*: C^1 -CO. *Right-Bottom Row*: ϵC^0 -CO. The results demonstrate that the advantage of employing a higher-order C^n -CO over inflated C^0 -CO materializes stronger for slow agent dynamics. Conversely, for fast agent dynamics, i.e., $\lambda_0 \in [0.5, \infty) \frac{1}{s}$, the performance of both approaches converges. The same conclusions apply for densely vs. sparsely populated scenarios. A color video of this experiment is available at <http://ieeexplore.ieee.org>.

1) *Local Navigation in a Circle Swap*: Let us consider a circular setup, composed out of eight agents employing C^1 -CO with $\lambda_0 = 0.5 \frac{1}{s}$. Agents need to exchange their position diametrically. The circle swap represents a hard scenario for decentralized local navigation approaches. In contrast to ordinary ORCA, local minima (occurring in the cluttered center) are much more difficult to resolve (via e.g., random walks) due to the constrained kinematics of the platforms considered. Conversely, once agents agree on an avoidance *topology*, the physical constraints preserve stability. By employing a scheme that prefers avoidance to one side (similar to humans), efficient patterns reminiscent of roundabouts emerge—see Figure 9.

2) *Local Navigation in Counterflow*: Next, let us consider a scenario composed of two sets of agents. The two sets are bounded by rectangles of $20m \times 10m$ size, and are separated by $40m$ (displayed in Figure 8–Left). Agents are circular with $r = 1m$ and obey $v_{\max} = 1.0m/s$. Sensor limits are modeled as uniformly distributed noise $\sigma \in [-0.05, 0.05]m$ on position estimates. Goal locations are placed in direction of the opposing set of agents at $\pm\infty m$. For each agent the time is elapsed between its start position and $50m$ towards the goal. Consequently, the first encounters between agents of different sets on average occur after $25m$. Simulations are aborted after $4t_{\min}$, where $t_{\min}$ is the minimal time required to traverse the scene in absence of other agents. Experiments are performed over increasingly more densely populated scenes (between 5% and 30% of the rectangular areas covered by agents), and decreasing agent dynamic capabilities $\lambda_0 \in \{0.05, 0.1, 0.25, 0.5, 1.0\} \frac{1}{s}$. For each parameter combination, statistics are drawn over 20 simulations. Figure 8 displays the 25, 50, and 75 percentile convergence times for the described randomized experiments. Note that deadlocks usually resolve spontaneously due to perception noise. The results confirm our earlier finding that the advantage of employing a higher-order C^n -CO over inflated C^0 -CO

appears strongest for slow agent dynamics. Conversely, for fast agent dynamics, $\lambda_0 \in (0.5, \infty] \frac{1}{s}$, the performance of both approaches converges. The same conclusions apply for densely versus sparsely populated scenarios.

C. Demonstration of C^n -CO in Partial Motion Planning

Natural environments usually consist of static obstacles in addition to dynamic agents. These obstacles may attain arbitrary shapes and are often concave, which renders VO-based methods in their basic form prone to local minima. In Section V we have proposed a framework which allows for the seamless integration of C^n -CO into global graph search. Figure 10 demonstrates this framework in simulation. It visualizes snapshots at several time instances along a scenario, where two agents need to exchange their position within a maze. Global convergence to their respective goals is obtained despite the presence of concave static obstacles. Both agents are modeled according to C^1 -CO with $\lambda_0 = 0.5 \frac{1}{s}$. The time horizon after which interaction effects are neglected has been selected at $\tau = 4s$ (corresponding to t^{ext} , see Example 3). v_{∞} is sampled exhaustively using a discretization of $0.1m/s$ and $v_{\max} = 1.0m/s$. This results in a tree composed of at most 441 segments at depth one. Tree segments in violation with linear constraints stemming from interaction or in collision with the static map are discarded immediately. The remaining segments are penalized according to the cells they traverse. Graph search terminates at a tree depth of one, at which point it connects to a field Dijkstra heuristic function (displayed via the color gradient). The edge segment with the lowest combined cost is selected as solution and executed for the duration of one control cycle. As a result the solution approaches global optimality in purely static environments. In dynamic environments, on the other hand, no *formal* optimality guarantees can be provided due to the decentralized nature of C^n -CO.

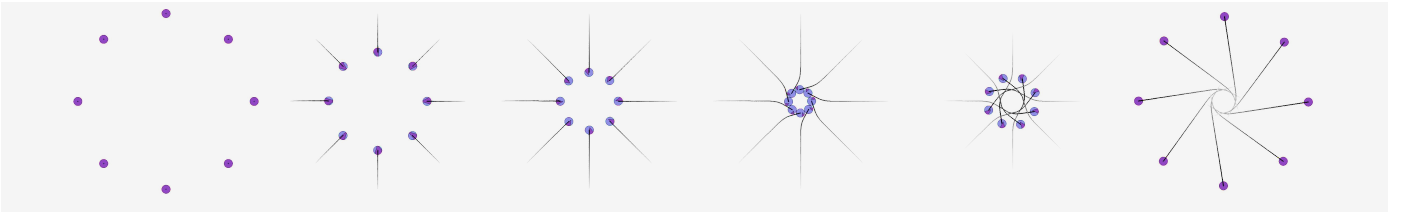


Fig. 9. Local multi-agent navigation. Circle swap with eight agents employing C^1 -CO. Eigenvalues are set at $\lambda_0 = 0.5 \frac{1}{s}$. Once agents have agreed on an avoidance *topology*, the physical constraints inherent to the agents preserve stability. By employing a navigation scheme with a slight avoidance preference to a common side, efficient avoidance patterns reminiscent of roundabouts emerge.

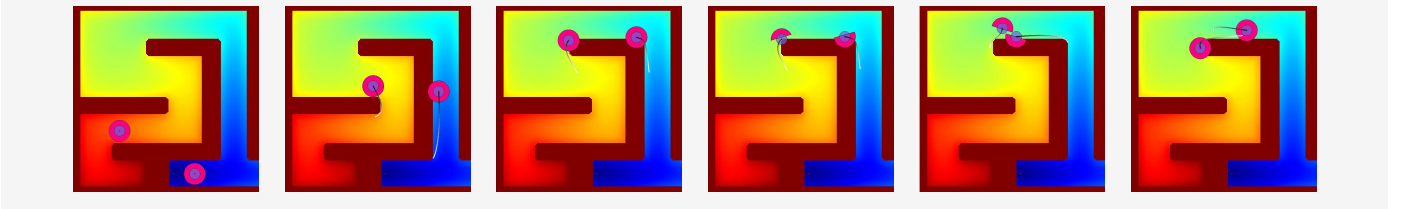


Fig. 10. Global multi-agent navigation. Both agents employ the heuristic graph search strategy described in Section VI-C: first, fair partitions in absolute velocity space are computed between interacting agents using the C^1 -CO method with a time horizon of $\tau = 4$ s (colored in pink). Then, the associated trajectory segments are collision checked using the static map. At the same time a Field Dijkstra approach computes the cost-to-go in absence of dynamic objects (displayed as a color gradient). Finally, the sampled C^1 -continuous trajectory admitting the lowest combined cost is selected for execution during the next control cycle. A color video of this experiment is available at <http://ieeexplore.ieee.org>.

D. Runtime Analysis

Robotic collision avoidance approaches by definition need to be executable online. It is thus of interest to analyze the performance and scalability of the methods presented in Sections III–IV. C^n -CO is composed out of five main steps, including the setting of a preferred velocity $\mathbf{v}^{\text{pref}}$, the determination of neighboring agents within interaction distance, the computation of a collision cone (CC) for each neighboring agent, the solution of the associated linear program (LP), and the updating, and thus forward simulation of agents' states. While the first two and the fourth step are identical for the considered methods (but may depend on the overall number of agents in the scenario), steps three and five induce additional computational complexity for $n > 0$: the computation of a curved collision cone takes up to 10^3 times longer than in the linear case. This is owed to the required sampling of the cone's boundary, followed by a convex hull computation. Similarly, for $n > 0$, the update of agents' states needs to be performed via forward simulation of an agent model, which also produces an (albeit minor) computational overhead. Table I provides maximal runtime results for each of these steps for C^0 -CO (i.e., ORCA [19]), εC^0 -CO (i.e., NH-ORCA [21]), as well as C^1 -CO. The provided timings are in seconds per agent (considering $n \in \{1, 10, 100\}$ neighbors, if applicable) and have been compiled on a single core of an Intel®Core™2 Duo running at 2.53 GHz. Overall, C^1 -CO is two orders of magnitude slower than C^0 -CO, but remains online capable.

VII. DISCUSSION

The presented Control Obstacle of order n (C^n -CO) generalizes the VO concept and allows *holonomic* robots to move along C^n -continuous trajectories while performing reciprocal

TABLE I
RUNTIME COMPARISON BETWEEN C^0 -CO, εC^0 -CO AND C^1 -CO IN LOCAL MULTI-AGENT SCENARIOS.

Step Descr.	# Neigh.	C^0 -CO	εC^0 -CO	C^1 -CO
Set $\mathbf{v}^{\text{pref}}$	N/A	$3 \cdot 10^{-7}$ s	$3 \cdot 10^{-7}$ s	$3 \cdot 10^{-7}$ s
Neigh. search	1	$2 \cdot 10^{-8}$ s	$2 \cdot 10^{-8}$ s	$2 \cdot 10^{-8}$ s
	10	$3 \cdot 10^{-7}$ s	$3 \cdot 10^{-7}$ s	$3 \cdot 10^{-7}$ s
	100	$5 \cdot 10^{-6}$ s	$5 \cdot 10^{-6}$ s	$5 \cdot 10^{-6}$ s
Compute CC	N/A	$4 \cdot 10^{-7}$ s	$4 \cdot 10^{-7}$ s	$6 \cdot 10^{-4}$ s
Solve LP	1	$2 \cdot 10^{-8}$ s	$2 \cdot 10^{-8}$ s	$2 \cdot 10^{-8}$ s
	10	$6 \cdot 10^{-8}$ s	$6 \cdot 10^{-8}$ s	$6 \cdot 10^{-8}$ s
	100	$4 \cdot 10^{-7}$ s	$4 \cdot 10^{-7}$ s	$4 \cdot 10^{-7}$ s
Update state	N/A	$1 \cdot 10^{-6}$ s	$2 \cdot 10^{-6}$ s	$2 \cdot 10^{-6}$ s

collision avoidance without the necessity of artificially inflating their shape. Furthermore, through feedback linearization we demonstrated that a large class of general linear as well as non-linear robot models can be brought into a form suitable for C^n -CO. The two methods covered—static and dynamic feedback linearization—both pose benefits and shortcomings.

As shown in Section IV-C static feedback linearization is applicable to fully controllable points only. This implies that the method is not suitable for non-holonomic systems directly, but requires a redefinition of the reference point away from the wheel axles. Since the relative velocity paradigm can only handle circular robots this usually results in an increased robot radius. On the other hand, the order of the statically feedback linearized system is retained and thus the same controller (with identical parameters) can be chosen for all interacting robots independent of their kinematic characteristics.

Conversely, as shown in Section IV-D, dynamic feedback linearization can often be directly applied to non-holonomic

reference points. This prevents an artificial inflation in robot radius. However, in heterogeneous environments it may result in robots following control laws of different order of continuity, which do not directly admit a common treatment under the presented framework. Equation (5) shows, however, that the trajectories of higher order models are composed of their lower-order counterparts and ever less relevant additional terms. These additional terms represent rapidly decaying elements (depicted in Figure 3). In presence of several robot types using C^n -COs of different degrees of continuity, an additional inflation on the robot shape (equal to the decaying parts) may hence be considered to compensate for the difference between the expected and the actual trajectory of other vehicles. This suggests the combination of C^n -CO with the method presented in [23] for certain multi-platform scenarios.

In a series of experiments we demonstrated the gain in navigation performance C^n -CO provides over inflated C^0 -CO. Benefits have been shown to be largest for slow agent dynamics (i.e., $\lambda_0 \leq 0.25 \frac{1}{s}$) or moderately to densely populated scenes (i.e., obstacle and agent density $\rho \geq 0.2$)—and it is under these circumstances where the employment of a higher-order C^n -CO formulation pays off most.

VIII. CONCLUSIONS & FUTURE WORK

In this article we introduced the Order n Continuous Control Obstacle (C^n -CO), which allows holonomic agents to move along collision-free C^n -continuous trajectories while performing local reciprocal collision avoidance among a homogeneous set of feedback-controlled agents. While holonomic robots rarely require trajectories of continuity higher than C^1 , we showed that a large class of robot models can be brought into the normal form suitable for C^n -CO via feedback linearization, thus extending the method's applicability to general linear and many nonlinear agents including differentially-driven robots, and Ackerman vehicles. We illustrated that any degree of continuity can be achieved at these models' *actuators* via appropriate selection of C^n -CO's order of continuity.

We further established that C^n -CO generalizes reciprocal VO and several recent VO extensions: first, the approach of [20] has been shown to be equivalent to applying C^0 -CO after static feedback linearization in an off-axis point—as discussed in Section IV-C. We further extended this method to cars and cars pulling trailers. Second, the approach of [22] represents a specific instance of C^1 -CO. Third, the LQR-Obstacle method [30] corresponds to C^n -CO for general *linear* systems where eigenvalues are determined according to the solution of an algebraic Riccati equation.

In summary, C^n -CO represents an extension of the reciprocal VO concept towards solution trajectories of higher continuity, including actuator continuity for non-holonomic agents and a unification of several recent extensions to the basic reciprocal VO framework. Given the promising properties of C^n -CO, we are currently investigating approaches to solve the main remaining limitations of the relative velocity concept as laid out in Section I-C.

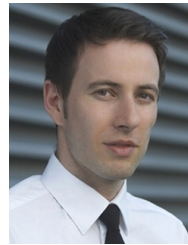
ACKNOWLEDGEMENT

This work has been supported by the European project *V-Charge* under contract number FP7-ICT 269916.

REFERENCES

- [1] H. J. Sussmann and J. C. Willems, “300 Years of Optimal Control: from the Brachistochrone to the Maximum Principle,” *IEEE Control Systems*, vol. 17, no. 3, pp. 32–44, 1997.
- [2] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA, USA: Athena Scientific, 2001.
- [3] P. E. Hart, N. J. Nilsson, and B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” in *Autonomous Mobile Robots: Perception, Mapping, and Navigation*, 1991.
- [4] S. Koenig and M. Likhachev, “Improved Fast Replanning for Robot Navigation in Unknown Terrain,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2002.
- [5] M. Pivtoraiko, R. A. Knepper, and A. Kelly, “Differentially Constrained Mobile Robot Motion Planning in State Lattices,” *Journal of Field Robotics*, vol. 26, no. 1, pp. 308–333, 2009.
- [6] M. Rufli and R. Siegwart, “On the Design of Deformable Input and State-Lattice Graphs,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2010.
- [7] S. M. LaValle and J. J. Kuffner, “Randomized Kinodynamic Planning,” *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.
- [8] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots, 2nd Ed.* Cambridge, MA, USA: Bradford Company, 2010.
- [9] S. Karaman and E. Frazzoli, “Incremental Sampling-based Algorithms for Optimal Motion Planning,” in *Robotics: Science and Systems (RSS)*, 2010.
- [10] J. P. Gonzalez and M. Likhachev, “Search-Based Planning with Provable Suboptimality Bounds for Continuous State Spaces,” in *Symposium on Combinatorial Search (SOCS)*, 2011.
- [11] P. R. Wurman, R. D’Andrea, and M. Mountz, “Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses,” in *Proc. of the Nat. Conf. on Innovative Applications of Artificial Intelligence*, 2007.
- [12] P. Trautman and A. Krause, “Unfreezing the Robot: Navigation in Dense, Interacting Crowds,” in *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.
- [13] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, “Image and Animation Display with Multiple Robots,” *Int. Journal of Robotics Research*, vol. 31, pp. 753–773, 2012.
- [14] S. Hoogendoorn and P. H.L. Bovy, “Simulation of Pedestrian Flows by Optimal Control and Differential Games,” *Optimal Control Applications and Methods*, vol. 24, no. 3, pp. 153–172, 2003.
- [15] D. Mellinger, A. Kushleyev, and V. Kumar, “Mixed-Integer Quadratic Program Trajectory Generation for

- Heterogeneous Quadrotor Teams,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012.
- [16] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. P. How, and G. Fiore, “Real-Time Motion Planning With Applications to Autonomous Urban Driving,” *IEEE Transactions on Control Systems Technology*, vol. 17, pp. 1105–1118, 2009.
- [17] N. E. Du Toit and J. W. Burdick, “Robotic Motion Planning in Dynamic, Cluttered, Uncertain Environments,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2010.
- [18] J. van den Berg, M. C. Lin, and D. Manocha, “Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2008.
- [19] J. van den Berg, S. J. Guy, M. C. Lin, and D. Manocha, “Reciprocal n-Body Collision Avoidance,” in *Proc. of the Int. Symposium on Robotics Research (ISRR)*, 2009.
- [20] J. Snape, J. van den Berg, S. J. Guy, and D. Manocha, “Smooth and Collision-Free Navigation for Multiple Robots Under Differential-Drive Constraints,” in *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.
- [21] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, “Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots,” in *Proc. of the Int. Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2010.
- [22] J. van den Berg, J. Snape, S. J. Guy, and D. Manocha, “Reciprocal Collision Avoidance with Acceleration-Velocity Obstacles,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.
- [23] J. Alonso-Mora, A. Breitenmoser, P. Beardsley, and R. Siegwart, “Reciprocal Collision Avoidance for Multiple Car-like Robots,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012.
- [24] P. Fiorini and Z. Shiller, “Motion Planning in Dynamic Environments using Velocity Obstacles,” *Int. Journal of Robotics Research*, vol. 17, pp. 760–772, 1998.
- [25] D. Wilkie, J. van den Berg, and D. Manocha, “Generalized Velocity Obstacles,” in *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.
- [26] J. Minguez and L. Montano, “Extending Reactive Collision Avoidance Methods to Consider any Vehicle Shape and the Kinematics and Dynamic Constraints,” *IEEE Transactions on Robotics*, vol. 25, no. 2, pp. 367–381, 2009.
- [27] E. W. Weisstein, *CRC Concise Encyclopedia of Mathematics, 2nd Ed.* Boca Raton, FL, USA: Chapman and Hall/CRC, 2002.
- [28] H. K. Khalil, *Nonlinear Systems, 3rd Ed.* Prentice Hall, New Jersey, NJ, USA, 2002.
- [29] A. De Luca, G. Oriolo, and C. Samson, “Feedback Control of a Nonholonomic Car-like Robot,” in *Robot Motion Planning and Control*, ser. Lecture Notes in Control and Information Sciences, J. Laumond, Ed. Springer Berlin / Heidelberg, 1998, vol. 229, pp. 171–253.
- [30] J. van den Berg, D. Wilkie, S. J. Guy, M. Niethammer, and D. Manocha, “LQG-Obstacles: Feedback Control with Collision Avoidance for Mobile Robots with Motion and Sensing Uncertainty,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012.
- [31] D. F. Bareiss and J. van den Berg, “Reciprocal Collision Avoidance for Quadrotor Helicopters using LQR-Obstacles,” in *AAAI Workshop on Multi-Agent Pathfinding (WOMP)*, 2012.
- [32] R. Philippsen, “A Light Formulation of the E* Interpolated Path Replanner,” Autonomous Systems Lab, Ecole Polytechnique Fédérale de Lausanne, Tech. Rep., 2006.
- [33] 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.



Martin Rufli is a postdoctoral research fellow and lecturer at the Autonomous Systems Lab at ETH Zurich since 2012. He holds a B.Sc. (2006) and a M.Sc. (2008) degree in Mechanical Engineering, as well as a Ph.D. (2012) in Robotics, all from ETH Zurich. His current research interests include the design of novel algorithms for robotic navigation with the aim to bring robots out of factory lines and into our everyday lives. The main focus in this regard concerns system-based online methods and algorithms which model robot-robot interaction.



arships from UPC Barcelona, the Spanish Research Council and the Swiss Government.

Javier Alonso-Mora is a PhD candidate at ETH Zurich and Disney Research Zurich. He holds Diplomas in Mathematics (2008) and Engineering (2010) from UPC Barcelona and a M.Sc. in Robotics, Systems and Control (2010) from ETH Zurich, for which he received the Willi-Studer award. His current research focuses on multi-robot navigation, with an emphasis on real-time algorithms that model inter-agent decision-making. Further, he is interested on developing intuitive methods for human-robot swarm interaction. He has received excellence schol-



arships from UPC Barcelona, the Spanish Research Council and the Swiss Government.

Roland Siegwart is full professor for Autonomous Systems and Vice President Research and Corporate Relations at ETH Zurich since 2006 and 2010, respectively. From 1996 to 2006 he was associate and later full professor for Autonomous Microsystems and Robotics at the Ecole Polytechnique Fédérale de Lausanne (EPFL). Roland Siegwart is member of the Swiss Academy of Engineering Sciences, IEEE Fellow and officer of the International Federation of Robotics Research (IFRR). He served as Vice President for Technical Activities (2004/05), Distinguished Lecturer (2006/07) and AdCom Member (2007–2009) of the IEEE Robotics and Automation Society. He leads a research group of around 30 people working on several aspects of robotics.

Information and Communication Technologies

Collaborative Project

“V-Charge”

Autonomous Valet Parking and Charging for e-Mobility

Grant Agreement Number 269916



Motion Planning for Parking - Status at June 2013

K. Mimura
(Robert Bosch GmbH, Corporate Research, CR/AEV)

1. Overview of Parking Task Processor

The parking task processor provides collision-free trajectories to park into assigned parking spots or charging stations or to pull out based on object information detected by sensors. The task processor is triggered when the vehicle approaches target parking spots or charging stations and executes path planning for the first time (first-planning). Due to limited sensor field of views, it is impossible to detect all objects in an environment when this task processor is triggered. For safety navigation, it is necessary to check if the current trajectory causes any collisions with newly detected objects or not and to update it to avoid collisions till the vehicle reaches the target position (re-planning). For this purpose, the parking task processor consists of three main parts: path planning algorithm, re-planning logic and state machine.

As shown in Figure 1, the state machine decides next actions (for example, execute the first-planning, execute the re-planning, abort the task and so on) depending on situations. The path planning algorithm is a core algorithm to provide trajectories which is used in both of the first-planning and the re-planning. For the first-planning, the core algorithm is used directly. On the other hand, there is an additional logic in the re-planning, in order to utilize already existing trajectory.

These three parts are described in detail in the following sections.

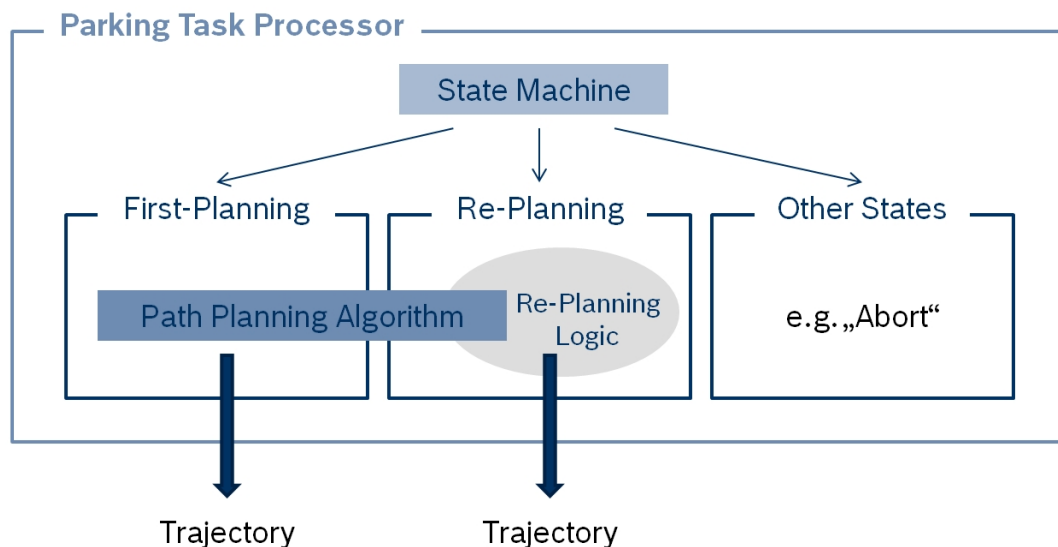


Figure 1: Structure of Parking Task Processor

2. Path Planning Algorithm

The focuses in the development of the core path planning algorithm are to minimize calculation time and to provide intuitive forms of trajectories. In order to react in dynamic environments safely, the calculation time for path planning has to be minimized to avoid collisions especially with dynamic objects. Furthermore, the robot vehicles need to behave like those driven by human drivers for high acceptance in a mixed traffic with robot

vehicles and human drivers. It is assumed that a number of driving direction changes and their positions are important factors for subjective impressions on the trajectory forms.

Due to these focuses, an algorithm with a combination of a geometric path generation based on polynomial and a graph-search based on A* in a 3D state space which is called "State Lattice Planner [1]" is developed. The algorithm executes the geometric path generation first of all (Step-1 in Figure 2). The geometric path generation calculates a polynomial with 5 orders to connect 2 poses in 4D expression (position in X, position in Y, heading and curvature) with considering kinematic constraints of the vehicle. If the generated trajectory is collision-free, then the trajectory is provided as a solution (Case "a" in Figure 2). If the trajectory causes a collision (Case "b" in Figure 2) or the geometric path generation cannot connect two poses due to the kinematic constraints, then the graph-search is triggered (Step-2 in Figure 2). For the graph-search, A* is implemented in a 3D state space (position in X, position in Y and heading). The resolution of the state space is adjustable through parameters. Currently, the resolution for the positions (in X and Y) is defined between 10 and 20 [cm] and for the heading defined between 15 and 22.5 [deg]. In tight spaces, a fine resolution of the state space (10 [cm] and 15 [deg]) increases the probability to find trajectories, but it increases the calculation-time. The resolution is fixed independent of situations at this moment. Variable resolution depending on situations is a possibility for further modifications. In order to guarantee accuracy at the goal pose, the state space is aligned so that the goal pose is exactly located on one of states.

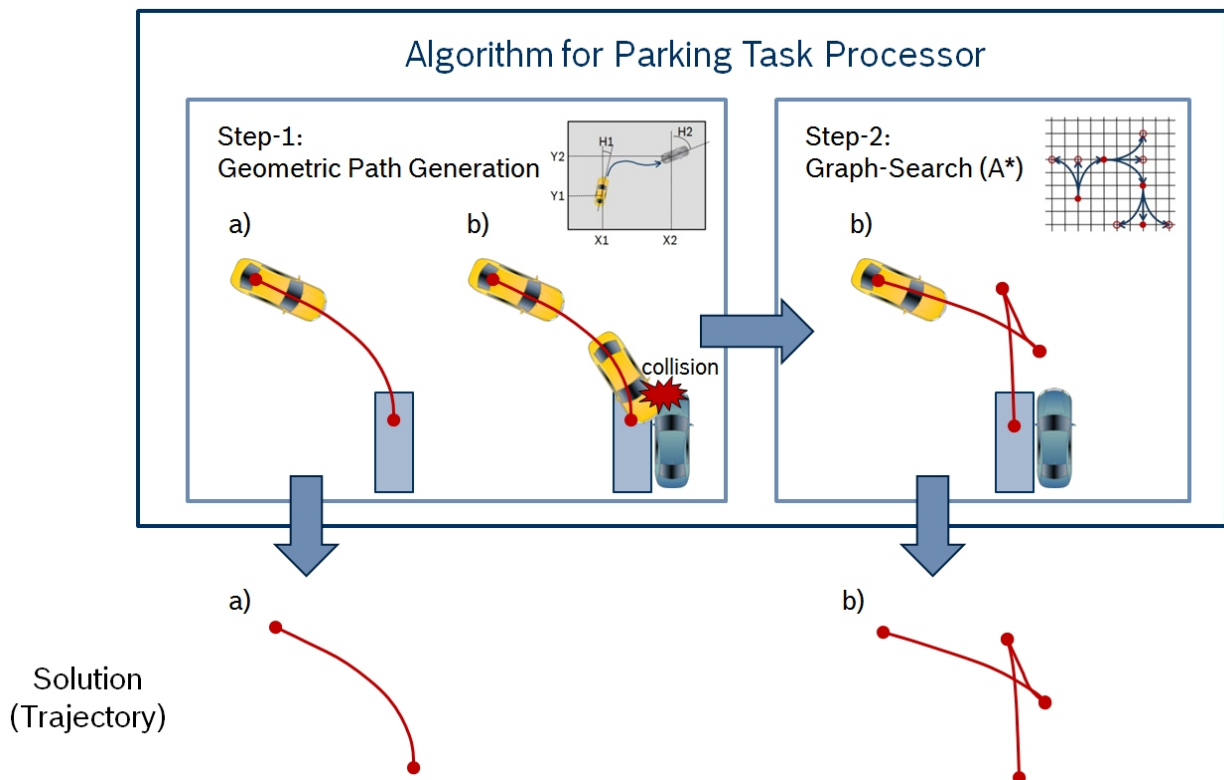


Figure 2: Algorithm of the parking task processor

The geometric path generation provides smooth trajectories between two poses quickly. However, its flexibility is limited because it is impossible to provide trajectories with collision avoidances or with driving direction changes. As there is no driving direction

change, all trajectories provided by the geometric path generation are assumed as intuitive.

On the other hand, the graph-search finds collision-free trajectories with high probability in various situations (high flexibility), but it takes longer time to find a trajectory due to necessary iterations. In order to overcome this disadvantage of the graph-search, some countermeasures are implemented:

- Motion primitives and driving swath (area passed over by the vehicle) are calculated offline and referred to during online planning
- Heuristic Look-Up Table [2] is created offline based on Dijkstra's search and referred to during online planning to reduce the iterations

In order to adjust the number of the driving direction changes or their positions, some parameters and functions to calculate penalty costs for driving direction changes and extremely short segments between two driving direction changes are implemented.

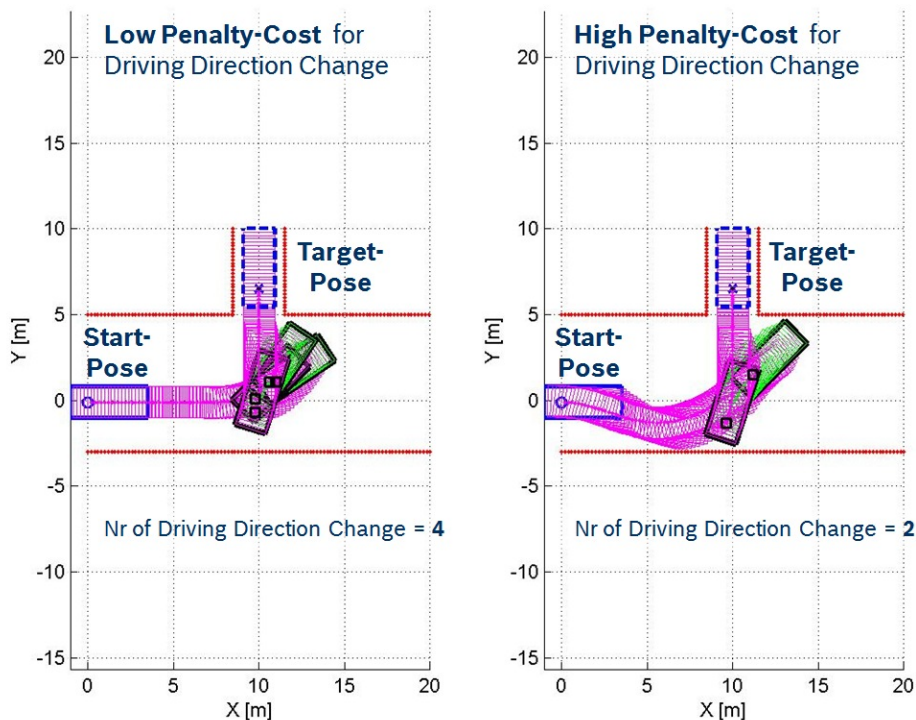


Figure 3: Effect of a penalty cost for driving direction changes

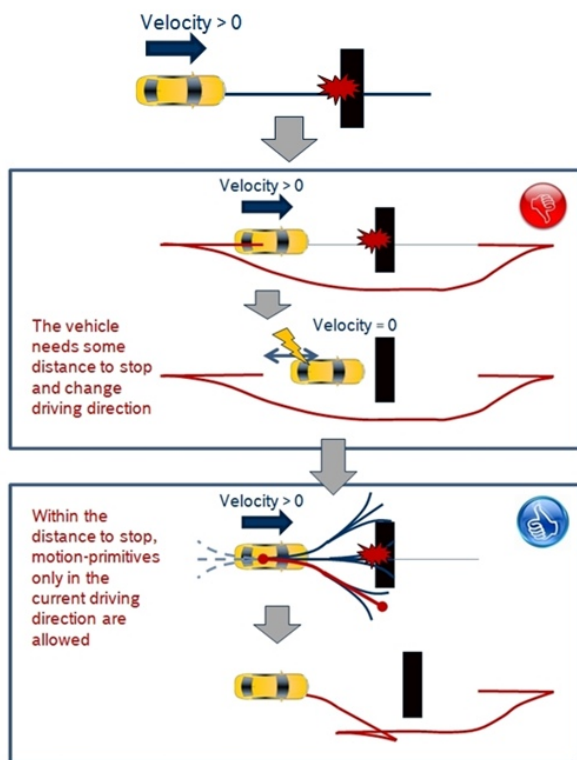
An example showing the effect of a penalty cost for driving direction changes is shown in Figure 3. If a smaller value is set to the penalty cost, more driving direction changes are included. By adjusting these penalty costs, there is a potential to realize intuitive trajectories.

As it is expected that the path planning is executed during driving especially in case of re-planning, a concept of "path planning constraint" is implemented to provide traceable trajectories with considering actual vehicle conditions. There are two kinds of constraints; driving direction constraint and curvature constraint.

The driving direction constraint (Case “a” in Figure 4) limits the driving direction in a certain distance from the start pose to avoid any new trajectories starting with the other driving direction. As the vehicle needs some distance to stop and to change the driving direction, such trajectory with the other driving direction at the beginning is not traceable.

The curvature constraint (Case “b” in Figure 4) selects a motion primitive whose required steering angle at the start pose is close to the actual steering angle of the vehicle for the first part of the new trajectory. With this constraint, any trajectories requiring discontinuous steering behavior are avoided.

a) Driving direction constraint



b) Curvature constraint

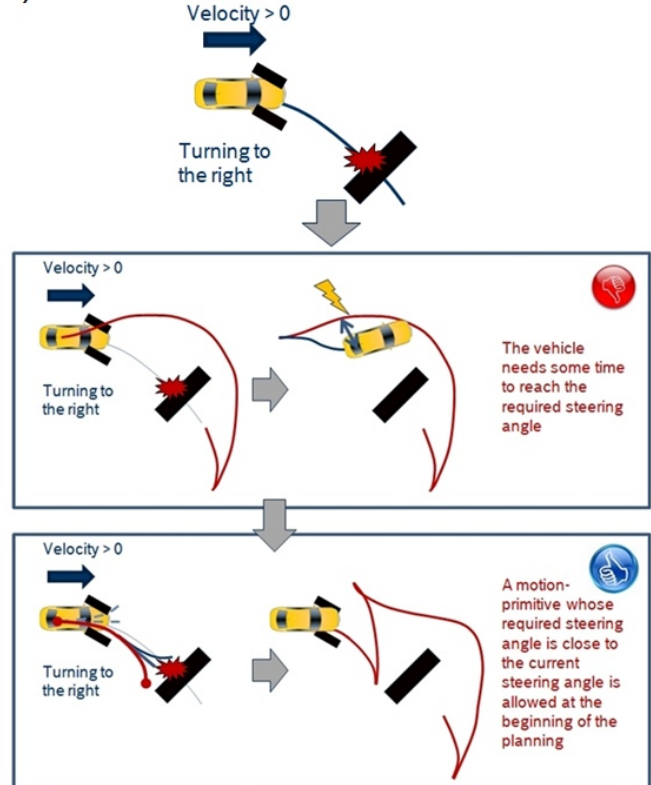


Figure 4: Driving direction constraint and curvature constraint in the path planning

Furthermore, performance of the developed algorithm needs to be evaluated. The term of performance means mainly the calculation time and the “trajectory quality” here. The trajectory quality indicates how intuitive a trajectory is.

For the performance evaluation, a simulation environment was prepared. The algorithm is integrated as a MEX-File into a MATLAB simulation environment. As the calculation time of the graph-search highly depends on situations, some simulation scenarios (lateral parking scenario) with various start-poses (different positions and different headings) are defined (Figure 5). From each start-pose to a goal-pose defined in the scenario, a path planning is executed and the calculation time is measured. After executing from all start-poses, the averaged calculation time is calculated and the trajectory quality is analyzed. The simulation is executed on a computer with a 1.6 GHz CPU and 8 GB of RAM.

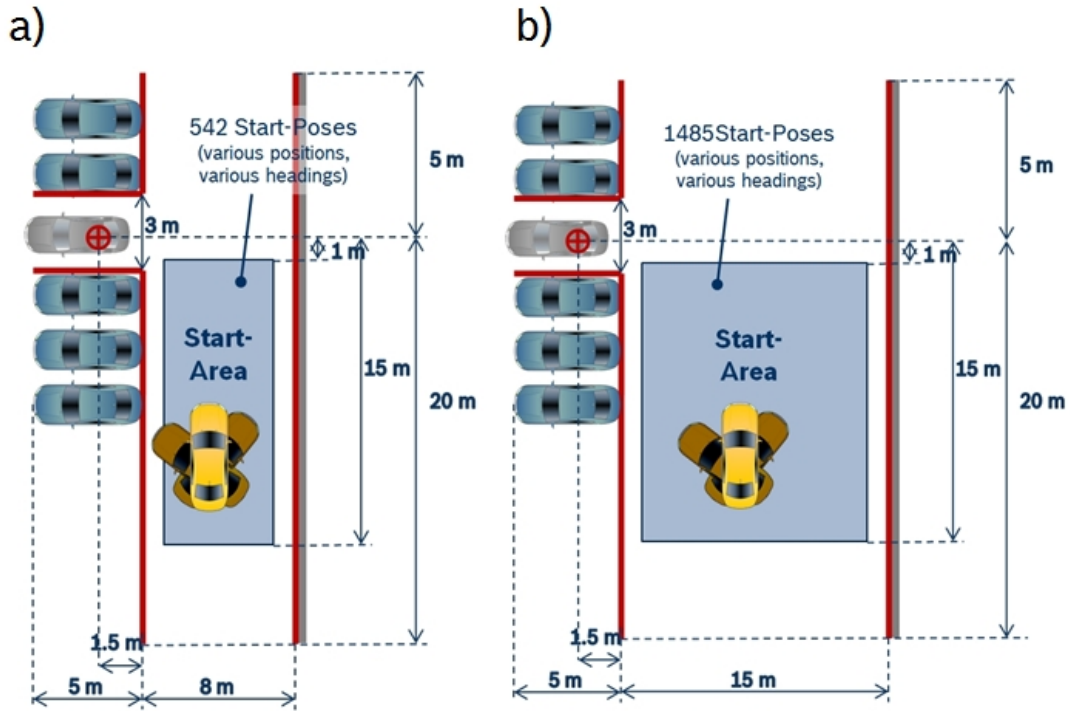


Figure 5: Examples of the simulation scenario

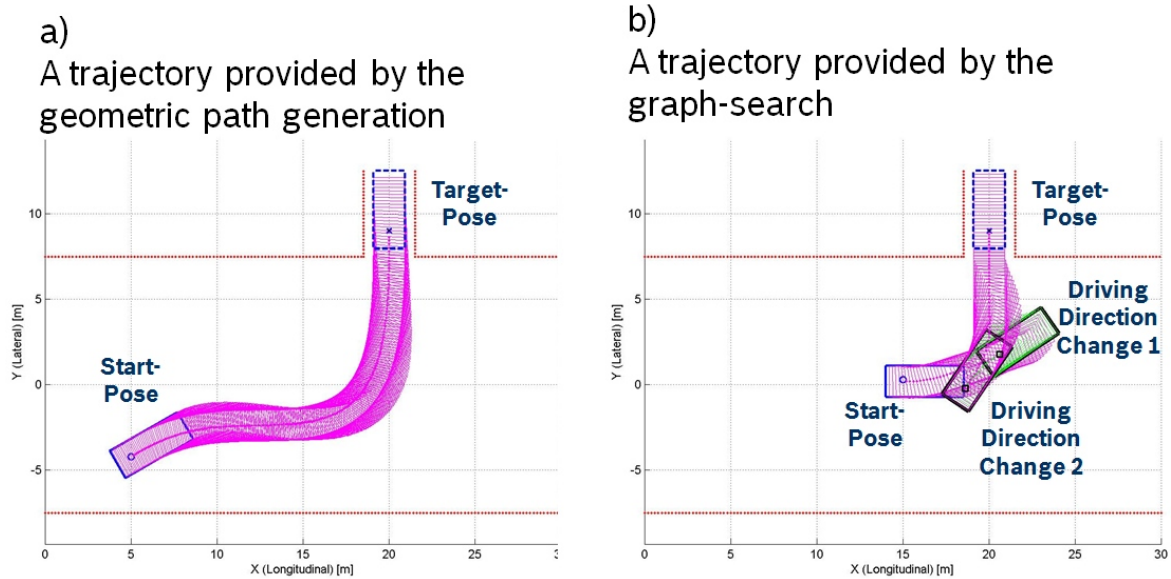


Figure 6: Examples of trajectories from simulations (scenario b in Fig.3)

In the scenario a) in Figure 5, there is no case where the geometric path generation provides collision-free trajectories. In all cases, trajectories are found by the graph-search. The averaged calculation time for the graph-search is less than 1 [s] with state space resolution of 20 [cm] in positions and 15 [deg] in headings.

In another scenario b) in Figure 5, there are 212 cases (about 14% of all cases) where the geometric path generation provides collision-free trajectories. The calculation time for the geometric path generation is less than 5 [ms]. The averaged calculation time for the

graph-search in all other cases is comparable to the scenario a) with state space resolution of 20 [cm] in positions and 15 [deg] in headings.

Regarding the trajectory quality, the number of driving direction changes are analyzed and compared between various settings for the penalty costs. Furthermore, criteria to a trajectory as “not intuitive” based on the positions of driving direction changes have not been defined yet. It is necessary to pick up “not intuitive” trajectories based on a subjective impression, extract criteria to detect them and compare the number of “not intuitive” trajectories between various penalty costs settings.

Further possibilities to modify the current algorithm are described as follows:

- Speed-up of algorithm by optimizing memory access
- Implementation of variable state space resolution depending on situations
- Definition of the best setting for penalty costs to realize intuitive forms of trajectories

3. Re-Planning

Provided trajectories are evaluated every 100 [ms] during parking maneuver. Updating trajectories (re-planning) is executed in two cases; deviation between the actual vehicle position and the existing trajectory exceeds a threshold, the existing trajectory has collisions with newly detected objects.

The reasons not to execute a new path planning every time but to implement the re-planning logic are explained as follows. First, the calculation time for updating has to be minimized for safety in dynamic environments as the re-planning is triggered mostly during driving. Because of this, existing trajectories need to be utilized as much as possible and a completely new path planning should be avoided. Second, often updating of trajectories potentially causes divergent behavior of the lateral controller because references for the controller possibly often change discontinuously. From this aspect, a short part of an existing trajectory from the actual vehicle pose should be held even if an updating of the trajectory is necessary to realize smooth controlling.

3.1 *Re-Planning due to large deviation between the vehicle and the trajectory*

Once the deviation between the actual vehicle position and the existing trajectory exceeds a threshold, the re-planning logic tries to connect the actual vehicle pose and one of the poses on the existing trajectory by the geometric path generation. If it is possible to connect two poses, the new trajectory to one of the poses on the existing trajectory and a part of the existing trajectory are merged and provided as a new trajectory (Figure 7). If there is no possibility to connect by the geometric path generation due to kinematic constraints or collisions with objects, the core algorithm is triggered and provides a completely new trajectory from the actual vehicle pose to the target pose. As explained in the sub section for the core path planning algorithm, the driving direction constraint and the curvature constraint are considered in the re-planning with the core algorithm.

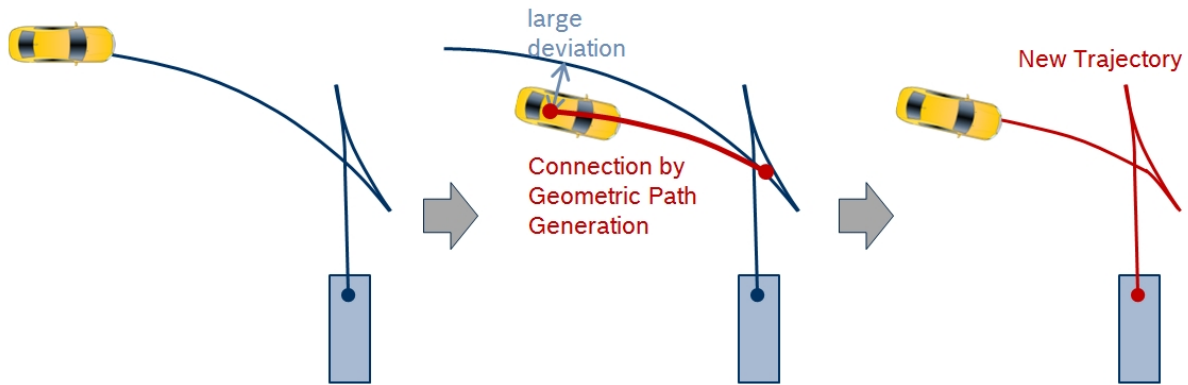


Figure 7: Re-Planning due to large deviation between the vehicle and the trajectory

3.2 Re-Planning due to collision

Once any collisions with newly detected objects are found, the re-planning logic checks if there is a driving direction change between the actual vehicle pose and a pose where a collision is detected. If there is no driving direction change in between, a certain length of the existing trajectory from the actual vehicle pose is held and the core algorithm provides a trajectory from the end pose of the part of the existing trajectory kept by the logic to the target pose (Case “a” in Figure 8). If there is a driving direction change before the next collision pose, the core algorithm provides a trajectory from the next driving direction change (Case “b” in Figure 8).

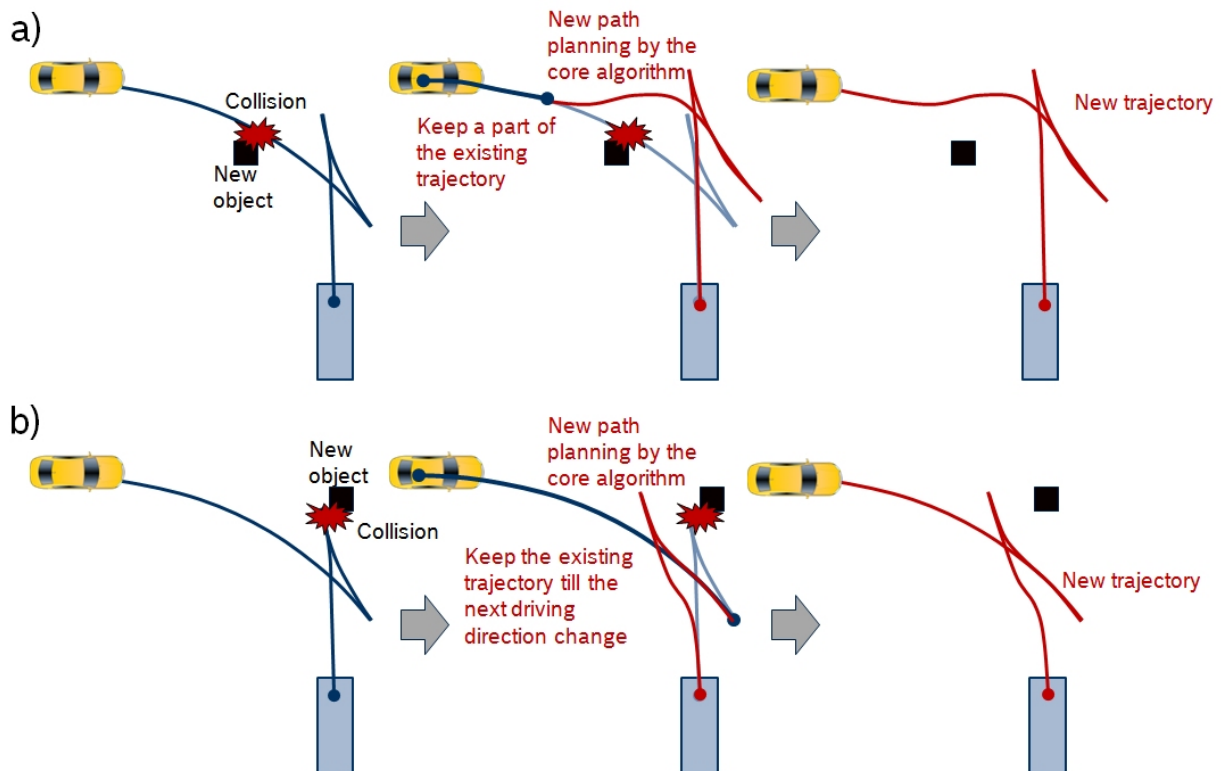


Figure 8: Re-Planning due to collision

4. State Machine

The state machine is responsible mainly to trigger path planning periodically. Once the parking task processor is triggered, the first planning state is activated. If there is a trajectory found, then the state is switched to the second planning. In the second planning phase, existing trajectories are evaluated periodically and re-planning is executed if necessary. The state stays in the second planning till the vehicle reaches the target pose.

Another important task of the state machine is error handling. In order to increase the robustness of the system, the parking process is not immediately aborted if no trajectory is found. If the vehicle is driving and no trajectory is found, a stop-trajectory is provided to stop the vehicle as soon as possible. Once the vehicle stops, the “recovery planning” is triggered. As the vehicle holds its position, the requirement for the short calculation time is not as strong as in the path planning during driving. In the recovery planning, the maximum limit for the iteration of the graph search is extended which results in more probability to find a trajectory. It is also an option in the recovery planning to take finer resolution of the state space for the graph search in the future.

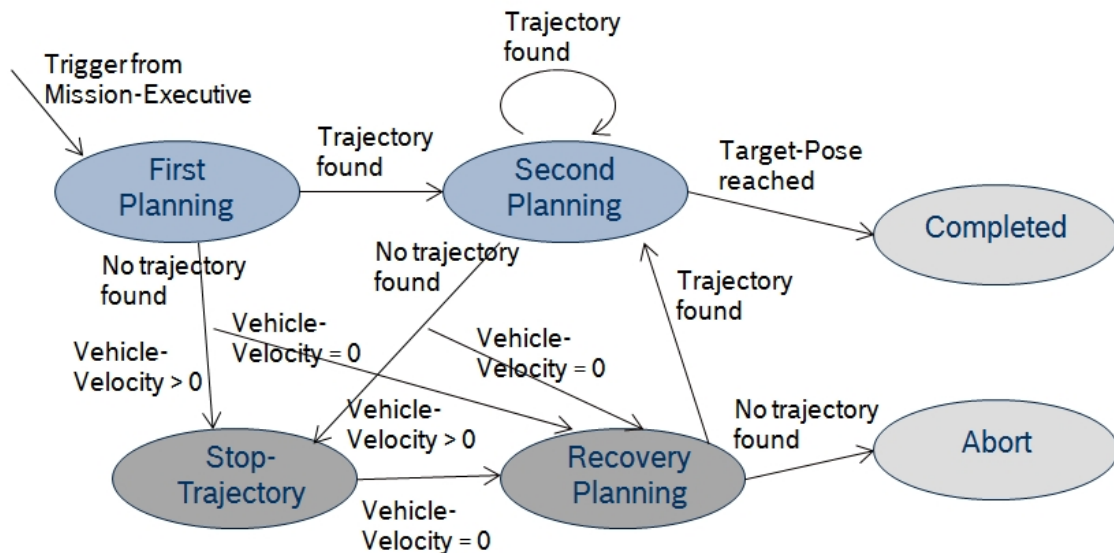


Figure 9: Actual design of the state machine

Further modifications are planned to minimize cases of task abort. At the same time, the state machine needs to be designed to react in dynamic environments. A new concept of the state machine will be designed and implemented in 2014.

5. References

- [1] Mihail Pivtoraiko, RA Ross A Knepper, and Alonzo Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308-333, 2009.
- [2] R Knepper and A Kelly. High Performance State Lattice Planning Using Heuristic Look-Up Tables. In *Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006.