Information and Communication Technologies
Collaborative Project

"V-Charge"

Autonomous Valet Parking and Charging for e-Mobility

Grant Agreement Number 269916



V-CHARGE

Deliverable Number	4.3
Deliverable Name	Motion Planning and Vehicle Control
WP Number	4
Date of Preparation	July 13, 2014
Date of Submission	June 1, 2014
Editor/Beneficiary	U. Schwesinger/ETH Zürich K. Mimura/RBG, R. Waldmann/VW

Contents

1 Introduction **2**

2 Navigation Modules Progress Overview **2**

2.1 Global Task Planner (VW) . . . 2

2.2 Online Partial Motion Planning (ETHZ-ASL) . . . 5

2.2.1 Scenarios in a Static Environment . . . 7

2.2.1.1 Scenario 3: Obstacle-Free Navigation Along a Reference Path . . . 7

2.2.1.2 Scenario 6: Static Obstacle Handling . . . 9

2.2.2 Scenarios in a Dynamic Environment . . . 11

2.2.2.1 Collision Detection . . . 11

2.2.2.2 Adaptations Due to Current Limitations in Perception . . . 12

2.2.2.3 Scenario 7: Platooning . . . 13

2.2.2.4 Scenario 9: Another Car Pulls Out . . . 14

2.2.2.5 Scenario 10+11: Coping With Pedestrians . . . 17

2.3 Off-Lane Planner (Bosch) . . . 19

2.3.1 Justification for On-Lane/Off-Lane Module-Separation . . . 19

2.3.2 Integration Advancements . . . 20

2.3.3 Performance of the Off-Lane Planning Module: Success Rate . . . 20

2.3.4 Overview of Sporadic Issues Hindering Successful Parking . . . 21

2.3.5 Category “Perception”: False-positives . . . 21

2.3.6 Category “Kinematics” . . . 22

3 Additional Efforts **26**

3.1 Automatic Summaries of Test Runs (ETHZ-ASL) . . . 26

4 Conclusion **28**

A Fast collision detection through bounding volume hierarchies in workspace-time space for sampling-based motion planners **29**

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.

This deliverable at hand details the efforts undertaken in workpackage 4 "Motion Planning and Vehicle Control". This work contains the following tasks:

- 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. The approaches taken within the V-Charge consortium to tackle tasks 4.2–4.4 were presented in deliverable 4.2. The remainder of this deliverable focusses on both novel aspects within the different modules forming the V-Charge planning framework and evaluation of these in real-world operation.

2 Navigation Modules Progress Overview

2.1 Global Task Planner (VW)

Within the V-Charge system, the global task planner is responsible to provide a global reference path to guide the automated vehicle in the static parking lot environment. The global reference path is computed considering all static information stored in the RoadGraph, encoding topological and semantic information about the parking lot structure. The global reference path is utilized by the local planner in order to navigate from the drop-off zone to its dedicated parking spot.

At the current stage of the project the global reference path is generated solely once at the beginning of a mission. This might yield dead lock situations in case the road is entirely blocked by obstacles that cannot be easily passed, causing the vehicle to wait in an endless loop. In case the target pose–i.e. the dedicated parking spot–is already occupied (e.g. by a human-operated vehicle that was not registered by the parking lot manager), the mission likewise fails. These use-cases motivate the integration of a replanning step in the high-level planning modules of the V-Charge system.

In the remainder of the project the global reference path will be recomputed whenever such a situation is detected by the currently active task processors executing the current mission. Within this section of the deliverable, we will describe the architecture enabling global replanning that will be implemented in the remainder of the project.

Figure 1 depicts a message sequence diagram for global replanning in a "blocked road" scenario. The local planner navigates along the reference path provided by the global task planner in the beginning but may experience a situation where it has to stop due to an unexpected event in front of it. The V-Charge vehicle will wait for a reasonable amount of time and issues a failure response to the mission executive if continuation

RoadGraphUpdate	Description
sequence ⟨EntityUpdate⟩ *entities*	array of updates

EntityUpdate	Description
long long *id*	the id of the entity to update, e.g. lane id
UpdateDescription *description*	all data necessary for the update

UpdateDescription	Description
bool *blocked*	entity is blocked or free
enum {LANE_BLOCKED, ROAD_BLOCKED, ...}	type of entity to update

Table 1: Description of *RoadGraphUpdate* message used for updating the topological road graph during global replanning.

of the mission is impossible. Additionally it will send a *RoadGraphUpdate* message, marking the according lane as not drivable. The involved data types for the *RoadGraphUpdate* message are summarized in table 1.

This message is forwarded to the global planner and to the server which distributes the information to other automated vehicles, which might benefit from the newly obtained information. The mission executive, acting as a coordinating instance, transfers a new mission goal to the global planner which includes the same target pose. The global planner replans with the updated roadgraph information. The resulting global path will differ from the previous one since the global planner received the lane update information. After sending a new task list to the mission executive, the next mission will be started.

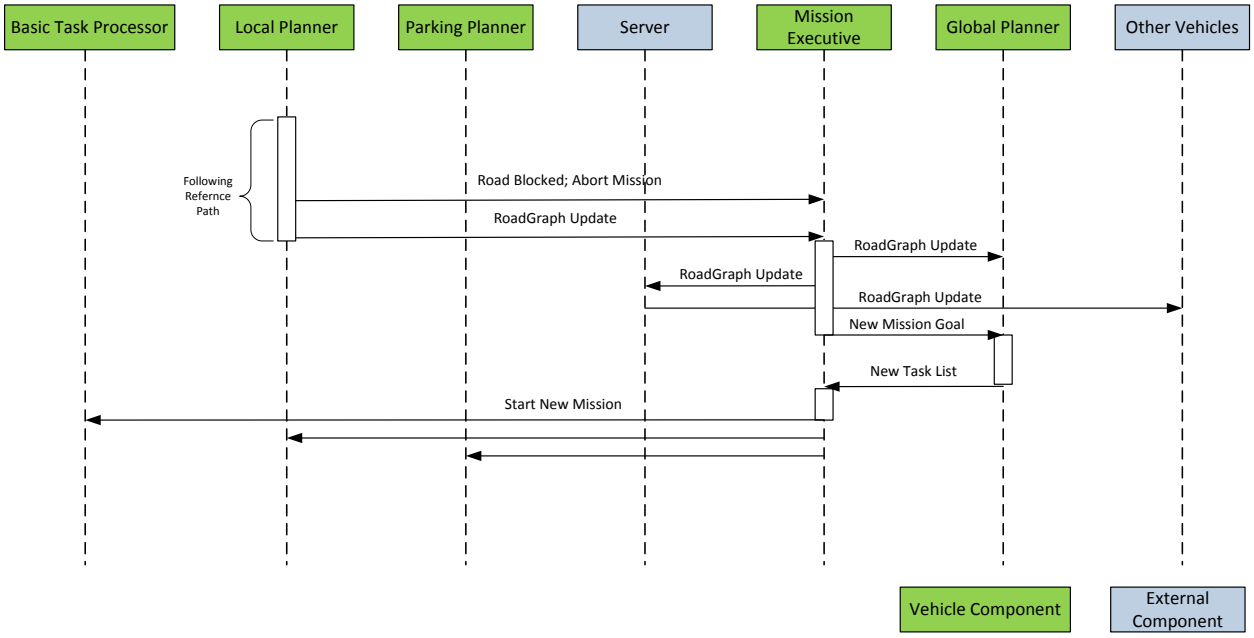


Figure 1: Sequence diagram of the signal flow if a road is blocked.

The second "blocked parking spot" scenario is illustrated in Figure 2 . Again, the vehicle follows the reference path but instead of a blocked road, now the designated parking spot is already occupied. The local planner reports the successfulll completion of navigating to the parking spot. The mission executive then triggers

the parking task. The parking planner is unable to obtain a collision-free path into the parking spot and consequently triggers a *RoadGraphUpdate* message which contains the id of the unreachable parking spot. A new parking spot is assigned by the server triggering the global planning chain again.

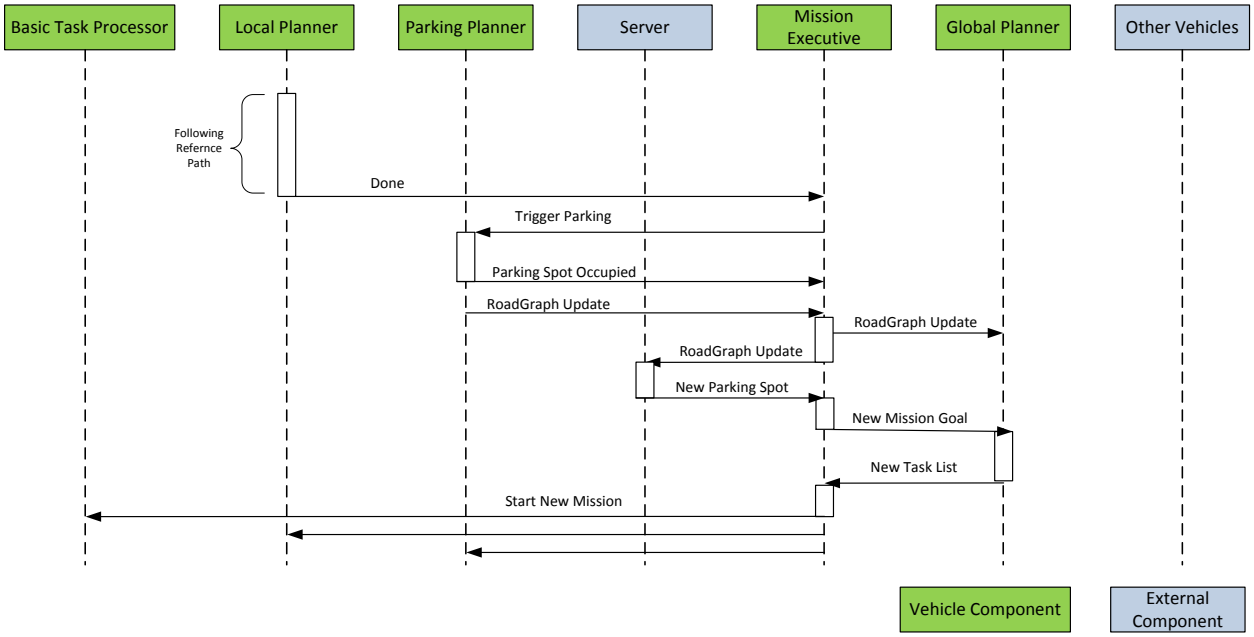


Figure 2: Sequence diagram of the signal flow if the parking spot is occupied.

2.2 Online Partial Motion Planning (ETHZ-ASL)

This section reports on the progress and performance of the online local planning framework deployed on the V-Charge vehicles. In the previous deliverable 4.2 focus was laid on the approaches chosen and the implementation of such. In the document at hand we focus on the performance of the planning suite in different situations.

The local motion planning framework is operating on the V-Charge vehicles for more than one year. The major changes since the last report can be summarized as follows:

- Fine-tuning of vehicle model: Incorporation of acceleration deadband and handbrake within the model prediction step of the local planner (numerical forward simulation of vehicle model).
- Principled treatment of moving objects through fast collision detection over time with a bounding volume hierarchy approach in workspace-time space, see A.
- Platooning mode for smooth following behavior when encountering a preceding car. This is achieved by adapting the reference speed for the planning module based on distance and relative speed along the lane with respect to the target vehicle. This approach avoids undesirable lateral motions in case the ego vehicle tries to overtake while maintaining the full capabilities of collision avoidance with both static and dynamic objects.

Illustrative material (see Figure 3 for a detailed explanation of the plots) will be shown from data recorded during the extensive V-Charge demo preparations in the Bosch parking garage in Stuttgart, Germany.

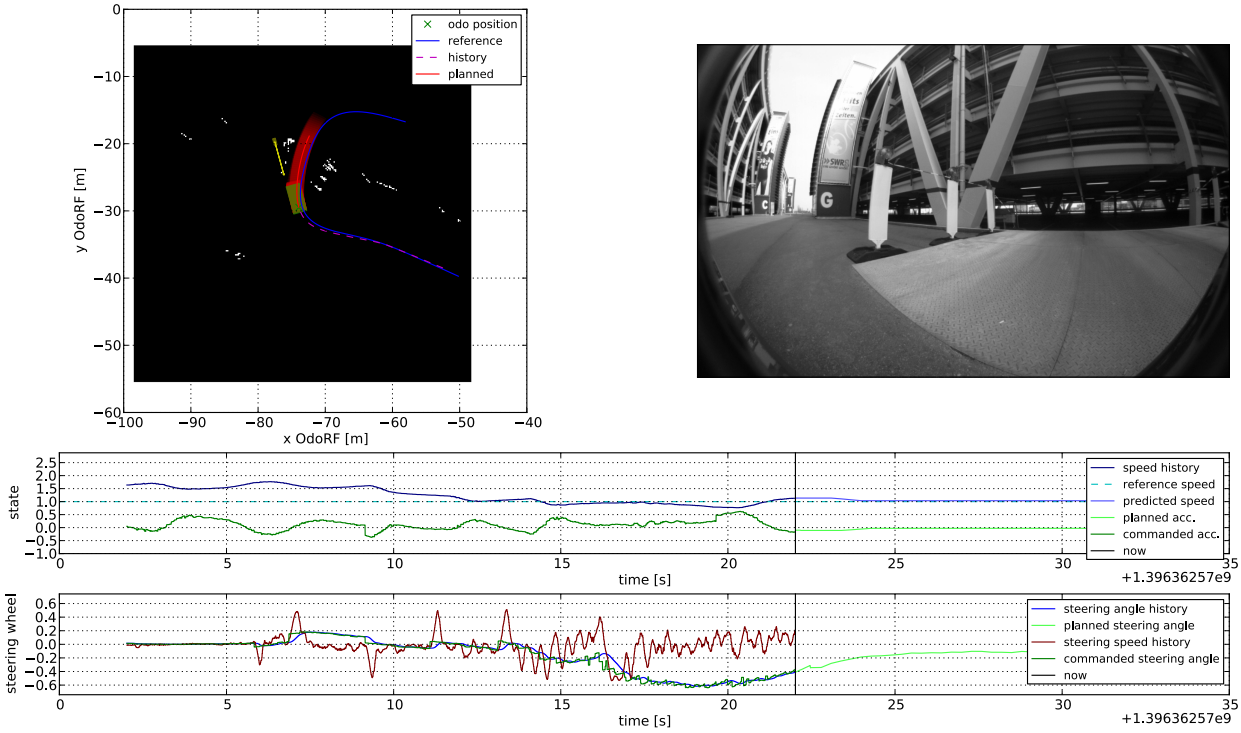


Figure 3: Exemplary plot for the evaluation of the local planning framework. *Top left*: occupancy grid (black/white = free/occupied cells), ego position (green patch), dynamic objects (yellow patch) together with their velocity vector (yellow arrow, tip at $t_{\text{now}} + 10\,\text{sec}$), a section from the reference path (blue solid line), driven path (dotted magenta line) and local plan produced by the local planning module (red line) together with the predicted swept volume (red patches) is shown. *Top right*: Image from the front monocular camera. *Middle*: Plot of the longitudinal vehicle states. Reference speed (dotted cyan line), sequence of acceleration system inputs from the plan (solid light green line in future), predicted speed of the vehicle from the local planner (light solid blue line in future) and history of vehicle speed (dark solid blue line) and acceleration commands (dark solid green line) are depicted. *Bottom*: Lateral vehicle states. Sequence of steering angle system inputs from the plan (solid light green line) and history of steering angles (dark solid blue line), steering speeds (solid dark red line) and steering angle commands (solid dark green line) are shown.

The performance is demonstrated in various scenarios as defined in deliverable 6.1 and depicted in Figure 4. The scenarios which received most attention from developer side so far are scenarios 3, 6, 7 and 8. Scenarios 10 and 11 can only be partially tackled at the time of preparation as the pedestrian classification and tracking framework has not been integrated into the online system. Scenario 9 is feasible regarding perception, but performance can be improved on both perception side and planning. Scenario 12 is currently not practicable regarding perception, and scenario 13 was not tackled up to this point.

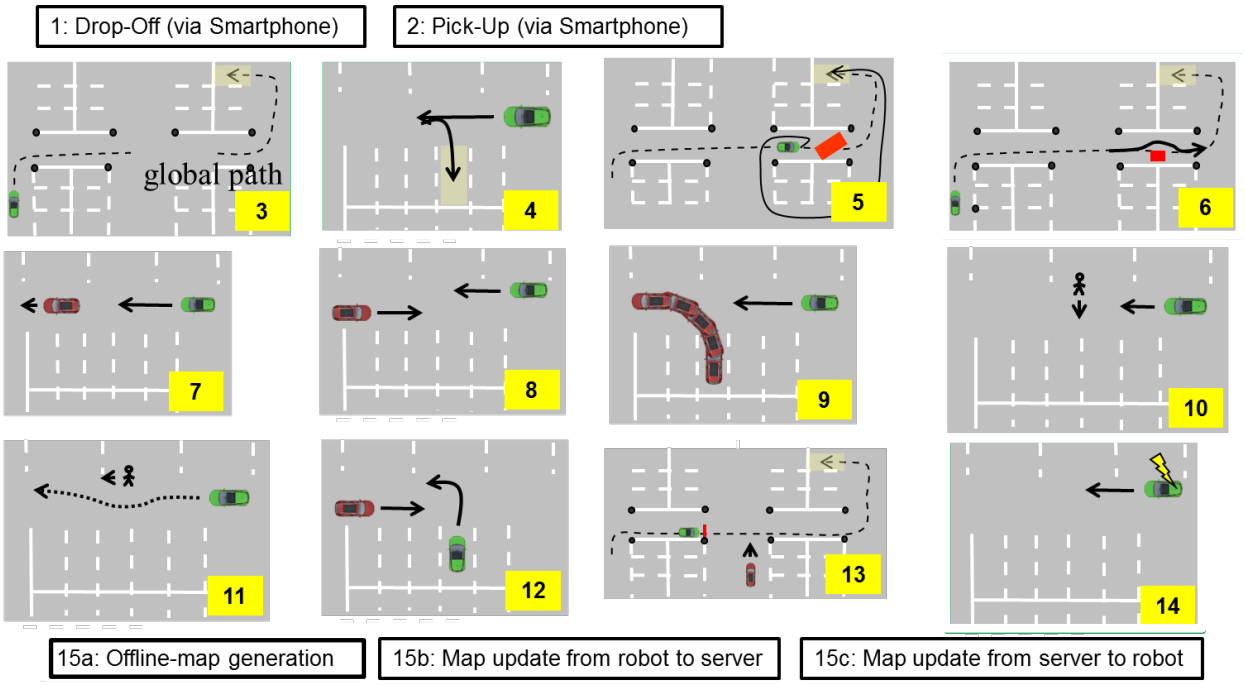
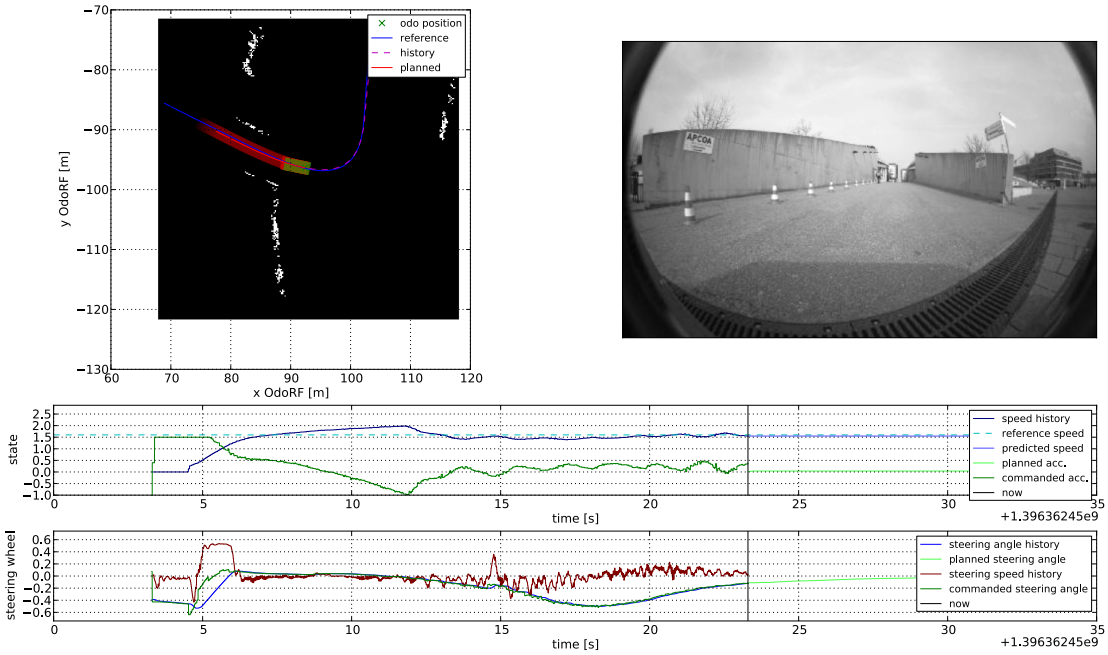


Figure 4: Evaluation scenarios as described in deliverable 6.1

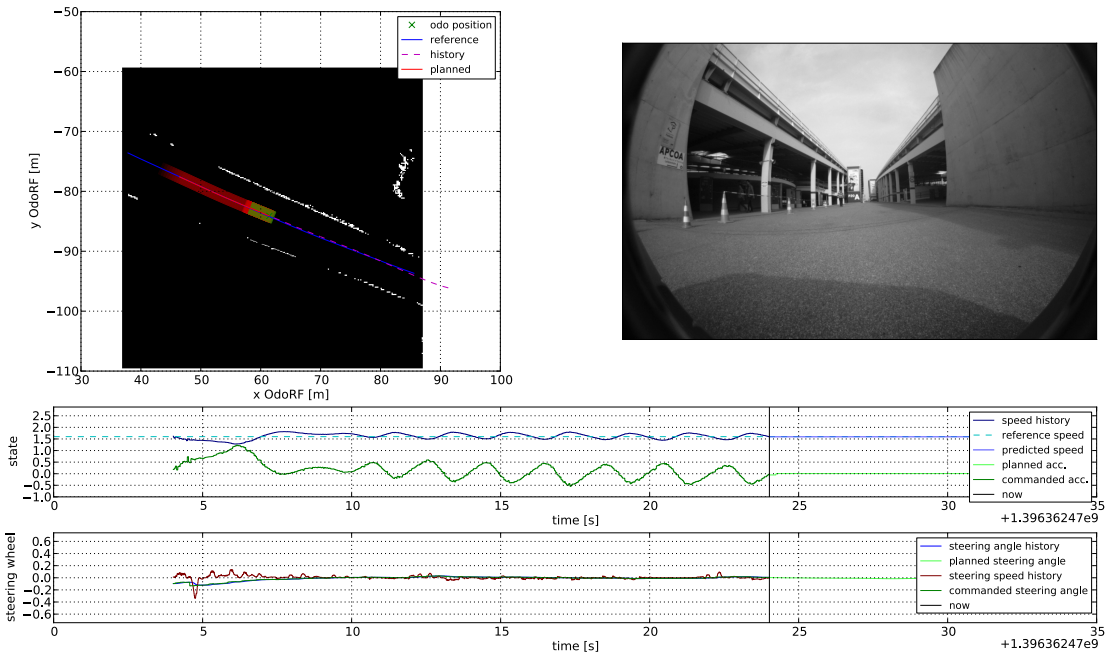
The local planning module does not contain any hard-coded behaviors tailored towards different scenarios except an adaptation to demonstrate scenario 7 to the media in a specific way as explained in section 2.2.2.3.

2.2.1 Scenarios in a Static Environment

2.2.1.1 Scenario 3: Obstacle-Free Navigation Along a Reference Path In the absence of any obstacles in the driving corridor around the reference path planned by the global planning module, the local planner acts as a model predictive controller minimizing speed offsets from a reference speed, lateral offsets from the reference path, steering effort and maximizing the distance to the closest obstacle. This basic navigation task was extensively tested not only in the Bosch parking garage in Stuttgart for over two months, but also at the Mobile Life Campus in Wolfsburg. The vehicle is able to handle inclined ramps as depicted in Figure 5 with the help of the underlying acceleration control loop with integral control (see deliverable 4.2 chapter 3.3). Longitudinal control shows mild oscillations which are however hardly noticeable for a passenger or from the outside. An estimation of the inclination and derivable feed-forward terms for the controller could remedy this issue.



(5.a) Inclined ramp from Stuttgart Messevorplatz to Bosch parking garage.



(5.b) Longitudinal vehicle speed is kept with mild oscillations during uphill navigation at an inclined ramp.

Figure 5: Uphill navigation at an inclined ramp at the Bosch parking garage, Stuttgart.

Narrow passages are dealt with equally well, illustrated in Figure 6. The local planner effectively adapts the lateral offset to maximize the distance to the obstacles in the gridmap via its soft penalization of obstacle

distances, automatically centering the vehicle on the ramp. The reference speed on the ramp is reduced to 1 m/ sec by the global planning stage to yield a more conservative behavior on the ramp.

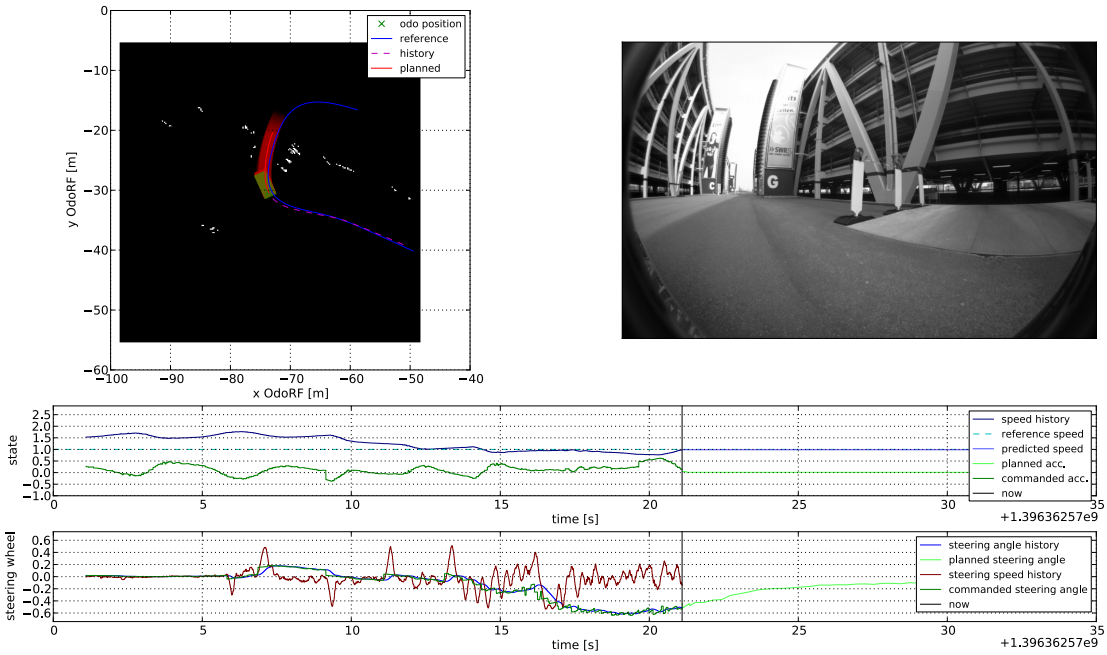


Figure 6: Narrow entry ramp at the Bosch parking garage. The local planner slightly diverges to the left to maximize distance to obstacles.

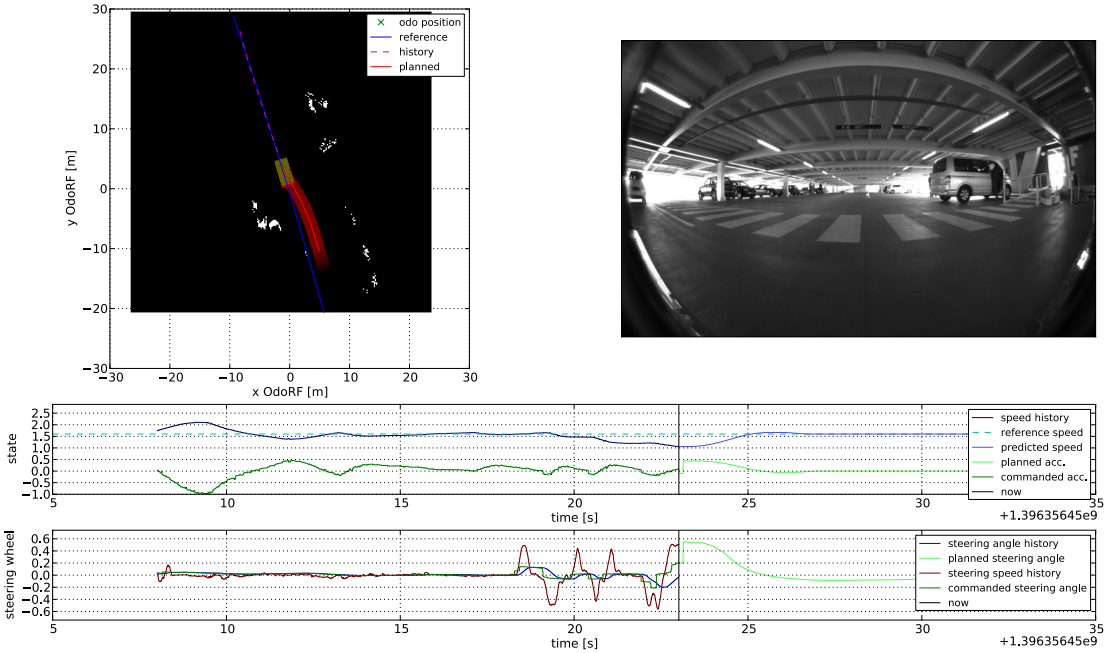
Furthermore navigation is smooth even at very low speeds down to 0.5 m/ sec where control of the combustion engine V-Charge vehicle becomes challenging. The performance of the local planning module fully fulfills the requirements in this scenario.

Problems observed during the performance of this scenario that might prevent the V-Charge vehicle from reaching it's destination are rather large mislocalizations. Though still preserving safety, mislocalizations from the visual localization module will lead to a misplacement of the reference path which serves as a strong guidance for the local planning module. Smaller mislocalizations may be compensated by means of a lateral divergence from the reference path in the presence of obstacles ahead, but as the maximum lateral offset is limited, the capability to compensate is as well. This observation of course holds not only for scenario 3, but is a potential source of failure for all of them. The issue was observed several times during the V-Charge demo preparations at the transition from the outside of the parking garage into it or vice versa due to harsh lighting changes. Upon these situations the V-Charge vehicle continued along its mislocated reference path until it reached an obstacle impossible to pass with a given maximum lateral offset. These mislocalization issues however have been greatly reduced within the localization pipeline and never caused any troubles in the later stages of the preparations for the demonstration.

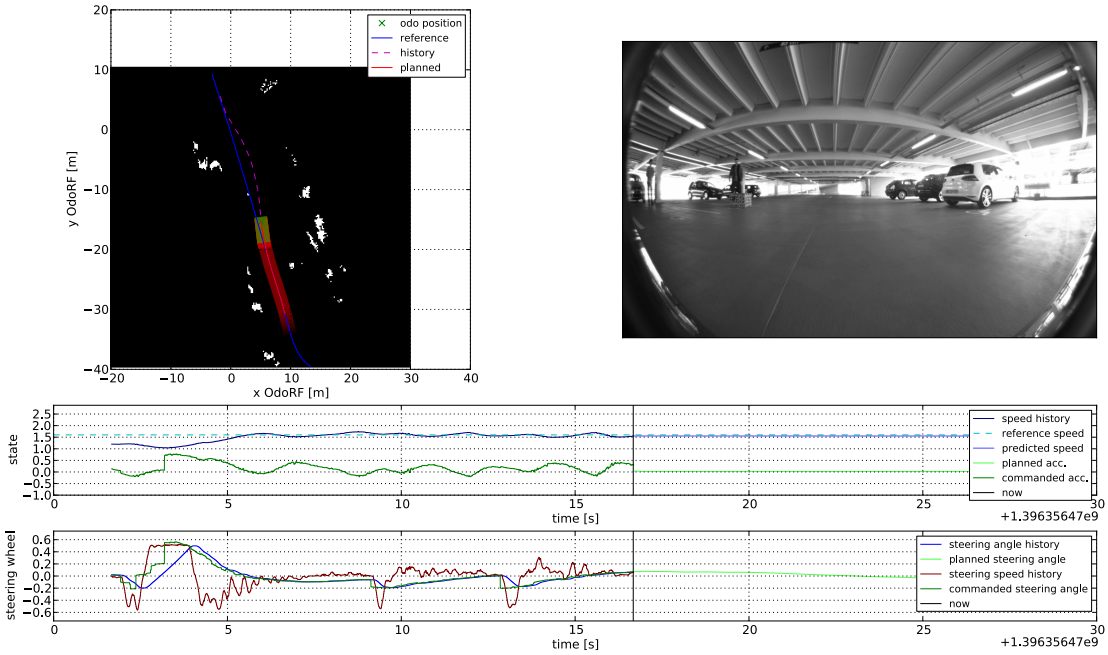
2.2.1.2 Scenario 6: Static Obstacle Handling Passing obstacles appearing in the local static gridmap is considered a basic navigation task. As long as objects are recognized by the stereo sensor, passing static objects is fully reliable. Collision detection within the local planning framework regarding static obstacles is done via an approximation of the rectangular vehicle shape with a collection of equally sized circles and a fast map inflation with the circles' radius (Minkowski sum of gridmap and circle). The amount of circles is

computed to ensure full coverage of the rectangular vehicle shape (ensures safety) and enforcing a maximum overlap to the front and to the sides of 0.4 m and 0.2 m respectively, resulting in an optimal coverage with 10 circles. The minimum amount of circles needed given the constraints on maximum overlap is dynamically computed.

Figure 7 depicts a maneuver passing a static pylon placed on the vehicle's lane. The vehicle navigates safely around the obstacle and returns to its reference path. This behavior is highly repeatable and reliable and fulfills the requirements in this scenario.



(7.a) The pylon in the center of the lane is detected at an early state and the local planner is able to react early.



(7.b) The pylon is passed and the local planner returns to the reference path.

Figure 7: Passing a static pylon in Bosch parking garage, Stuttgart.

2.2.2 Scenarios in a Dynamic Environment

This section outlines the approach taken to cope with dynamic elements in the scene and illustrates the performance of the V-Charge system operating in such. It will be highlighted that present shortcomings arise from current limitations in perception as well as the negligence of interaction and uncertainty in the planning stage. The latter shortcoming is being tackled by our current research in progress.

2.2.2.1 Collision Detection The V-Charge setting poses two challenging requirements on the planning stage regarding the handling of dynamic objects:

- Object shape recognition is extremely challenging due to the difficult separation of closely parked vehicles lining the ego vehicle's lane. Shapes are often not properly separated as illustrated in figure 8 (leftmost object). Using e.g. circular approximations of the objects for faster collision detection is unfeasible in this case. Thus the collision detection approach has to be able to work with oriented bounding boxes, which renders naive approaches based on multiple interference tests impracticable.
- The amount of extracted dynamic objects from the scene can be large in cases where the V-Charge vehicle has to drive past lines of parked cars.

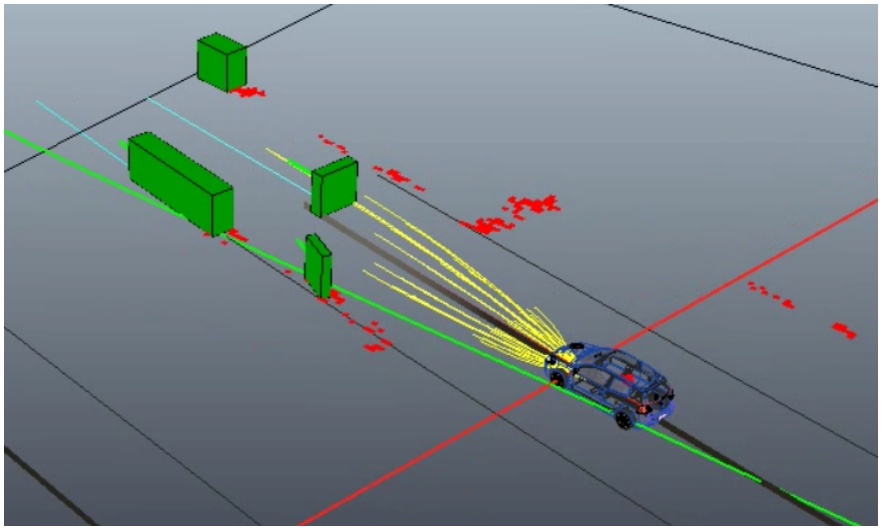


Figure 8: Output of the dynamic object detection and tracking module during navigation along a lane lined by orthogonally parked cars. Often adjacent vehicles are merged into one spaciously extended shape (leftmost object).

Therefore a collision detection scheme commonly applied in game engines was adopted. Our approach to fast collision detection is described in the attached preliminary paper A, to be submitted in similar form to an upcoming conference. It describes the usage of a bounding volume hierarchy data structure in workspace-time space to accelerate collision queries for the local planner's candidate trajectories. It allows for arbitrary motion predictions of other agents and scales well with the number of dynamic elements in the scene. The approach was fully integrated into the V-Charge system and tested during numerous real-world test runs.

As the attached paper does not explicitly focus on the implementation within the V-Charge system, the settings of the dynamic collision detection approach are highlighted in the following. As the V-Charge local planner constructs its candidate trajectories on the fly without approximations or discretizations and the candidate trajectories are dependent on the shape of the reference path, precomputations of the accelerating bounding volume hierarchy data structures are not possible. Therefore we integrated the approach labeled as "Isolated Configuration Queries" without utilizing convex hull representations in workspace-time space. Computing the tree structures and convex hull representations for every candidate trajectory would exceed the computation time available during a planning cycle (100 ms - 200 ms).

2.2.2.2 Adaptations Due to Current Limitations in Perception The local planning framework assumes known motions of surrounding agents in the future (motion predictions). This renders the motion planning problem feasible for the sampling-based approach by neglecting the effect of the ego vehicle's plan on the other agents, but rises the necessity for good estimation of not only the other agent's position but also dynamic information about them such as velocity. While position estimates originating from the classification and tracking pipeline were impressively good, incorporating dynamic information from it was problematic in the local planning stage for two cases. Firstly ghost objects with fairly high velocity estimates occured from time to time due to repetitive elements misclassified as vehicles in the parking garage and temporal mismatches between them. Secondly the noise level of the velocity estimates for slow and stationary objects (parked cars on the side) were significant. This can be attributed to the difficult separation of objects lining the lane. The latter issue degraded navigation performance to an unsatisfactory level when passing between rows of parked cars, each of them appearing to start driving into the ego vehicle's path. As a temporary countermeasure, the velocities of all dynamic objects–except a potential vehicle that was selected as a target

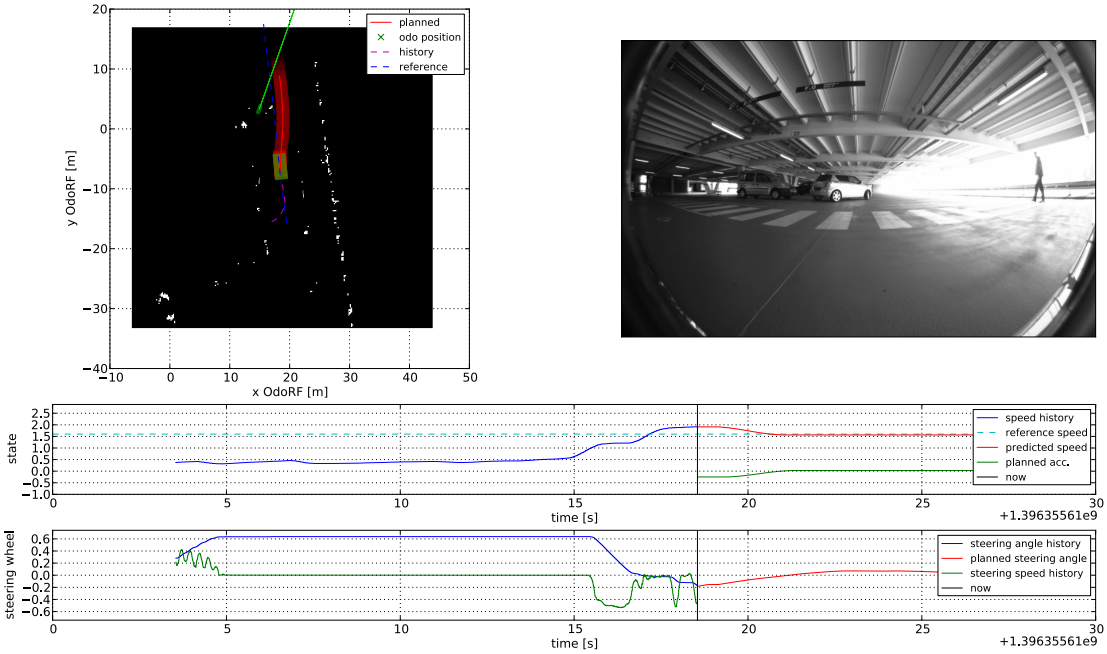
vehicle (platooning))–were assumed to be zero. The navigation performance in the platooning scenario shows the capabilities of the general approach with dynamic elements in the scene. Full motion predictions of all traffic participants will be re-enabled when the estimation of their dynamic information becomes mature.

We utilized a constant translational velocity prediction model for the target vehicle. As the estimates of other states of the dynamic objects become adequate, the collision detection framework within the local planner is readily prepared to incorporate e.g. acceleration and rotation rate.

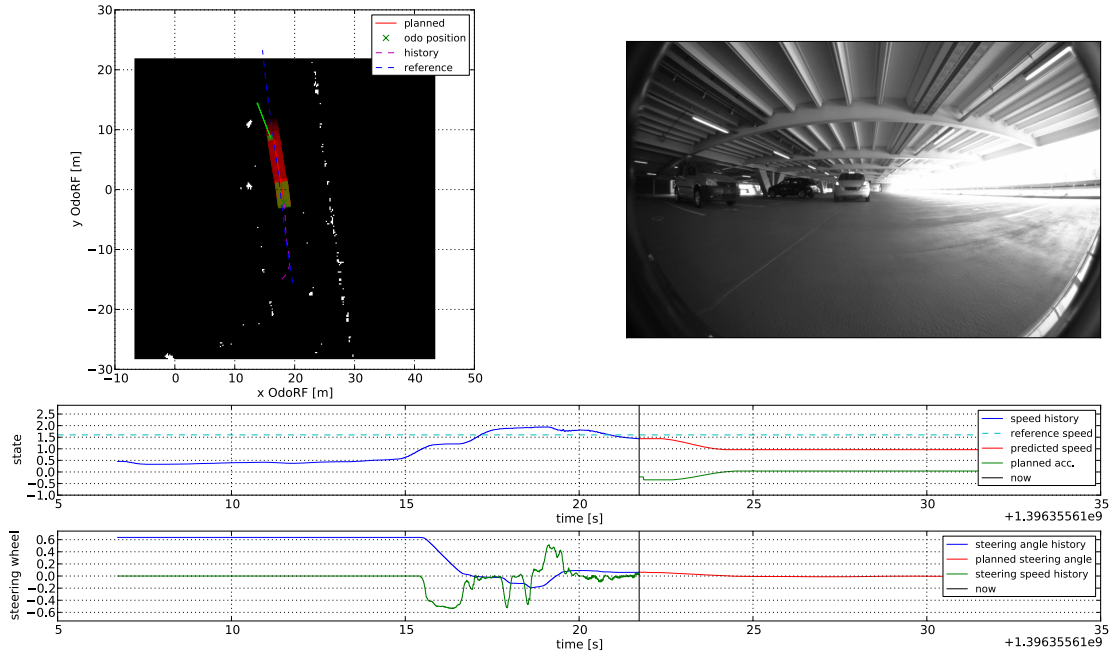
2.2.2.3 Scenario 7: Platooning Driven by the requirements for the V-Charge press demo, a smooth platooning mode was implemented. The scene presented to the media should depict the approach towards a stationary vehicle on the lane, whereupon the ego vehicle should–in the same manner–come to a standstill. As the target vehicle continues, the ego vehicle is supposed to follow the target within a safe distance.

Without any changes to the local planning framework, the ego vehicle would simply pass the parked car on the lane if space allows. To avoid this behaviour, a mechanism was implemented that selects at most one of the classified cars in the scene as a target vehicle and adapts the reference speed for the planner in order to yield a certain time-gap or minimum distance. The selection of the target vehicle was limited to a corridor around the reference path of the ego vehicle and an anisotropic distance function in the Frenet frame of the reference path was applied. The distance function was amplified along the lateral direction to increase importance of a small lateral offset in the matching. After the selection of a potential target vehicle, the local planner adapts its reference speed to reach a desired minimum time-gap or minimum distance.

An exemplary situation for the platooning scenario is shown in Figure 9. It reveals an additional problem: While the object track is not yet stable, artifacts originating from the dynamic object remain in the static map as it can be seen in the gridmap in figure 99.a. The local planner is forced to react to the static objects, which results in undesirable steering motions upon first contact.



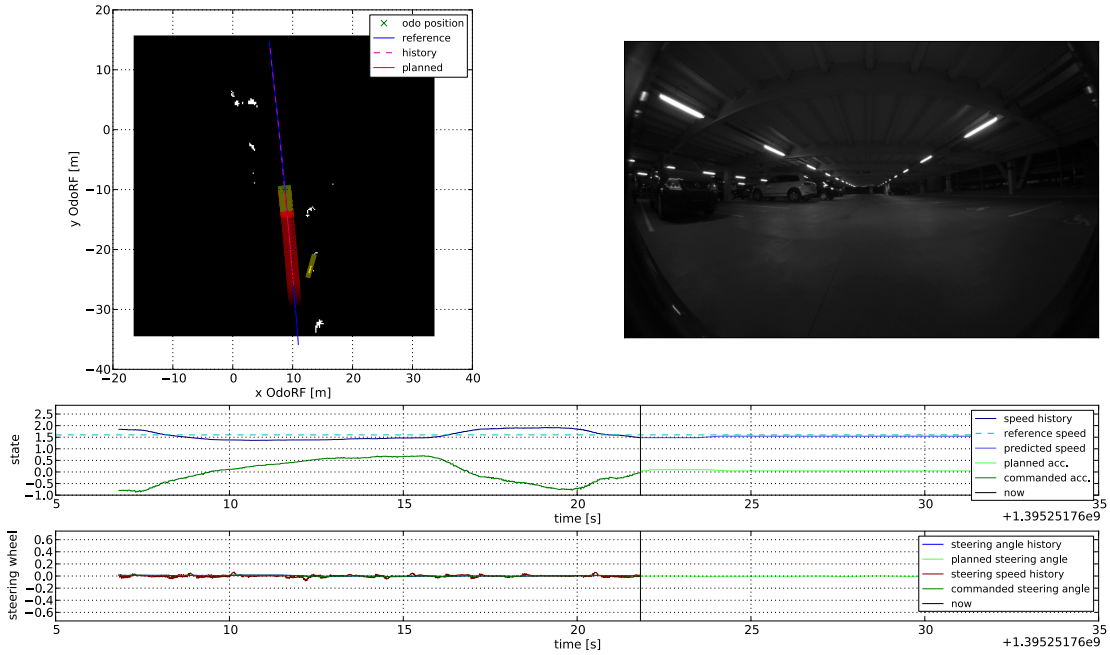
(9.a) First approach to the target vehicle for a platooning scenario after park-out. Parts of the vehicle pulling out are entered in the static gridmap. The local planner necessarily reacts on these static objects and pulls to the left.



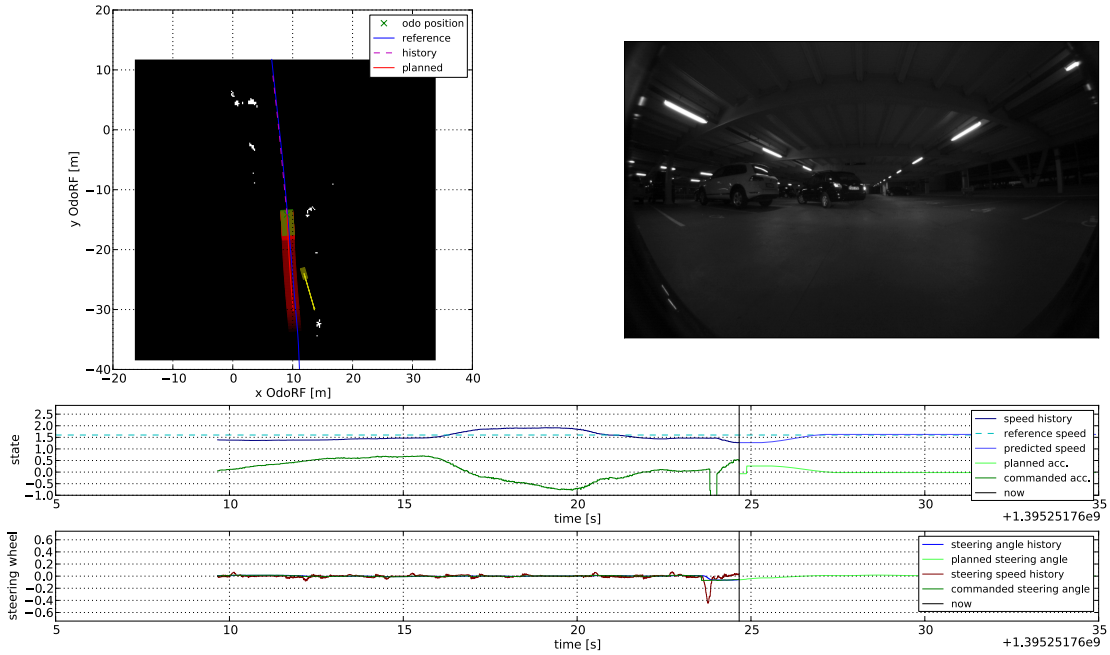
(9.b) After a short time, the dynamic object track is stable and no artifacts remain in the gridmap. The local planner adapts its reference speed based on distance on relative velocity to the target vehicle and diverges from the global reference speed (cyan dotted line in middle plot). The minimum of the local planner's objective function is shifted and no undesired lateral avoidance maneuver takes place.

Figure 9: Platooning mode in Bosch parking garage, Stuttgart. The ego vehicle smoothly follows the target car.

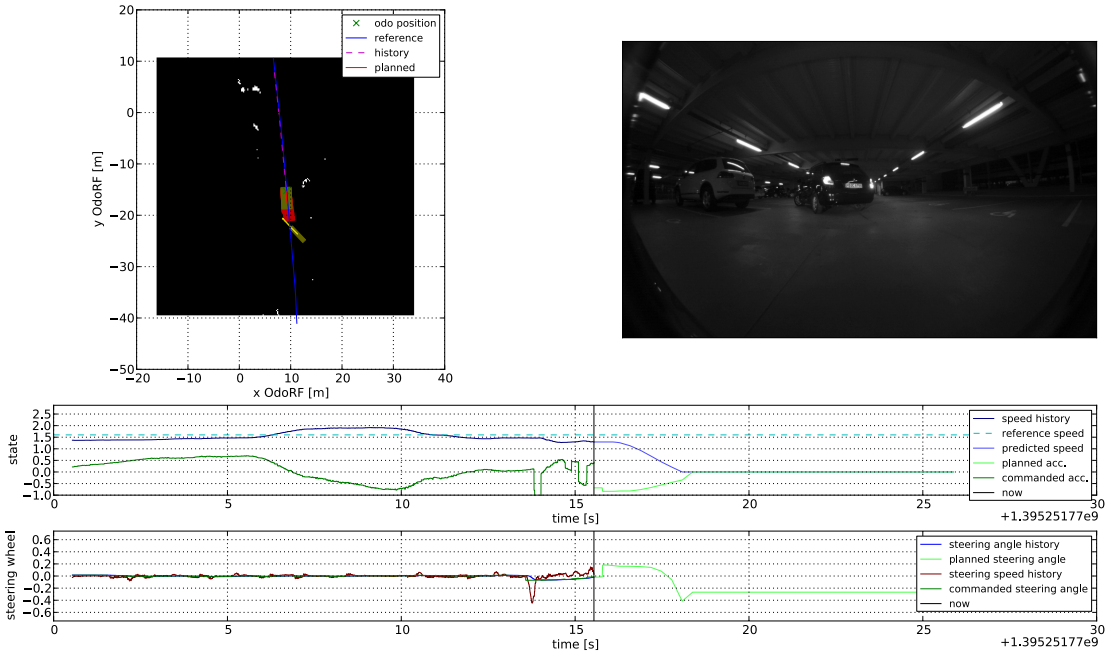
2.2.2.4 Scenario 9: Another Car Pulls Out Item 9 in Figure 4 describes the scenario where a parked car starts to pull out of its parking spot while the ego vehicle approaches. Figure 10 illustrate the ego vehicle's behavior in this scenario: Though utilizing a simple constant velocity prediction model and having to deal with improper velocity estimates of the vehicle pulling out, the reaction of the ego vehicle is appropriate (however late). The late reaction in this case stems from a wrongly estimated velocity vector (pointing away from the ego vehicle instead of towards it), that lets the V-Charge vehicle assume there will be more room for navigation in the future.
It is apparent that linear velocity predictions do not model the movement of the car parking out well in this scenario. However estimates of the vehicle's rotation rate were too noisy to be used directly. Incorporating information about the road-network into the prediction step of other agents' movements will clearly boost the performance of the local planning stage.



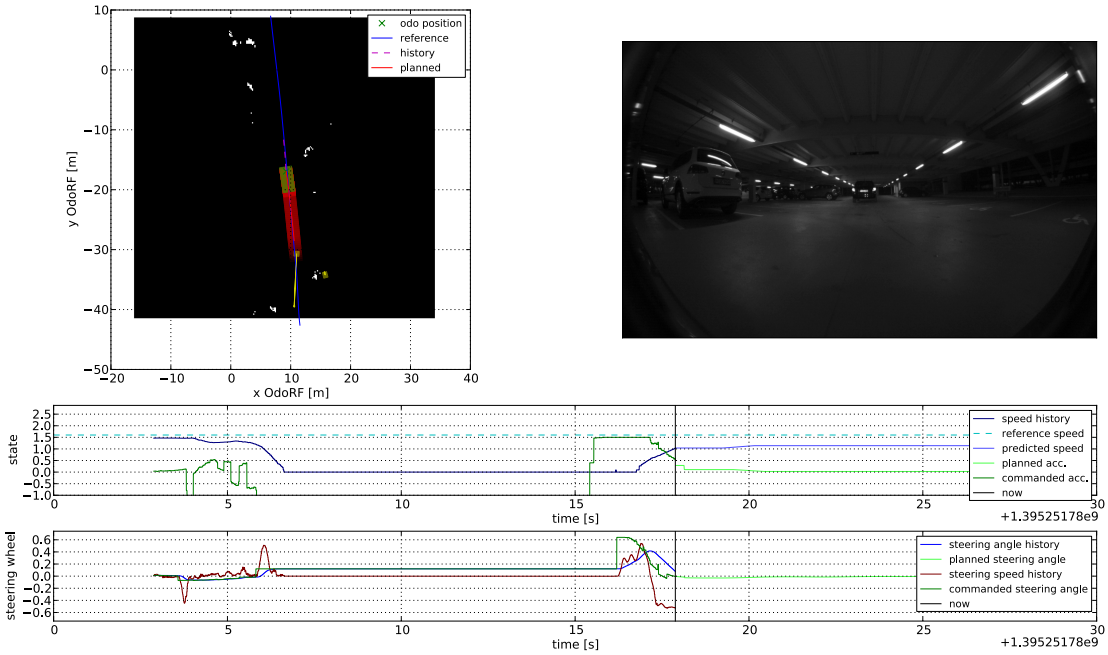
(10.a) Initial detection of the vehicle pulling out. No reaction of the local planner due to the zero-speed assumption on vehicles outside a corridor around the reference path.



(10.b) Minimal reaction due to a wrongly estimated velocity vector of the vehicle pulling out of his parking spot (vehicle is actually moving backwards). With a constant velocity prediction over time, the vehicle will not be blocking the path in the future.



(10.c) A harsh braking maneuver is issued as soon as the velocity vector pointing towards the ego vehicle is correctly estimated.

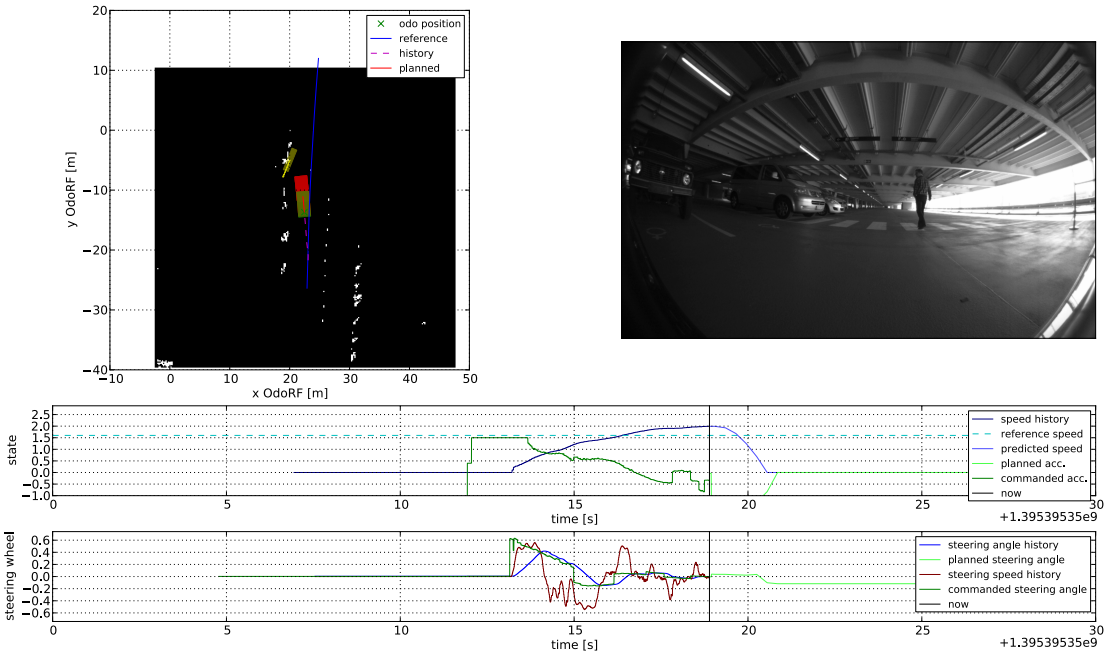


(10.d) As the vehicle pulling out is driving away, the ego vehicle accelerates in platooning mode.

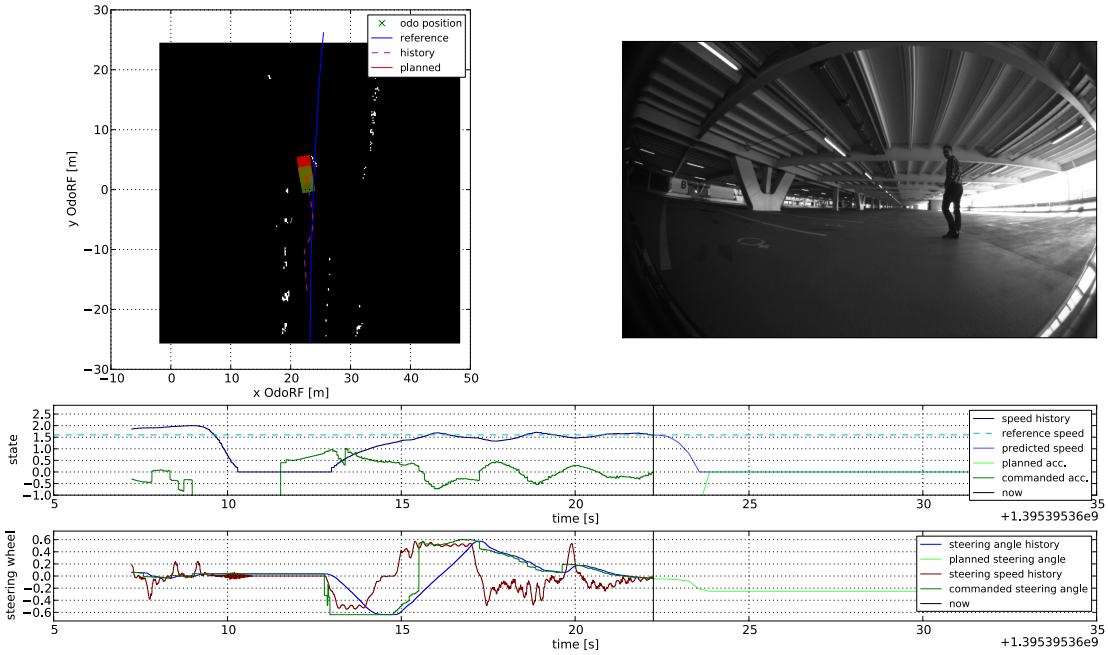
Figure 10: Encountering a vehicle pulling out of its parking spot in Bosch parking garage, Stuttgart.

2.2.2.5 Scenario 10+11: Coping With Pedestrians At the time of preparation of this document, a module for pedestrian detection and tracking was not readily available. Consequently, pedestrians are detected by the front stereo sensor and entered in the static occupancy grid. The static occupancy grid lacks information about velocities of objects, resulting in a degraded navigation performance when dealing with moving entities not classified as such in the scene. We show the behavior of the V-Charge vehicle in this situation for the sake of completeness of this report, but want to highlight that the system is not yet able to tackle this scenario due to missing inputs to the planning stage.

Figure 11 depicts a situation in which a pedestrian is walking randomly in front of the car, constantly changing sides. The navigation performance in the presence of the swerving pedestrian is rather poor, involving frequent switches from an avoidance maneuver from left to right and vice versa. This underlines the necessity for a classification and tracking module for pedestrians such that velocity information for pedestrians is available to the planning stage.



(11.a) The local planner has to issue a braking maneuver. The pedestrian was pushing it to the left, due to an earlier choice of the local planner on the topology.



(11.b) Within the next seconds, the local planner swerves (dotted magenta line)) to avoid the pedestrian constantly changing sides. Finally the pedestrian gets too close and a braking maneuver is issued. A portion of the grid cells labeled as occupied remains in the map since they went out of the field of view. This prevents the local planner from continuing his travel.

Figure 11: Vehicle/pedestrian interaction in Bosch parking garage, Stuttgart.

2.3 Off-Lane Planner (Bosch)

This section will report on the current status of the off-lane planner integrated in the V-Charge vehicles. The off-lane planner is responsible for motion planning in unstructured environment, including maneuvers requiring driving direction changes. Expected application of this planner is to park in and out of a parking spot (scenario 4 defined in deliverable 6.1, see Figure 4) as well as providing U-Turn maneuvers in case a path is blocked by obstacles (scenario 5). Though potential of the off-lane planner for applications in various scenes, parking situations have been focused on up to now (Fig. 12). Within this deliverable, we evaluate the performance in such application.

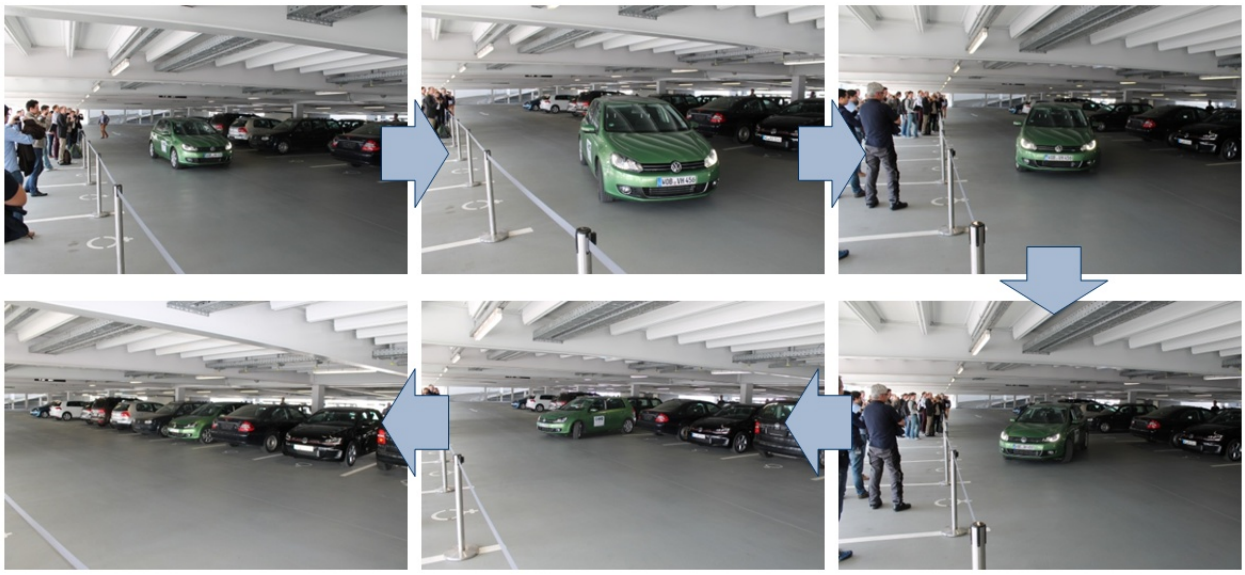


Figure 12: Parking at V-Charge demonstration event on 02.04.2014 in "Bosch Parkhaus", Stuttgart. *Top-Left*: Seamless handover to the off-lane planner. *Top-Center to Bottom*: The off-lane planner leads the vehicle into a vacant parking spot assigned by the parking lot manager.

2.3.1 Justification for On-Lane/Off-Lane Module-Separation

On-lane navigation reveals various differences to the parking application, which motivated the separation of these two approaches.
Firstly, during on-lane navigation, lane information gives a strong guidance for the planning stage, allowing for an optimized implementation. In the absence of lane information–as it is the case for parking maneuvers–search for the optimal path/trajectory becomes more complex as it might involve multiple turn maneuvers. Secondly, as space for navigation is usually tight for parking maneuvers, this poses increased demands on the accuracy of the collision detection scheme.

The above mentioned facts led to the decision of utilizing separate approaches for the two tasks at hand. Due to the overall high integration effort requirements of the project, the two approaches have not been combined so far. Yet joining the two approaches in a hierarchical manner does not pose a problem in general. The local planning framework is in general able to take reference path inputs from any module, including the off-lane planner. This approach would give the off-lane planner access to the local planner's model predictive control features as well as it's ability to incorporate predictions of moving entities in the scene in a principled manner. As incorporating dynamic objects becomes especially important during park

out maneuvers (approaching cars from left and right) but object detection from pure monocular images in the vehicle's side mirrors is not readily available at the moment, we will consider the combined approach at a later stage.

2.3.2 Integration Advancements

Besides numerous improvements in detail, the most visible advancement for outside observers of the V-Charge vehicle is the smooth handover between the local planner and the parking planner. Via a special response message to the mission executive, each task processor can signal its readiness to start processing the next task in the sequence. Consequently tasks can seamlessly be executed without interruptive behavior.

The parking planner starts to search a path leading into the designated parking spot while driving on a lane and signal to the mission executive when a collision-free path is found. The vehicle is safely controlled by the local planner until a collision-free path for parking is on-hand. Then, the mission executive assigns a task to the off-lane planner and the off-lane planner takes over the control till the vehicle finishes parking procedure.

2.3.3 Performance of the Off-Lane Planning Module: Success Rate

Data sets used for the evaluation of the off-lane planning module were recorded during test drives in the "Bosch Parkhaus" in Stuttgart Germany during the preparations for the V-Charge demonstration event on 02.04.2014. For this report, parking maneuvers which targets to park into parking spots (not charging stations) assigned by the parking lot manager are evaluated.

In these maneuvers, poses of assigned parking spots are defined in global coordinates and fed to the off-lane planner as target poses for path-planning by the parking lot manager. As there are two cars parked on the neighboring parking spots, the off-lane planner is required to adjust the target pose based on actual perception information to lead the vehicle in the middle of neighbouring cars.

The success rate for evaluated parking maneuvers is 64.7%. This success rate is a conclusive performance for the whole V-Charge system and not attributed to the off-lane planning module alone. In the following subsections, issues that prevented successful execution of the parking maneuver are explained and countermeasures for eliminating such will be discussed.

In Fig. 13, a typical success case is shown. The off-lane planner starts to search a path for parking while driving on a lane (plot on the left). The target pose for the path is assigned by the parking lot manager. When a collision-free path (blue line) is found, the off-lane planner requests the mission executive to let it take the control. Then, the mission executive assigns a task to the off-lane planner and it starts to take control. During parking, the off-lane planner adjusts the target pose based on actual perception information to lead the vehicle to the middle of two neighbouring cars (plot on the right). Then, the path is also adjusted to reach the new target pose (green line).

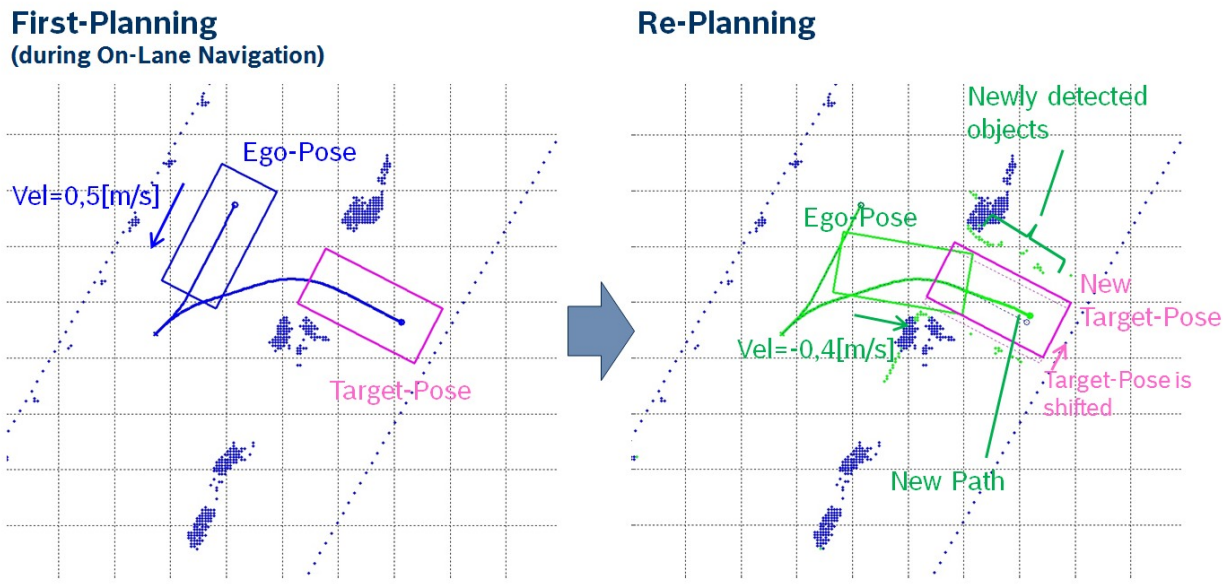


Figure 13: A typical success case of parking maneuvers. *Left*: "First-Planning" while driving on a lane. *Right*: "Re-Planning" corresponding to parking spot adjustment

2.3.4 Overview of Sporadic Issues Hindering Successful Parking

Extensive analysis of recorded log files revealed several issues which can be classified into two categories: perception and kinematics.

- **Perception**: Parking tasks are aborted due to false-positives in the gridmap. False-positives in the targeted parking spot label such as non-approachable, hence rendering a parking maneuver impossible. As the V-Charge project is utilizing low-cost sensors only, issues in perception are foreseeable. Work-package 3, Task 3.2 continuously works on the improvement of the robot-centric gridmap which has showed great progress throughout the project. Recent efforts to integrate ultrasonic data have been undertaken, yet the fusion of the various sources of (noisy) perceptive sensor data has not reached the status of completely robust operation yet.
- **Kinematics**: The V-Charge vehicle is not able to closely follow paths provided by the off-lane planner. An estimation of the vehicle's minimum turning radius performed after the V-Charge demonstration with highly accurate GPS+inertial data revealed a mismatch between the vehicle's specifications and the real platform. Consequently the vehicle was not able to perform sharp turns as expected at the planning stages, resulting in large deviation between a path and the vehicle which will require multiple re-planning phases.

2.3.5 Category "Perception": False-positives

There are two types of false-positives to be observed: "Clutter" and "Single-Pixel" (see Fig. 14).

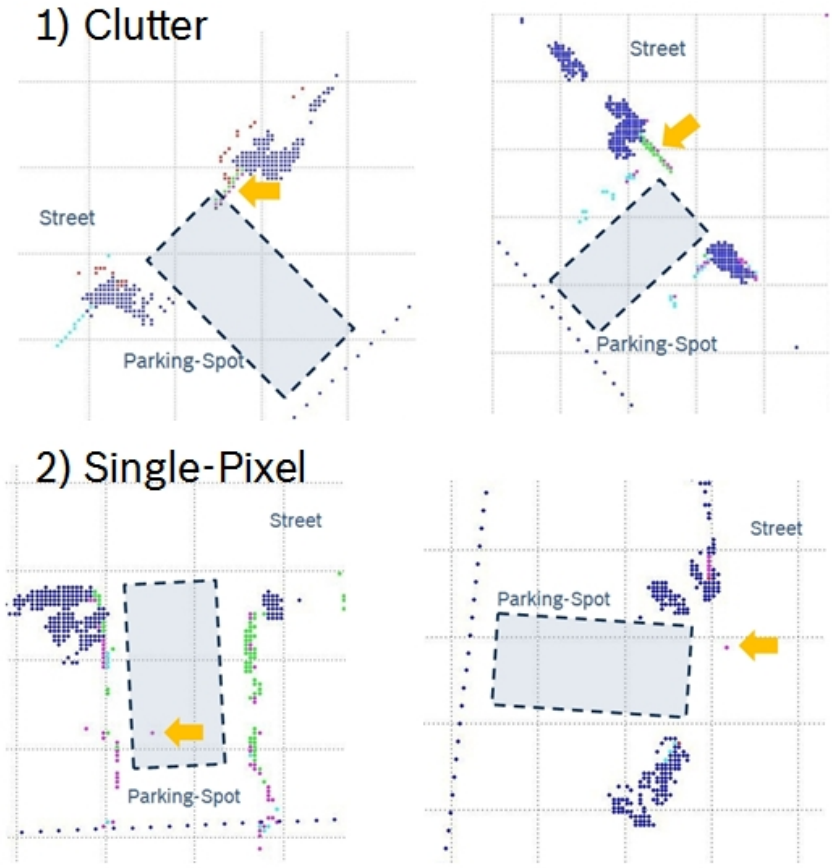


Figure 14: Example of "clutter" and "single-pixel" false-positives originating from the perception pipeline.

For both types of perception errors, the off-lane planner will react with one of the following behaviors. First, if any false-positive is detected directly in or in front of the designated parking spot that prevents the planner from acquiring a collision-free path, the off-lane planner reports an error back to the mission executive and aborts the parking task into the designated spot. Currently ongoing implementation efforts target the re-planning in the high-level parking lot management upon this event. In the future, the V-Charge vehicle will be sent to a new vacant spot in case the off-lane planner reports a blockage of the currently assigned spot. The second case is a situation where it is still possible to find a collision-free path with avoiding falsely detects obstacles at a re-planning. This path might differ drastically from the previous plan, resulting in non-intuitive driving behavior for an external observer.

2.3.6 Category "Kinematics"

During test drives, rather large deviations between ego position and the provided path were observed during tracking. Especially, such deviations were significant on paths with a large curvature. It was observed from recorded data that though applying a maximum steering wheel angle, the vehicle's tracking error constantly increased in segments of the planned path containing turning radii close to the platforms limitations (minimum turning radius based on specifications: 3.49 m). Figures 15 and 16 display a parking maneuver with a requested minimum turning radius at the vehicle limit.



Max-Deviation	Nr of Cases
< 10 [cm]	2
10 – 30 [cm]	6
> 30 [cm]	6

Figure 15: Statistic for deviations between ego positions and paths provided by the off-lane planner (on odometry coordinates) based on analysis for 14 cases

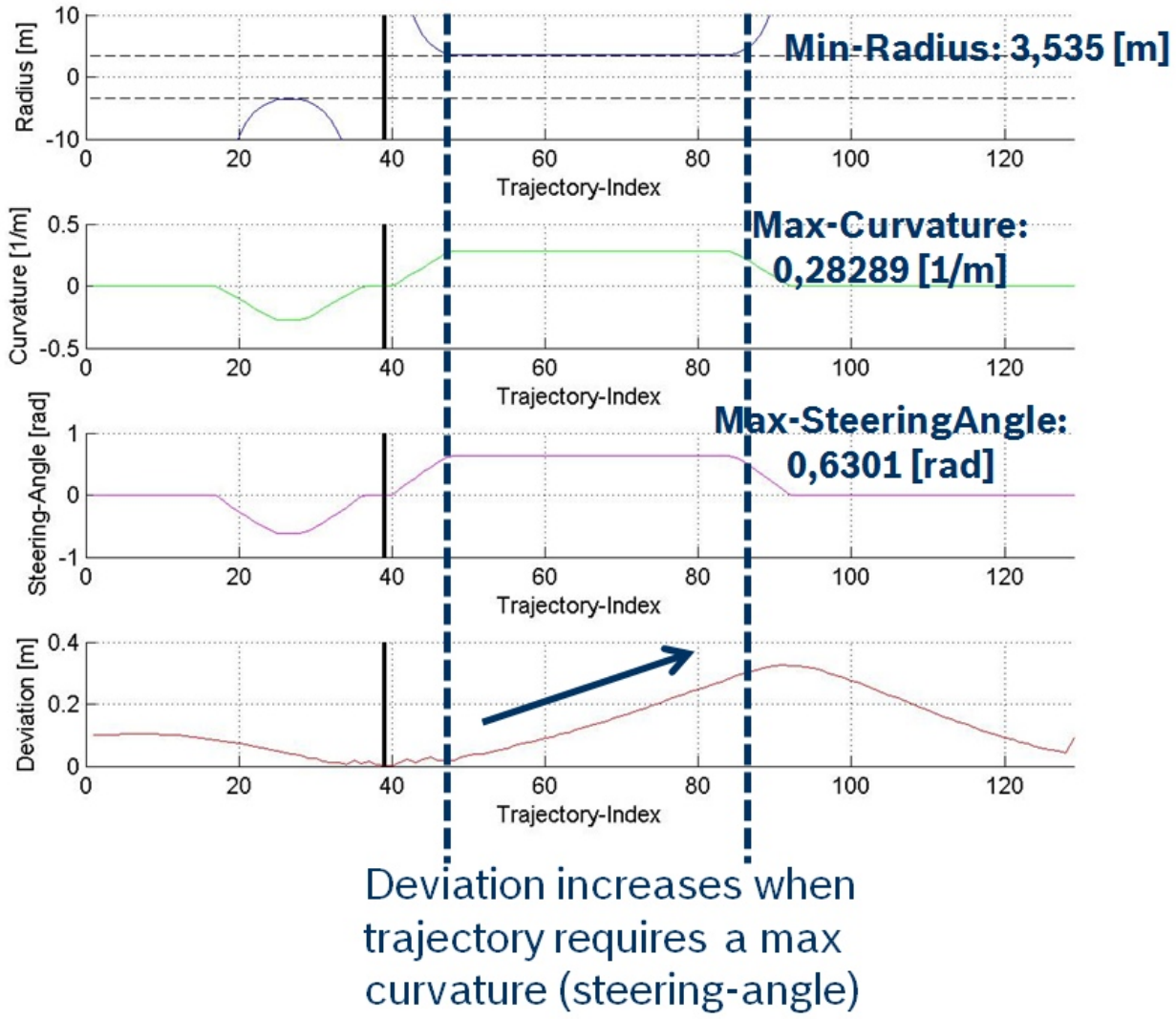


Figure 16: Requested turning radius, curvature and steeringwheel angle of a trajectory during increasing tracking error

The tracking error constantly increases whenever a small turning radius close to the vehicle's limitations (about 3.50 m in this example) is requested. In Fig. 17, requested and actual steeringwheel angles are displayed for the same parking maneuver as shown in Fig. 16. It shows that the trajectory required steering wheel angles at the platform's limit and the vehicle indeed applied the maximum steeringwheel angle. Based on this observation, it is concluded that the test vehicles (VW Golf6) cannot turn with a radius of 3.50 m though a minimum turning radius for this type of vehicle is defined as 3.49 m.

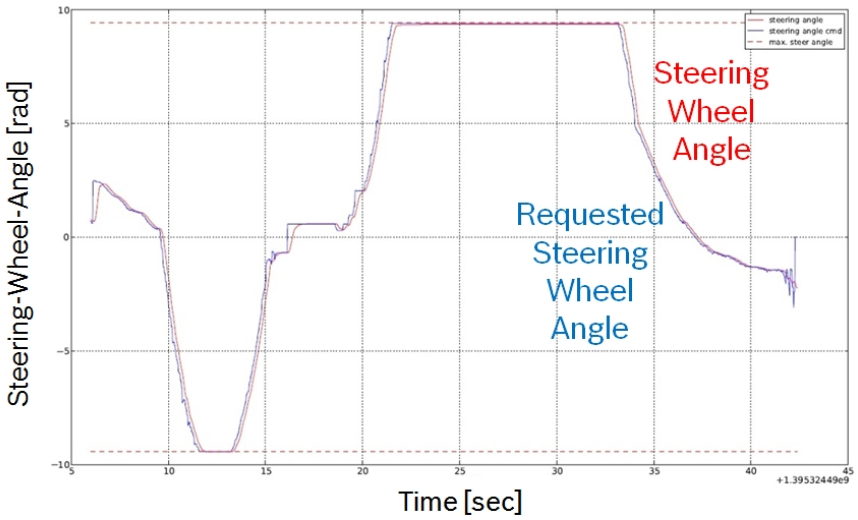


Figure 17: Requested and actual steeringwheel angle

An estimation of the vehicle's minimum turning radius was performed after the V-Charge demonstration with highly accurate GPS+inertial data. It revealed indeed a larger minimum turning radius than specified. We are confident that this finding and the elimination of this issue (adapted control set for the state-lattice search in the off-lane planner) will improve the success rate of parking maneuvers.

3 Additional Efforts

3.1 Automatic Summaries of Test Runs (ETHZ-ASL)

As the amount of recorded data from test runs increases, it gets harder to find specific information in the test logs. Up to now several terabyte of data were uploaded to the webshare.

Therefore a tool was written which automatically parses uploaded recordings from the V-Charge vehicles and creates informative summaries of the data. Different levels of detail are provided, e.g. simple overview of events during the run (see Figure 18), detailed information about data rates and delays (Figure 19) or an evaluation of the parking process (Figure 20).

Task Processor Responses

Stream	Num Samples	Errors	dt (mean +- std [min -- max]) ms				delay (mean +- std [min -- max]) ms				link
AslLocalPlanner	1412	0	231.751 +-	191.201 [	0.665 --	1000.129]	0.083 +-	0.526 [	0.000 --	11.527]	link
VisualLocalization	644	0	508.288 +-	441.299 [	0.000 --	1005.101]	0.216 +-	0.358 [	0.000 --	2.580]	link
RbParkPlanner	657	0	498.196 +-	442.838 [	0.000 --	1003.200]	0.363 +-	0.621 [	0.000 --	9.587]	link
BasicTaskProcessor	369	0	891.304 +-	292.538 [	19.197 --	1000.190]	0.071 +-	0.692 [	0.000 --	11.998]	link

Ego Stream Evaluation

Ego Stream	Num Samples	Num Jumps	link
vehicleodometry_with_acceleration_out	8092	0	link
VisualLocalizationEgo_out	16413	10	link
vehicle_odometry_out	14467	0	link

Emergency Brake Evaluation

# emergency brake events	link
898	link

Gateway Evaluation

# gateway error events	# brake pedal hit events	# door open events	link
0	0	0	link

Task List Evaluation

task list id	task index	task type	started	finished	duration [s]	start pose	end pose
535	0	WaitForDropoff	N/A	N/A	N/A	N/A	N/A
536	0	StartVisualLocalization	2014-04-01 16:26:52.836525	2014-04-01 16:27:26.435317	0.00:0.00:33.598792 [hh::min::sec]	link	link
538	0	ActivateVehicle	2014-04-01 16:27:28.638758	2014-04-01 16:27:29.834997	0.00:0.00:1.196239 [hh::min::sec]	link	link
538	1	FollowTrajectory	2014-04-01 16:27:29.834997	2014-04-01 16:28:07.636405	0.00:0.00:37.801408 [hh::min::sec]	link	link
538	2	FollowTrajectory	2014-04-01 16:28:07.636405	2014-04-01 16:28:39.439284	0.00:0.00:31.802879 [hh::min::sec]	link	link
538	3	FollowTrajectory	2014-04-01 16:28:39.439284	2014-04-01 16:29:12.240864	0.00:0.00:32.80158 [hh::min::sec]	link	link
538	4	FollowTrajectory	2014-04-01 16:29:12.240864	2014-04-01 16:29:45.442883	0.00:0.00:33.202019 [hh::min::sec]	link	link
538	5	FollowTrajectory	2014-04-01 16:29:45.442883	2014-04-01 16:30:31.444484	0.00:0.00:46.001601 [hh::min::sec]	link	link
538	6	FollowTrajectory	2014-04-01 16:30:31.444484	2014-04-01 16:31:04.846193	0.00:0.00:33.401709 [hh::min::sec]	link	link
538	7	FollowTrajectory	2014-04-01 16:31:04.846193	2014-04-01 16:31:35.848122	0.00:0.00:31.001929 [hh::min::sec]	link	link
538	8	Park	2014-04-01 16:31:35.848122	2014-04-01 16:32:10.649701	0.00:0.00:34.801579 [hh::min::sec]	link	link
538	9	DeactivateVehicle	2014-04-01 16:32:10.649701	2014-04-01 16:32:13.650001	0.00:0.00:3.0003 [hh::min::sec]	link	link

Figure 18: Overview page of the automatically created HTML mission logs

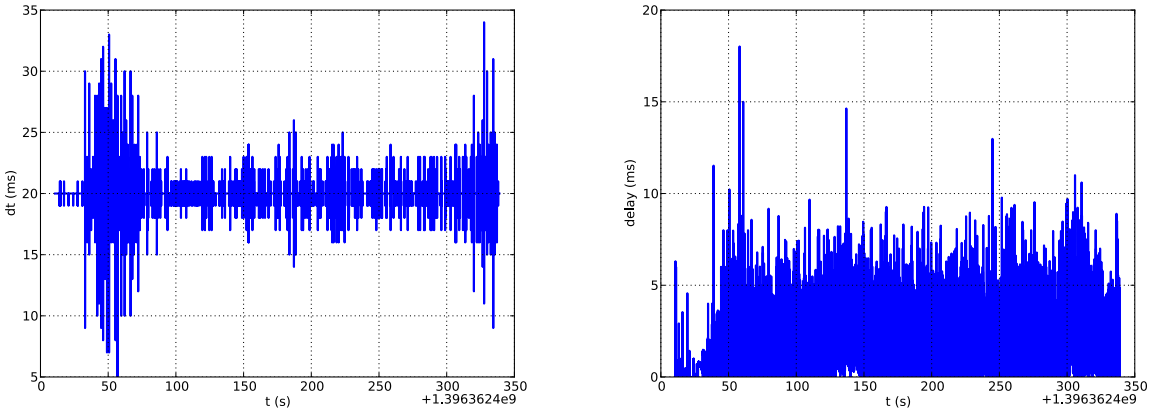


Figure 19: Message rate (left) and delay (right) overview plots of single data streams created by the mission parser.

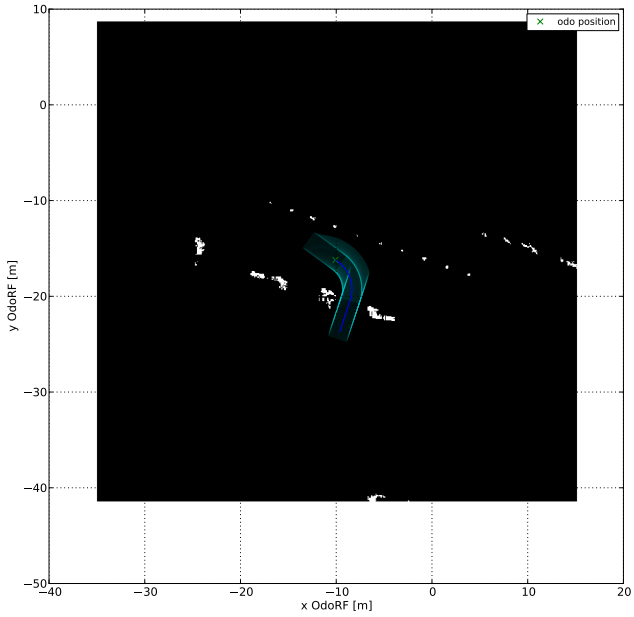


Figure 20: Detailed information of a parking process automatically created by the mission parser.

4 Conclusion

In this deliverable we gave an overview on scientific and implementation progress related to WP 4 as well as an evaluation of the V-Charge planning modules in extensive real-world tests.

We showed that the system is able to reliably navigate in a static environment. Remaining issues degrading the performance of the system during parking were identified:

- Sporadic false-positives in perception during park-in procedures.
- A mismatch between the specifications of the platform's minimum turning radius and the real platform. A new set of parameters is already obtained at the time of preparation of this deliverable.

An approach able to cope with a fairly large amount of dynamic objects in the scene that makes no assumptions on the predicted motions or the objects' collision shapes was developed and tested on the real platform. The local navigation module is readily prepared to incorporate arbitrary motion predictions of all objects in the scene. Yet perception-side limitations to be tackled in the future required certain preliminary counter measurements within the local planning module.

We believe that modules developed in the context of workpackage 4 are on-track regarding their performance in real-world operation. They are well integrated on the V-Charge vehicle and enable reliable, safe automated driving.

A Fast collision detection through bounding volume hierarchies in workspace-time space for sampling-based motion planners

Authors: Schwesinger, Siegwart, Furgale

status: to be submitted

Link to Table of Contents.

Fast collision detection through bounding volume hierarchies in workspace-time space for sampling-based motion planners

Ulrich Schwesinger[1], Roland Siegwart[1] and Paul Furgale[1]

***Abstract*— This paper presents a method for fast collision detection for sampling-based motion planners based on a bounding volume hierarchy approach in workspace-time space. By introducing time as an additional dimension to the robot's workspace, the method is able to quickly evaluate time-indexed candidate trajectories for collision with other agent's known future motions. The approach makes no assumptions on the shape of the objects and is able to handle arbitrary motions. We highlight implementation details regarding the application of the collision detection technique within an online planning framework for automotive application. Furthermore, we give detailed profiling information to show the capabilities for real-time operation.**

***Index Terms*— Collision detection, workspace-time space, bounding volume hierarchy, axis-aligned bounding box tree**

I. Introduction

For a large number of motion planners, collision detection is the most time-consuming operation in its planning cycle. Especially sampling-based planners like the Rapidly Exploring Random Tree algorithm (RRT) [1] and RRT* [2], but also state-lattice approaches [3] have to evaluate numerous explicitly constructed paths for collision. The majority of implementations of sampling-based motion planners operates in the time-agnostic configuration- or state-space of the robot []. Operation in dynamic environments is tackled through frequent re-planning cycles. Planning in dynamic environments however eventually necessitates the detection of collisions of time-parametrized trajectories. Objects are considered not be in collision if they occupy the same space at a different point in time. This problem can be transformed into the problem of static collision detection in configuration-time space $\mathcal{CT}$ [4], [5] by introducing time as an additional dimension.

Valuable work was done applying sampling-based motion planning techniques in configuration-time space directly [6], [7]. In [7], Hsu et al. adopt a Probabilistic Roadmap Planner in state-time space to plan for kinodynamic systems amongst moving obstacles. Though mentioning the existence of more sophisticated collision detection techniques, this task is performed with a naive approach of checking every robot pose against each obstacle. Van den Berg et al. approach the planning problem in a dynamic environment by precomputing a roadmap of motions avoiding static obstacles. Later, they utilize a two-dimensional state-time space to find a collision-free trajectory on the roadmap [8]. Collision detection is done with the help of a regular grid in the two-dimensional state-time space. Lawitzky et al. presented an algebraic collision detection algorithm for automotive application in highway scenarios [9]. It assumes trajectories for rectangular objects given as quintic polynomials and neglects the orientation of the vehicles. Given these restrictions, it provides runtimes around $5\,\mu s$ for a collision test of two individual trajectories. As the authors mention, the negligence of the object's orientation is acceptable for highspeed driving in certain environments such as highways, but intolerable for most other applications which clearly limits the range of application of this method. In [10], Ziegler et al. target planning for a car-like vehicle along a reference path. A set of candidate trajectories is generated and tested for collision with moving objects with a method presented in [11]. This approach however necessitates a circular approximation of all agents' shapes, which might result in poor navigation performance in tightly occupied workspaces (imagine a car navigating through a corridor framed by sideways parked cars).

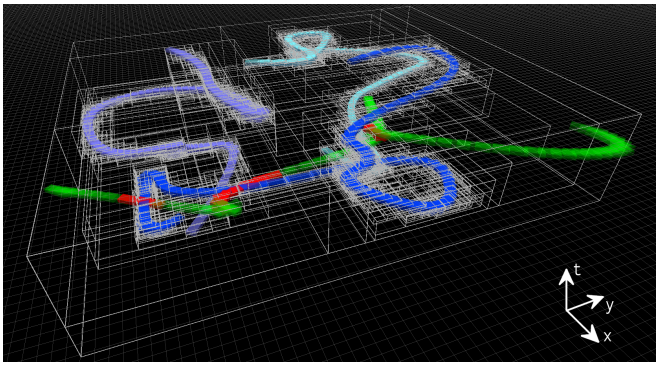


Fig. 1: Visualization of an AABB-Tree in workspace-time space for three obstacle trajectories generated by a random walk (blue shades). The ego trajectory (green/red) can be efficiently checked for collision with the help of the tree structure (wireframes).

Within this paper we report on the implementation of an Axis-Aligned Bounding Box Tree (AABB-Tree) data structure in workspace-time space for fast collision detection of candidate trajectories. This collision detection approach allows for fast trajectory collision queries with no constraints on the objects' shapes or motions. The approach was implemented in the sampling-based online motion planning framework presented in [12] and proved its real-time capabilities for automotive application.

*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).

[1]Autonomous Systems Lab, ETH Zurich, Tannenstrasse 3, 8092 Zurich
Corresponding author: ulrich.schwesinger@mavt.ethz.ch

The Axis-Aligned Bounding Box Tree belongs to the class of Bounding Volume Hierarchy (BVH) approaches. They are widely used in computer graphics and physical simulation [13], [14] in the so-called *broadphase* of the collision detection. They aim at reducing the number of exact collision computations (narrow-phase)) between collision shapes by pruning most of them through a hierarchically ordered sequence of simpler queries. Bounding Volume Hierarchy approaches build a tree of bounding volumes on an object level. Bounding volumes proposed so far are bounding spheres, axis-aligned bounding boxes (AABB), oriented bounding boxes [15] or discrete-orientation polytypes [16]. For applications in 3D, implementations are readily available in well-maintained software packages like the Open Dynamics Engine [13], the Bullet Physics Library [14] or the recent Flexible Collision Library (FCL) [17].
Closely related to BVH approaches, spatial partitioning techniques like KD-Trees, Quadtrees, Octrees and voxel- or grid-based collision detection approaches apply the partitioning in space rather than on the level of objects. The choice between spatial and object-oriented partitioning techniques strongly depends on the size of the environment under consideration and the amount of obstacles occupying it. Since in this paper we target the application in an online automotive receding-horizon planning framework [12] where a moderate amount of objects has to be considered within the vicinity of the vehicle, an object-based partitioning scheme is superior.

In his pioneering work [18]–[20], S. Cameron introduces three approaches to the collision detection problem of moving rigid objects. Firstly "Multiple Interference Detection", which tests the shapes for collision at subsequent fixed instances in time. Secondly the "Swept Volume" approach that aims at computing the volumes in workspace that is swept by the objects over time and testing the volumes for intersection. However it may happen that the swept volumes intersect but no collision actually takes place due to different time instances at which the objects will occupy the same space. And thirdly "Extrusion", which is the process of creating a collision object in the workspace-time space that we follow within this paper. On overview of these approaches including more references is given in [21].

II. Contribution

Within this paper we report on the implementation of an Axis-Aligned Bounding Box Tree (AABB-Tree) data structure in workspace-time space for fast collision detection of candidate trajectories. Though the idea of using partitioning techniques in workspace-time is not new, we are not aware of any application within an online motion planning framework, especially within the automonous driving domain. The paper highlights various implementation considerations and provides detailed runtime statistics to facilitate decisions regarding the usage of this approach within other online planners operating in dynamic environments.

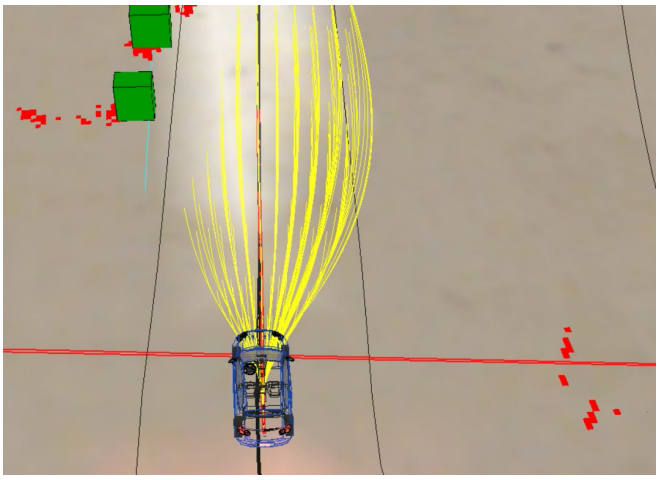


Fig. 2: Candidate trajectories (yellow) of motion planner to be checked for collision with dynamic objects (green) over time,

III. Approach

In this approach, we consider the task of detecting collisions of candidate trajectories for a single robot operating in a two-dimensional workspace $\mathcal{W}_2 = \mathbb{R}^2$. Candidate trajectories have to be checked for collision against movement predictions of objects over time. Configurations of objects are only considered as in collision if they collide at a certain time. An exemplary setup obtained with the motion planning framework presented in [12] is illustrated in 2. The collision tests are accelerated with the help of a bounding volume hierarchy data structure. Note that the application is not limited to two-dimensional workspaces. Bounding volumes might also be defined in higher dimensions.

First, we would like to recap the notion of a workspace and a configuration space and explain the domain in which the AABB-Tree is constructed. A rigid body that can translate and rotate within $\mathcal{W}_2$ has a three-dimensional configuration space $\mathcal{C} = \mathbb{R}^2 \times \mathbb{S}^1$. Obstacles moving within this workspace create obstacle regions at certain time instances, imposing a configuration-time space obstacle region $\mathcal{CT}_{obs}$ for the robot at hand. Explicitly constructing $\mathcal{CT}_{obs}$ is difficult and time-consuming for arbitrarily shaped robots, since it requires a description of the obstacle region with respect to all possible rotations. Therefore we explicitly construct the workspace-time obstacle region $\mathcal{WT}_{obs}$ and incorporate the rotation at query time.

The volume occupied in $\mathcal{WT}$ by an agent moving along its trajectory forms a collision object in such. For general motions this collision object is usually non-convex. We define a trajectory for agent i as a sequence of time-indexed configurations $\mathbf{T}^i = [(\mathbf{c}_0, t_0), (\mathbf{c}_1, t_1), \ldots, (\mathbf{c}_K, t_K)]$. This restriction on discretized movements stems from the general difficulty to compute intersections for arbitrary analytic curves. The union of all volumes of the N agents (excluding the agent acquiring the plan)

$$\mathcal{WT}_{obs} = \bigcup_{i=1}^{N} \underbrace{\bigcup_{k=1}^{M_i} \mathbf{T}_k^i}_{\substack{\text{workspace-time} \\ \text{collision object} \\ \text{of agent } i}} \tag{1}$$

defines the obstacle region $\mathcal{WT}_{obs}$.

In a naive implementation, at each point in time each ego configuration of the discretized candidate trajectory has to be checked against the configuration of each agent, resulting in $O(K\dot{N})$ collision tests with K being the number of configurations of the trajectory query, and N being the number of obstacle trajectories. By using a BVH data structure in a broadphase collision detection phase, the number of required exact checks (narrow-phase)) between collision objects can be greatly reduced. Therefore, each pose of each collision object is inserted into the AABB-Tree (see Figure 1). The root bounding box of the AABB-Tree fully contains all collision objects. Upon a query the tree is traversed in depth solely where the bounding boxes of the query and the tree intersect. Should the bounding box within the tree be a leaf node, the collision detection goes into a narrow-phase in order to test the associated collision shapes for overlap (e.g. box-to-box or sphere-to-sphere). Note that narrow-phase collision algorithms exist for numerous collision shapes. Within this paper we exclusively illustrate AABB-Trees for oriented box shapes.

In order to integrate this collision detection technique into an online planning framework, there are various considerations within this approach that require further attention.

A. Isolated Configuration Queries Against the AABB-Tree

The most simple application are isolated configuration queries. Each of the ego configurations $\mathbf{T}_k^i$ for each time instance k is checked for collision against the union of collision objects over time $\mathcal{WT}_{obs}$ individually. Though being inferior to the later presented "Tree vs. Tree" query in terms of computation time, this approach avoids the not negligible overhead of computing the BVH data structure for each trajectory in the set of queries. In a usual setting, the number of trajectories in the queries is much larger than the number of obstacles which could prevent this computation to be realizable with real-time constraints for a planner in which the candidate trajectories are generated on-the-fly.

Another fairly obvious choice is to abort the collision test for a trajectory, as soon as a collision is detected. Within the context of this paper, we call this the early-exit option. Without prior knowledge of the index of the collision within the pose sequence a linear traversal of the sequence is as good as any other. Applying a biased ordering (based on collision tests in previous planning cycles) can potentially turn out beneficial.

B. "Tree vs. Tree" Queries

Another option is to utilize a second BVH data structure for the trajectory in the query as displayed in Figure 3. In the broadphase of the collision detection therefore two trees are tested against each other. This is done by traversing both trees top to bottom. This improves runtime performance with respect to the query, but comes at the additional cost of creating the BVH data structure for every candidate trajectory. For planners with predefined motion primitives the tree structures for each candidate trajectory can be precomputed and thus do not further reduce the spare computation time for online operation. In case precomputation is infeasible and has to be performed in each planning cycle, we provide profiling information for this operation in section IV.

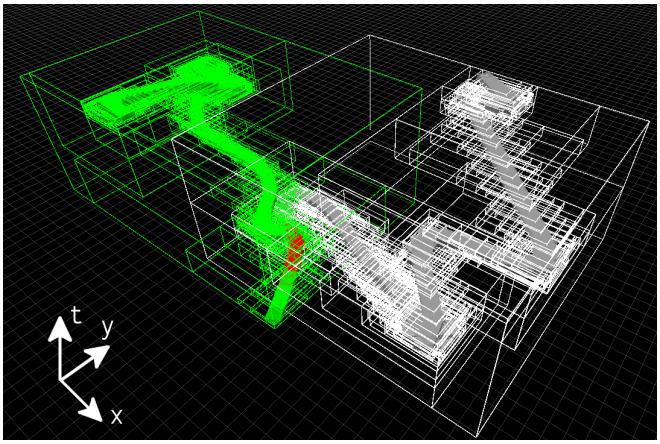


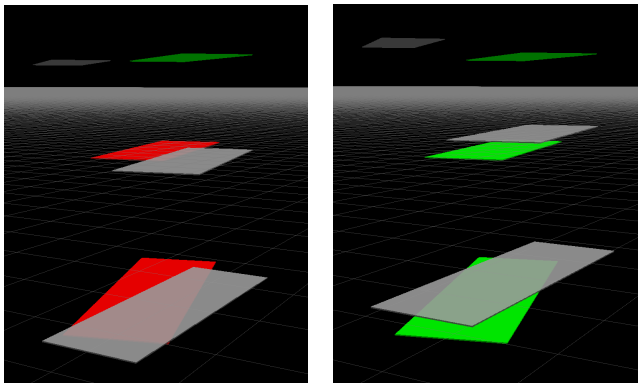
Fig. 3: Collision test with two AABB-Tree data structures. The white tree contains all collision objects, the green tree all query configurations of one trajectory. Collisions can be detected faster through a more efficient resolution of the candidate points of collision in workspace-time space.

C. Time Discretization of Motions and Temporal Extent of Primitives

By default, a collision object does not have a temporal extent, e.g. a discretized trajectory of rectangular collision objects consists of a sequence of flat rectangular objects. Special attention has to be paid to apply the same time discretizing for the obstacles' motion predictions and the candidate trajectories. Collisions will certainly be missed unless the positions in the time dimension are equally spaced as illustrated in figure 4.

Though applying an equal time discretization eliminates the danger of missed collisions for two-dimensional objects, there are two arguments that speak for introducing 3D primitives.

- Enforcing a *minimum time-gap* between trajectories becomes straight forward by setting an extent of the collision shapes in time dimension. The minimum time-gap setting will label configurations as colliding if they occupy the same space close in time. This is a desirable behavior for many motion planning tasks—above all in automotive applications—as it results in less aggressive behavior of the robot.
- Allows the use of well-maintained software libraries [13], [14], [17] without modification as these libraries



(a) (b)

Fig. 4: Problems arising from discretized trajectories. Grey boxes: obstacle shapes, red/green boxes: query shapes detected as colliding/not colliding, a) Equal time discretization for ego and obstacle trajectories. Collisions are detected correctly. b) Different time-discretization for ego and obstacle trajectories causes missed collision detections.

are designed for collision detection between three-dimensional objects.

D. Convex Hull Representation of Trajectories

The usage of discretized motions gives rise to potentially missed collisions depending on the coarseness of the time-resolution. A way to overcome this issue is to compute the convex hull of every subsequent tuple of configurations within a trajectory. The convex hull computation should be performed for both the collision objects and the query trajectories. The workspace-time obstacle of agent i is represented as

$$\mathcal{WT}_{obs,i} = \bigcup_{j=2}^{N} \text{convex\_hull}\left(\mathbf{T}^i_{j-1}, \mathbf{T}^i_j\right) , \qquad (2)$$

resulting in a sequence of $N-1$ convex hulls. Again, each of the convex hull shapes can be inserted into the AABB-Tree for accelerated collision queries. Figure 5 depicts the convex hull representations and the resulting AABB-Tree for three obstacle and one candidate ego trajectory, each coarsely discretized.

IV. Results

Within this section we mainly provide runtime results in order to give implementation guidance for the interested reader. The computations were performed on a single core of an Intel Core i7@2.67 GHz processor with 32 kB/256 kB/4096 kB of L1/L2/L3 cache. Profiling was done measuring CPU cycles. For the sake of a more intuitive interpretation, CPU cycles are converted to timings with the above mentioned processor at hand ($t_{\text{comp}} = {}^{N_{\text{cycles}}}/_{2.67 \cdot 1E9}$). To account for disturbances, each timing measurement is generated by the minimum of multiple measurements. We utilized the AABB-Tree data structure from Willow Garage's Flexible Collision Library (status May 21st, 2014). The

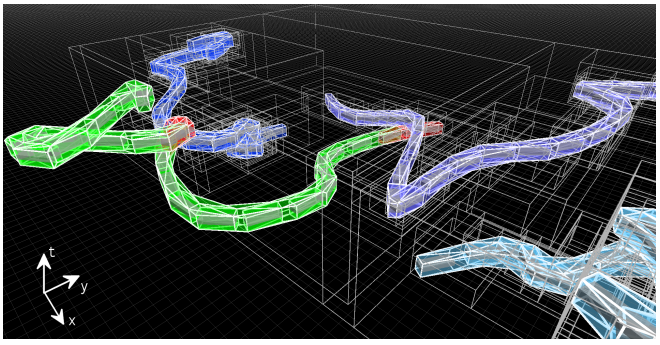


Fig. 5: Convex hull representations in workspace-time space for three obstacle trajectories (blue shades) and one ego trajectory (green/red). Collision queries between convex hull shapes can be accelerated with the AABB-Tree data structure (wireframe boxes).

code was compiled with gcc 4.6.3. Note that a slightly more efficient implementation is possible via a specialized narrowphase collision detection algorithm accounting for the fact that roll and pitch angles of the workspace-time space objects are always equal to zero.

Since runtimes are highly dependant on the proximity and collision state of the trajectories, trajectories are randomly generated. In order to create a randomized trajectory, a time vector with equidistant spacing is generated and a corresponding sequence of configurations is generated via a random walk.

The dependency of the runtimes of three different approaches on the number of obstacle trajectories are presented in Figure 6. The figure gives the runtimes of the "Isolated Configurations" (section III-A), "Tree vs. Tree" (section III-B) compared to the naive "Multiple Interference Test" approach. To provide a fair comparison, the same shape primitives (3D Boxes) and the same narrowphase collision detection implementation were used. For each number of obstacle trajectories, a total of 1000 random trajectories were generated and evaluated for collision with each approach. The timing information is separated for the cases in which the ego trajectory is in collision with at least one obstacle and for the cases in which the ego trajectory is collision-free. The case of a collision-free constellation represents the worst-case for the naive "Multiple Interference Test" approach, since all configurations have to be tested in order to rule out a potential collision. As expected this approach shows a linear dependency with respect to the number of obstacles. Runtimes for the approaches utilizing the accelerating AABB-Tree data structure likewise grow due to the increasing depth of the tree, yet outperform the naive approach more and more with a growing number of obstacle trajectories. The "Tree vs. Tree" approach—more effectively dissolving the potential points of collisions in workspace-time space—is superior to the "Isolated Configurations" approach.

The performance difference shrinks when collisions start to occur due to the random point of occurrence of the collision

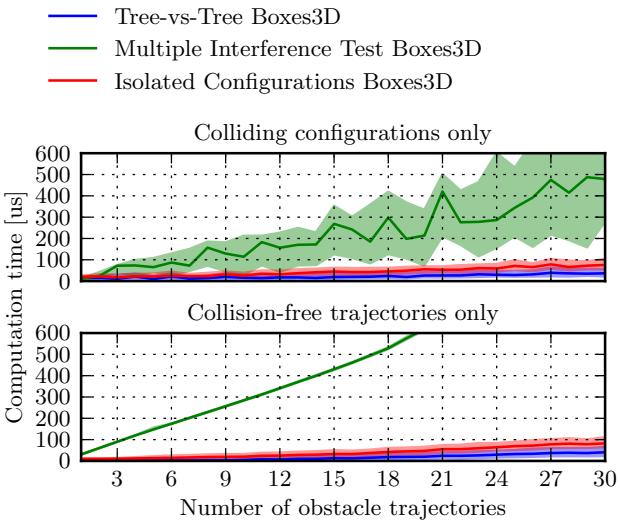


Fig. 6: Median (solid lines) and 0.25/0.75-percentiles (filled area) of runtimes given in µs for collision test of one query trajectory with 150 poses with respect to number of obstacle trajectories for three different approaches. In both cases the tree aproaches are superior to the naive "Multiple Interference Test" approach.

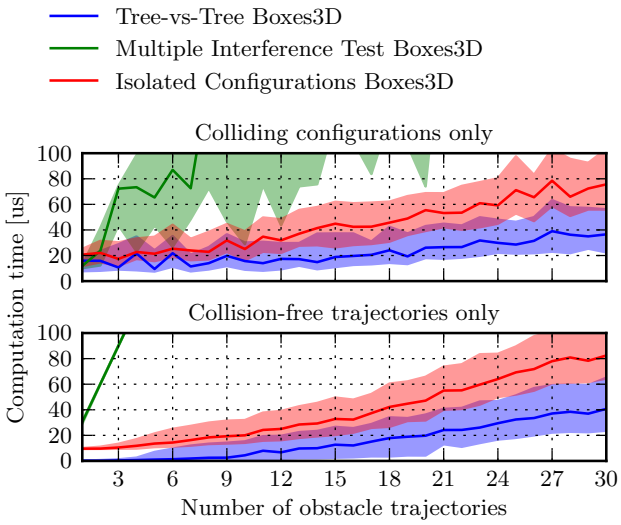


Fig. 7: Zoomed plot of figure 6

within the ego trajectory. In case the collision occurs early, there is a only a small number of narrowphase checks to be performed in the "Multiple Interference Test" approach. Depending on the depth of the tree, this is often faster than the sum of AABB tests that have to be performed upon tree traversal. Nonetheless the tree approaches tend to provide smaller computation times in average.
The slightly higher computation times of the tree approaches for the collision-free constellations result from a higher amount of narrowphase checks in cases where configurations get close, yet do not collide.

For better visibility, figure 6 is again displayed in figure 7 with different limits in the computation time axis. One can see that especially for no the no-collision case (which has to be taken as a reference regarding wort-case computation time compared to the naive "Multiple Interference Test" approach), the "Tree vs. Tree" approach provides remarkable runtimes.

Figure 8 illustrates the time needed for construction of the AABB-Tree data structures for an increasing amount of obstacle trajectories. In case of fixed motion predictions throughout the planning cycle, this operation has to be performed once. For a total of 30 obstacle trajectories, the computation time for constructing the accelerating tree structure is around 7.8 ms. We argue that 30 obstacles represent a fairly large amount for a broad spectrum of applications and that the computation time for constructing the tree structure for the obstacles is acceptable for online operation within sampling-based planners.

The runtime for the construction of the AABB-Tree data structure for one trajectory is 233 µs. This clearly limits the number of ego trajectories that can be tested for collision within the "Tree vs. Tree" approach in a real-time scenario. Sampling-based planners such as presented in [10] and [12] explore hundreds of candidate trajectories, hence rendering the 'Tree vs. Tree" approach unfeasible for planners relying on an online generation of the candidate trajectories. Yet it is a valuable option in case the tree structures can be precomputed for the candidate trajectories.

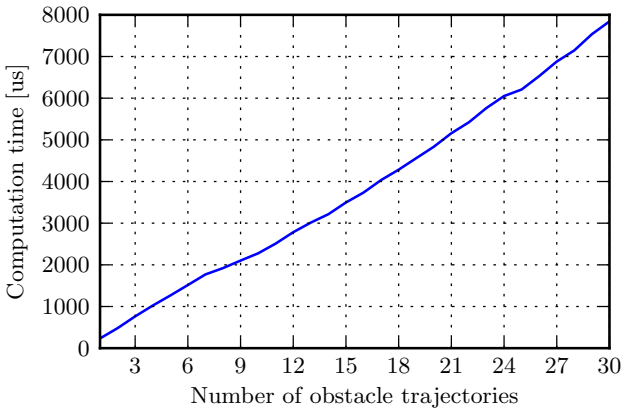


Fig. 8: Mean runtimes given in µs for construction of AABB-Tree data structure for varying number of obstacle trajectories.

This preliminary version of the paper still misses the evaluation regarding convex hull representations of trajectories. This will be included in the final version of this paper.

V. Conclusion

In this paper we reported on the implementation of an AABB-Tree data structure for fast collision queries in workspace-time space. The approach makes no assumptions on the shapes of moving obstacles and can handle

arbitrary motions. We discussed and compared several different approaches. Testing single configurations against an AABB-Tree capturing the union of workspace-time obstacles results in substantial improvements in runtimes over the naive multiple interference test approach, while keeping the computaional overhead that comes with the construction of the accelerating AABB-Tree data structures low. Our provided results on computation times underline the real-time applicability of this approach. Computing an AABB-Tree for every candidate trajectory provides further speed-up but comes with a significant additional computational cost for common sizes of the candidate trajectory set, rendering this approach only feasible if precomputation of the tree structures is possible.

REFERENCES

[1] S.M. LaValle. Rapidly-Exploring Random Trees: A New Tool for Path Planning. *Technical Report*, 129(98-11):11–98, 1998.

[2] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

[3] Mihail Pivtoraiko and Alonzo Kelly. Constrained Motion Planning in Discrete State Spaces. In *Field and Service Robotics*, pages 269–280, 2005.

[4] M Erdmann and T Lozano-Perez. On multiple moving objects. *Algorithmica*, 1987.

[5] JC Latombe. *Robot motion planning*. 1990.

[6] M Zucker, J Kuffner, and M Branicky. Multipartite RRTs for Rapid Replanning in Dynamic Environments. *Robotics and Automation*, 2007.

[7] David Hsu, Robert Kindel, Jean-Claude Latombe, and Stephen Rock. Randomized Kinodynamic Motion Planning with Moving Obstacles. *The International Journal of Robotics Research*, 21(3):233–255, 2002.

[8] J P van den Berg and M H Overmars. Roadmap-based motion planning in dynamic environments. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2, pages 1598–1605, 2004.

[9] A Lawitzky, D Wollherr, and M Buss. Maneuver-based risk assessment for high-speed automotive scenarios. *Intelligent Robots and Systems*, 2012.

[10] M. Werling, S. Kammel, J. Ziegler, and L. Groll. Optimal trajectories for time-critical street scenarios using discretized terminal manifolds. *The International Journal of Robotics Research*, 31(3):346–359, December 2011.

[11] J Ziegler and C Stiller. Fast collision checking for intelligent vehicle motion planning. In *The Intelligent Vehicles Symposium (IV)*, pages 518–522, June 2010.

[12] Ulrich Schwesinger, Martin Rufli, Paul Furgale, and Roland Siegwart. A Sampling-Based Partial Motion Planning Framework for System-Compliant Navigation along a Reference Path. *Proc. of the IEEE Intelligent Vehicles Symposium (IVS)*, (269916):6, 2013.

[13] Russell Smith. Open Dynamics Engine. `http://www.ode.org/`, May 2014.

[14] Bullet physics library. `http://bulletphysics.org/wordpress/`, May 2014.

[15] S Gottschalk, MC Lin, and D Manocha. OBBTree: A hierarchical structure for rapid interference detection. *SIGGRAPH Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996.

[16] Joseph S. B. Mitchell Henry Sowizral Karel Zikan James T. Klosowski, Martin Held. Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs.

[17] Jia Pan, Sachin Chitta, and Dinesh Manocha. FCL: A general purpose library for collision and proximity queries. In *2012 IEEE International Conference on Robotics and Automation*, pages 3859–3866. IEEE, May 2012.

[18] S. Cameron. Modelling Solids in Motion, 1984.

[19] S. Cameron. A study of the clash detection problem in robotics. *IEEE International Conference on Robotics and Automation*, 1985.

[20] S. Cameron. Collision detection by four-dimensional intersection testing. *Robotics and Automation, IEEE Transactions on*, 1990.

[21] P. Jiménez, F. Thomas, and C. Torras. Collision detection algorithms for motion planning. In J.-P. Laumond, editor, *Robot Motion Planning and Control*, pages 1–53. Springer-Verlag, Berlin, 1998.