UP-Drive

Automated Urban Parking and Driving

Research and Innovation Action - Horizon 2020
H2020-ICT-24-2015: Robotics
Grant Agreement Number 688652

Project start date: January 1, 2016, Duration: 48 months

Deliverable D7.1

Software specification and architecture for decision making and navigation

Coordinating partner: Volkswagen A.G., Group Research
Coordinating person: Wojciech Derendarz

Lead contractor for this deliverable: Volkswagen A.G.
Deliverable editor: René Waldmann

Due date of deliverable: June 30, 2016
Date of submission: 13.09.2016
Revision: 1.0

Dissemination level: Public

Executive Summary

This deliverable provides insights of the current software architecture representing the decision-making and navigation framework assigned to WP7 of the *UP-Drive* project. The presented tool chain has been evolved during other EU funded projects like *V-Charge*, *InteractIVe* or *AdaptIVe* [12, 11, 10] reflecting the experience gained in the course of these projects.

An introduction is given in section 1. The embedding of WP7 in the overall project context is explained first (section 1.1). Then the task formulation follows recapping what the system requirements on this work package are and why the decision-making and navigation stack is essential to generate self-driving capabilities beyond the state-of-the-art. A coarse layer overview is then presented in section 1.3.

Section 2 presents an overview of existing architectures proposed in the last decade to provide sophisticated decision-making behavior combined with advanced navigation and trajectory planning capabilities.

Finally, the applied methods and implemented modules of the comprehensive application stack are elucidated in detail in sections 3, 4, 5, and 6.

Contributing Authors/Partners

Partner	Author	Contribution
Volkswagen	René Waldmann	document editor

Revision Table

Revision	Date	Revision details	Contributors
1.0	September 13, 2016	Initial document submission.	René Waldmann

Contents

1 Introduction **6**
1.1 *UP-Drive* project context 6
1.2 The task formulation 7
1.3 Layer overview 7

2 Related Work **9**

3 Routing **10**
3.1 Global Path Planner 10
3.1.1 Interfaces 10
3.2 Local Path Planner 11
3.2.1 Interfaces 11

4 Decision-Making **12**
4.1 Global Strategy - Mission Coordinator 12
4.1.1 Interfaces 13
4.1.2 Interfaces 14
4.2 Local Decision - Scene Collector 15
4.2.1 Interfaces 16
4.2.2 Interfaces 17

5 Planning **18**
5.1 Trajectory Planner 19
5.1.1 Interfaces 19
5.2 Free Space Planner 20
5.2.1 Interfaces 20

6 Vehicle Control **22**
6.1 Trajectory Controller 22
6.1.1 Interfaces 22
6.2 Engine Controller 22
6.2.1 Interfaces 23

7 Conclusion **24**

1 Introduction

1.1 *UP-Drive* project context

The Deliverable 7.1 contributes to the *UP-Drive* (Automated Urban Parking and Driving) project which aims to create a car capable of self-driving in unconstrained urban environments with 30 km/h speed limits. *UP-Drive* goals are specified in the Description of Work [3] and the Deliverable 1.1 [2].

The *UP-Drive* consortium intends to build up a demonstrator platform consisting of a sensor-rich electric VW Golf sharing data with a cloud environment to demonstrate automated transportation in urban environments. *UP-Drive* exploits know-how and the technology of the previous *V-Charge* [12] project, in which partners VW and ETH Zurich participated.

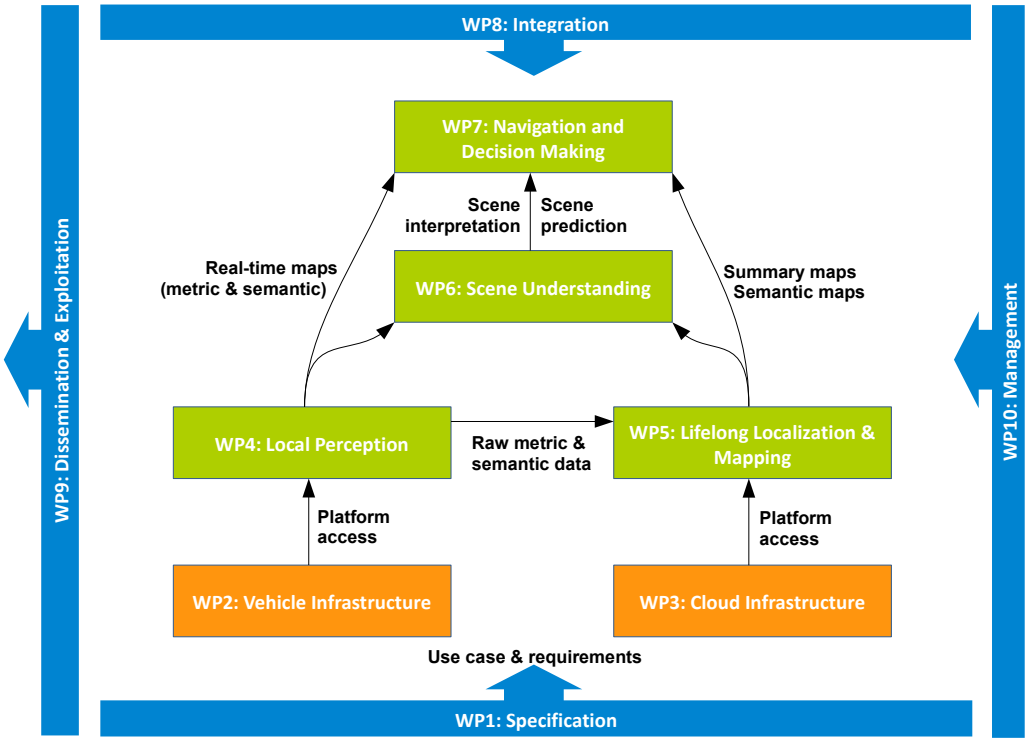

Figure 1: Work Package overview with key interfaces between the individual work packages.

The *UP-Drive* scientific and technological scope is outlined by three use-cases to be demonstrated:

- UCG-1 On-street valet parking.
- UCG-2 Picking up kids from a school, grandparents or a train station.
- UCG-3 Picking up groceries from the supermarket.

The functionality needed to deliver *UP-Drive* skills, which will enable to demonstrate the above use-cases, was decomposed into a top level system architecture as given in Fig. 1

Deliverable 7.1 will provide the general architecture of the navigation and decision-making stack as specified in the Description of Work at page 30. This report includes listing all components involved in the process as well as their interfaces employed to share data.

1.2 The task formulation

WP7 aims to provide all required functionalities for routing, decision-making, and trajectory planning. To this end, mandatory vehicle capabilities resulting from this WP definition have been derived (see [2], section 4.6):

- Smooth driving within own lane while keeping the track error very low
- Keeping safety distance to leading vehicle
- Constantly adapt the lateral distance to static objects within the own lane
- Decide whether to wait behind or to pass a vehicle standing in the own lane
- Compute collision free trajectories to handle pedestrians crossing the street
- Stop at zebra crossings to let pedestrians pass by
- Plan collision free trajectories to overtake pedestrians walking along the lane
- Compute a safe strategy to overtake and pass obstacles in presence of oncoming traffic
- Search for free parking spots and park in and out
- Do lane changes when necessary
- Follow traffic rules while passing intersections

Note that this is only a very rough summary of the system requirements. A detailed overview is available in Deliverable 1.1. However, the list implies a substantial complexity that comes with inner-city traffic. In such scenarios, it is insufficient to generate trajectories while not taking into account either some high level strategy to reach the goal or the intention of other traffic participants. Consequently, in the absence of such capabilities, the ego vehicle motion is very likely not to resemble human driving behavior. For instance, when overtaking obstacles in presence of oncoming traffic, safety critic decisions have to be made which requires good knowledge and understanding about what is going on around the vehicle. In such situations, an excellent decision-making layer closely collaborating with trajectory planning units is mandatory.

1.3 Layer overview

A coarse overview of the current architecture is depicted in Fig.2. The layers relevant for this deliverable are color marked and will be elucidated in details. The arrows indicate the interfaces connecting the decision and navigation modules with other components which may reside outside of the planning level.

For the sake of clarity, the responsibilities of the individual layers are shortly recapped.

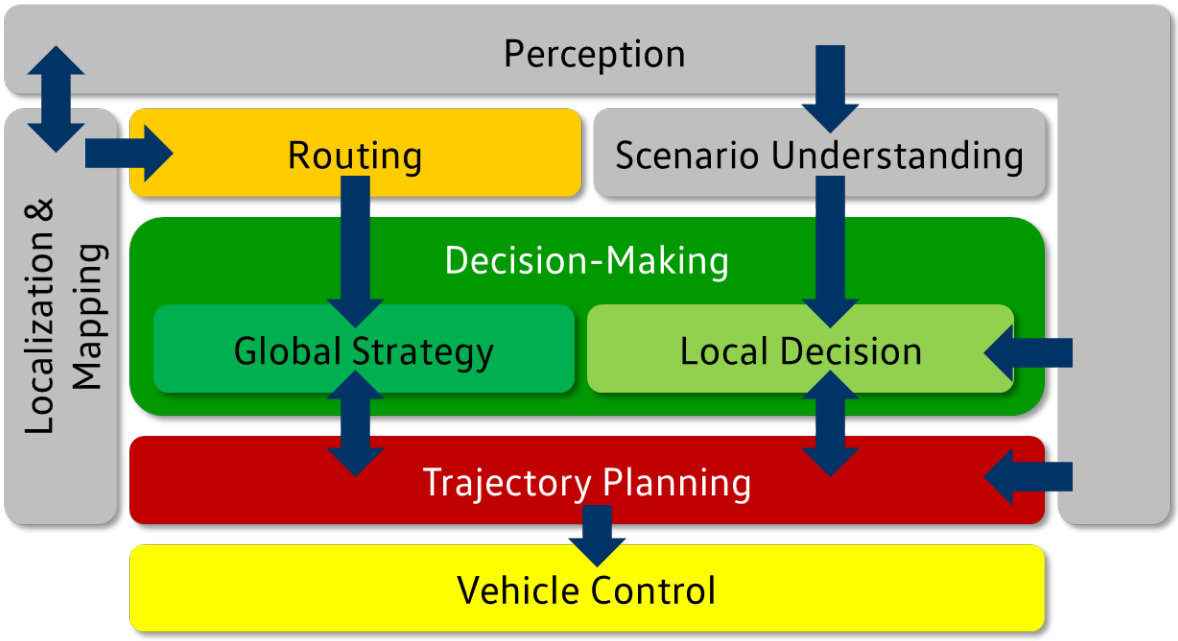

Figure 2: Overview of the navigation and decision-making system.

Routing This layer provides all functionalities necessary to compute a global reference path to the mission goal specified by the user (note that mission goals are being explained later in this section). Here, a reference path could either be a smoothed drivable point list or a set of consecutive edges provided by a standard navigation system. The routing layer gets ego data from the localization system defining the start pose for the navigation routine. The result is being forwarded the decision-making layer strategically guiding the vehicle to the target.

Decision-Making The decision-maker layer is being grouped into the global strategy and local decision sub-layers both sharing data with the trajectory planning modules.

Global Strategy All modules hosted within global strategy take the result from the routing layer as well as the overall mission goal as input and compute a strategy to successfully accomplish the mission. To be more precise, this implies to coordinate other functional entities guiding the vehicle to its target.

Local Decision This layer is responsible to adequately make the car react to the local traffic situation. E.g., whenever the vehicle approaches an intersection, decisions have to be made frequently to pass the intersection and to constantly adapt the current strategy to the behavior of other vehicles, pedestrians, etc. To this end, the scenario understanding system forwards the interpretation of the local scenario to the decision modules. Another input comes from perception providing (dynamic) traffic objects.

Trajectory Planning All modules capable of performing path and trajectory planning are subsumed in this layer. The planners receive trigger from the global strategy system and get data from perception as well as local decision. Both data streams serve as basis for collision free and natural driving maneuvers. The planning results are forwarded to the controller.

Vehicle Control The controller takes the trajectories as input and computes the according control commands,e.g., acceleration and steering wheel velocity.

2 Related Work

In the past, many complex and impressive autonomous vehicle systems have been demonstrated which internally hosted navigation modules controlled by behavior layers. The first prominent example is *Stanley*, the vehicle that won the DARPA Grand Challenge in 2005 [16]. Although, only one single route guided the competitors through the Mohave dessert, *Stanley* already had a state management system with a global driving mode maintained in a finite state machine. However, it remains unclear how this driving mode exactly worked. The navigation component consisted of a global path planner smoothing the road network data and a local path planner which was capable of avoiding obstacles and guiding the vehicle back to the reference path.

The vehicle *Junior* [8] which competed in the DARPA Urban Challenge (2007), featured a hierarchical state machine with 13 states to handle urban traffic scenarios. Typical driving behavior modes were *drive forward*, *stop at intersections*, or *wait at stop signs*. On the navigation side, a global path planner has been applied to compute an optimal route from every location in the map to the next check point using dynamic programming.

The winning vehicle *Boss* [17] was able to perform three different driving categories, namely *lane driving*, *passing intersections*, and *parking* (which was called *achieving a zone pose* by the authors). Furthermore, a mission planner connected the road network data to a graph structure where each node contains an optimal path to the goal. This information has constantly been updated. Then, a motion planner computes a set of feasible trajectories along the reference path to reach the goals provided by the mission planner.

The vehicle *Odin* which achieved the 3^{rd} place, had a driving behavior layer which internally managed three different driver types, i.e., *target point driver*, *speed driver*, and *lane driver* [15]. The selection of the preferred driver type has been conducted by using a *winner-takes-it-all* strategy. A motion planner is considered as decision-making layer between the driving behavior module and the vehicle interface converting the incoming target points, lane and speed goals into low level vehicle commands using a trajectory planner. However, before computing the trajectory, a navigation strategy has been generated by roughly subdividing the scenarios into lane navigation and zone navigation (where parking takes place).

Other well known urban challenge vehicle examples are *Caroline* [14], *AnnieWAY* [6], *Skynet* [7], or *Knight Rider* [9] which all featured comparable driving behavior, decision-making, and navigation stacks.

Another real-time decision-making and navigation approach has been presented by Furda and Vlacic[4]. They propose a decision layer consisting of two states. The first stage generates a set of feasible driving maneuvers, i.e., maneuvers which are safe and follow the traffic rules. To this end, an event based system has been set up forwarding the incoming events to a Petri Net model. Furthermore, route planning information are fed to the Petri Net. The latter is important as overtaking another vehicle right before a planned turn might not be feasible. The second stage takes the set of all feasible driving maneuvers as input and selects one maneuver according to objectives as efficiency and comfort.

However, *V-Charge* [5, 12] as predecessor project of *UP-Drive* used a system where at a first stage a global, drivable path has been computed once, guiding the vehicle over the parking lot. Then a mission has been planned generating a set of tasks necessary to reach the parking spot. Typical tasks were *switch on engine* and *follow lane*. So-called task processors

for each task have been defined. E.g., the trajectory planner was capable to follow the lane or to pass intersections. At the beginning of a mission, every task processor registered his capabilities at a central coordinating instance, which was called *mission executive*. This module continuously processed the task list and activated the responsible task processors. Decision-making took place in situations were the pre-planned task list did not hold, e.g., when the parking spot was occupied. Then the system had to decide which strategy to follow next to reach another parking spot.

3 Routing

This is the topological navigation layer constituting the first tactical unit in the architecture. Routing is being subdivided into a global and a local part. Global planning is similar to what standard navigation systems do by computing a sequence of roads guiding the vehicle to its goal. In contrast, local planning frequently computes a short distance path on lane level telling the vehicle how to traverse the roads coming from the global planner.

3.1 Global Path Planner

The global path planner is responsible for computing a path from the current vehicle location to the position of the mission goal. Here, the orientation of the final pose is not important as only a set of edges connecting the current pose and the target is generated. In contrast to the goal, the active vehicle orientation is being considered in order to let the route start in the current driving direction. The edges represent the road network which the vehicle is supposed to follow. It is important to note that the path constitutes a coarse global strategy to reach the goal which does not include detailed lane information. Furthermore, the global path is only computed once unless the mission goal changes or a complete road is blocked. The global routing concept is different to the one employed in the *V-Charge* project [5],[12],[18]. *V-Charge* did not employ local path planning on a topological level (see 3.2). Moreover, the global path is not being smoothed anymore as we assume the lane model to be driveable now.

3.1.1 Interfaces

Road Description The global path planner collects the following data from the road description module:

- Road data on a coarse level, i.e., no detailed lane information
- Links connecting the roads to a street network
- Slip roads
- Intersections

Mission Coordinator The mission coordinator gets the mission goal and a topological path from the global path planner. The topological path consists of a set of connected roads guiding the vehicle to its target. This data are very similar to the routing result of a standard navigation system. See section 4.1 for more details on the mission goal.

Global Ego Provider The global path planner receives the global localization result from the ego provider. This is mandatory for route planning.

3.2 Local Path Planner

The local path planner maps the global path to a lane-level in the local vehicle environment. Here, *local* means a few hundred meters up to two or three kilometers. The result is a set of lanes guiding the vehicle precisely towards its goal. In contrast to the global path planner, the local path planner continuously calculates the optimal local route considering the current traffic flow, etc..This information is stored as so-called lane advises and added to the road network. Later, this data are also added to the scene and can be exploited during maneuver planning (see section 4.2.1).

3.2.1 Interfaces

Road Description The local path planner collects the following data from the road description module:

- Roads
- Lanes
- Intersections
- Traffic signs
- Traffic lights

In return, the local path planner injects the lane advises which implies that every module which queries the road network has access to the local path.

Mission Coordinator The mission coordinator accepts the planning result of the global path planner as input including information on the mission goal. Moreover, the local path planner frequently forwards its result to the mission coordinator. Thus, the mission coordinator is capable to decide when the mission goal has been reached.

Explorer The explorer triggers the local path planner whenever the mission goal includes searching for free parking spots.

Object Fusion The object fusion module provides information on dynamic objects which is required by the local path planner to choose the optimal lane to follow.

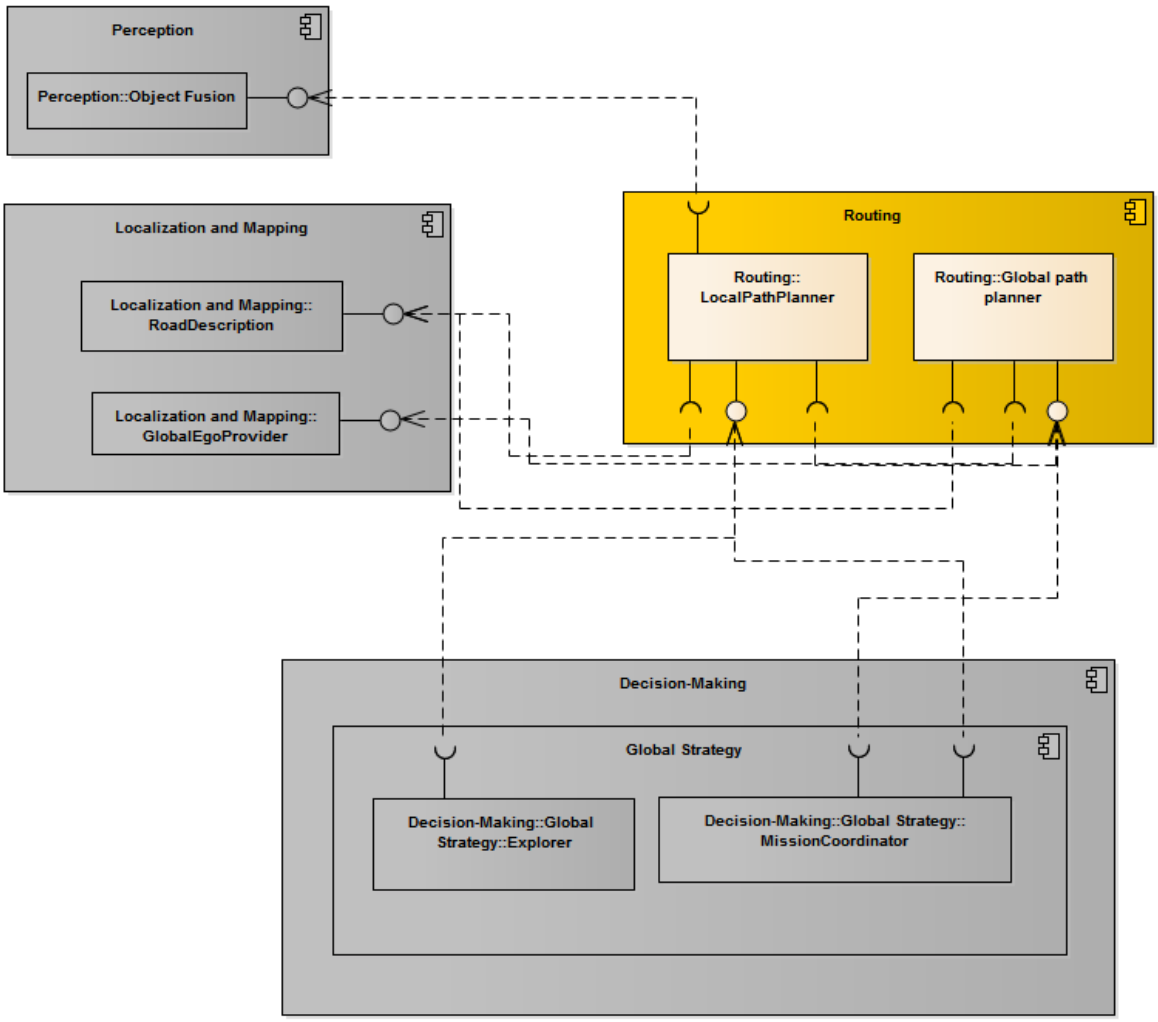

Figure 3: All components involved in the global routing process.

4 Decision-Making

This is the actual strategy layer coordinating the action modules like path planners and making decisions based on the current traffic situation. This layer is subdivided into a global strategy unit responsible for planning the mission and triggering the modules at the right time and a local part computing the vehicle maneuvers.

4.1 Global Strategy - Mission Coordinator

This module has already been described in detail in Deliverable 1.1 [2]. The mission coordinator is responsible to trigger the right functional modules depending on the current mission state. Currently, we consider three different mission goals.

- **Homing** This mission goal resembles a standard navigation task. The driver is in the vehicle and specifies a target address. Then the vehicle automatically drives to the target. As soon as the target address has been reached, the mission is over, i.e., no parking is being triggered.

- **Parking** Here, the driver may leave the vehicle. The target is a dedicated parking spot, e.g., a charging bay owned by the user. The vehicle starts somewhere on the road and automatically drives to its final pose on the parking spot. The target can either be defined using the vehicle human machine interface (HMI) or a mobile device. The use case is that the driver drops the vehicle off when the target address(e.g. the home) has been reached and the vehicle navigates to its dedicated parking spot. This is very similar to the mission considered in *V-Charge*.

- **Exploration** When triggering this mode, the driver is not supposed to be in the car anymore. The use case is to drop the vehicle off when the target address has been reached. Then the vehicle automatically searches for a free parking spot. The difference to the parking mission is that the parking spot is not known in advance.

An example of a homing mission is the user getting into the vehicle in the morning to drive to his workplace. The target address is being defined via the navigation system. Then the mission coordinator triggers the initialization of the car first to start the engine. As a next step, a parking planner is being launched to exit the parking spot. Finally, another trajectory planner resumes to follow the road to the target address.

The concept of the mission coordinator is closely related to the mission executive functionality presented in [18],[5]. Currently, we do not explicitly follow the task processor concept anymore to lower the complexity of the mission coordinator. Concretely, the objective was to reduce the communication between task processors and mission executive which was error-prone due to the amount of messages which have been exchanged. Instead, the mission coordinator just sends a trigger to let the functional modules start working and receives a feedback as soon as the dedicated task has been finished. Thus, the modules do not register at the mission coordinator anymore but are silently considered to be active. Consequently, launching the system will fail on purpose when required modules are not included in the current system configuration.

4.1.1 Interfaces

Global Path Planner This module provides the following data

- Mission goal
- Global topological path

Road Description The mission coordinator gets the road network of the road description module. The data consist of a graph structure including the following items:

- Roads
- Lanes
- Intersections
- Traffic signs
- Traffic lights

Local Ego Provider Here, local odometry data refering to the vehicle state are provided:

- Position
- Orientation
- Velocity
- Acceleration

Global Ego Provider This module provides the vehicle pose with respect to the world. The data are necessary to decide if the target has been reached.

Global Strategy - Explorer

The explorer module is used in a parking mission when the parking spot is not known in advance. It is triggered by the mission coordinator at the beginning of the mission. A suitable strategy will be devised in the course of this project to explore the streets close to the place where the driver has left the car to find free parking spots. Here, an exploration strategy implies to dexterously specify goal positions for the local path planner which avoid to traverse already explored streets several times. Moreover, statistical data about free parking spots along the roads may be used to maximize the success probability and to minimize the exploration time. During exploration, an active parking spot search will be employed and the vehicle will stop as soon as a free parking spot has been found. Then the task of the explorer is finished and the free space planner module (see 5.1) may take over to compute a collision free trajectory to park the vehicle.

4.1.2 Interfaces

Mission Coordinator The explorer exchanges the following data with the mission coordinator:

- Trigger to start the exploration
- Module state to tell the mission coordinator whether the task has been finished successfully or not.

Local Path Planner The explorer sends points of interest to the local path planner in order to initiate the described planning process.

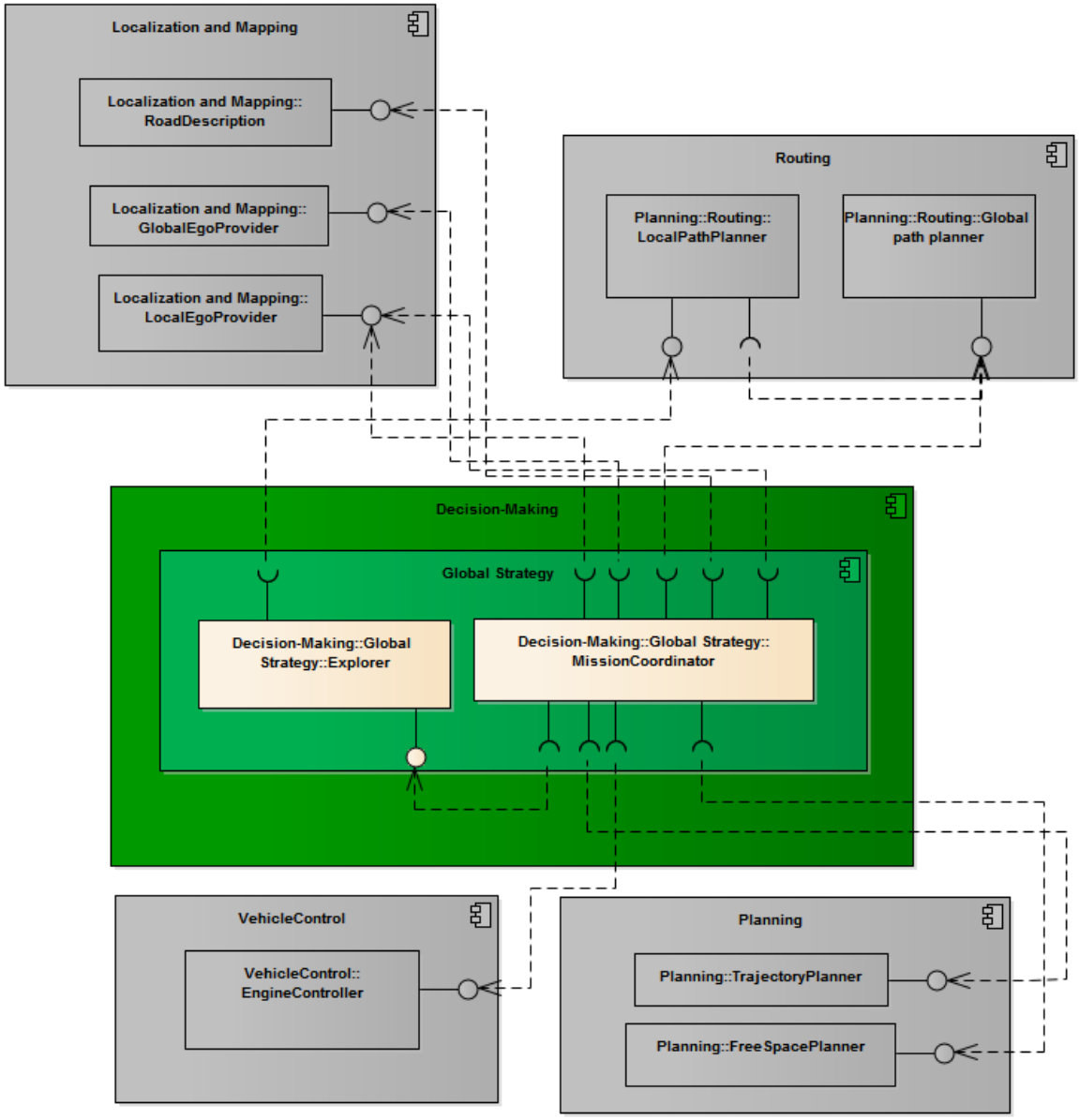

Figure 4: All components involved in computing a global vehicle navigation strategy.

4.2 Local Decision - Scene Collector

The scene collector generates a fundamental part of the input data of the trajectory planner [19]. The data computed here, are called *scene* in the reminder of this deliverable. Since the trajectory planner concepts requires input data given with respect to a Frenet frame, the scene data are transformed accordingly. In this module, relevant data are extracted from the road network, fused with dynamic object data and transformed to the Frenet world. See section 5.1 for more details on what the characteristics of the Frenet world are. Note, that the concept of the scene generation combined with the maneuver planning stage and eventually the trajectory planner has already been used in the EU-funded *InteractIVe* project [11] as well as Adaptive [10].

4.2.1 Interfaces

Road Description One part of the scene is the road network of the road description module. The data consist of a graph structure including the following items:

- Roads
- Lanes
- Intersections
- Traffic signs
- Traffic lights

Local Ego Provider Local odometry data refering to the vehicle state are provided. See section 4.1.1 for more details. The interface is used to add dynamic ego data to the scene.

Object Fusion This module is called object fusion as it fuses data from several sensors to compute a consistent state of all dynamic obstacles currently observed. Here, dynamic obstacles refer to cars, trucks, motorcycles, pedestrians, bicycles,etc.. Thus, it provides all moving traffic participants surrounding the ego vehicle to be added to the scene. The objects are then transformed to the Frenet fame.

Grid The Grid module computes a 2D array whose cells host a probability of being occupied or not. The grid does not contain any dynamic information but only (semi-)static obstacles like curbs or others cars parking along the road. The latter could also be treated as dynamic obstacles featuring a zero motion but currently those obstacles are registered in the grid.

Intention and Prediction This interface establishes the connection between WP 6 and 7 and will provide beyond state-of-the-art traffic information. Using this interface, the dynamic obstacles coming from the object fusion will be enriched with data about their intrinsic motivation, e.g., *Pedestrian with id x intends to cross the street* or *Vehicle with id y is currently parking.* Furthermore, advanced predictive data about the future states of the objects will be added. Note that prediction as defined in WP 6 implies to compute object states in 5 to 10 seconds from now while traditional object tracking considers time ranges of only a few hundred milliseconds. However, understanding what other traffic participants will do are mandatory information for computing sophisticated trajectories in complex inner-city scenarios as intended to address in the *UP-Drive* project [2].

Local Decision - Maneuver Planning

This is the core strategic module for trajectory planning. As described in [19], the trajectory planner needs so-called target points as input and computes a set of trajectories matching the target within given boundaries. Section 5.1 elucidates the target point concept in more detail. The task of the maneuver planning module is to generate target points which are constantly adapted to the current traffic situation. Therefore, this module constitutes the actual intelligence of our current trajectory planning stack. It is an expert system prioritizing different target options based on traffic rules and surrounding objects. The concept

is comparable to the work of Furda *et al.*[4]. Typical target points are *follow lane*, *follow car in front*, *stop at traffic light*, or *perform a lane change*. A hierarchical state machine sorts the target points depending on their priority. E.g., *follow lane* might be the preferred strategy but if there are other cars within the lane of the ego vehicle the trajectories eventually computed perform *follow cars in front* maneuvers. Furthermore, *stop at traffic light* has also a higher priority than *follow lane*. However, the system is currently optimized for highway traffic and city main roads but 30 km/h-scenarios featuring narrow streets often require more complex decisions. In particular this is the case when driving on the opposite lane becomes necessary to pass other vehicles parking along the road. Such scenarios still need to be incorporated into the state machine and also the information about the intention and prediction of other traffic participants has to be considered in the decision process. This is subject to research and will be investigated in the course of this project.

4.2.2 Interfaces

Scene Collector Delivers the scene as described in section 4.2 featuring the local road network with dynamic obstacles including their intention as well as grid data hosting static obstacles.

Global Ego Provider Provides the vehicle pose with respect to the world. The information is necessary to understand where the vehicle is currently located within the road network as basis for further strategic decisions.

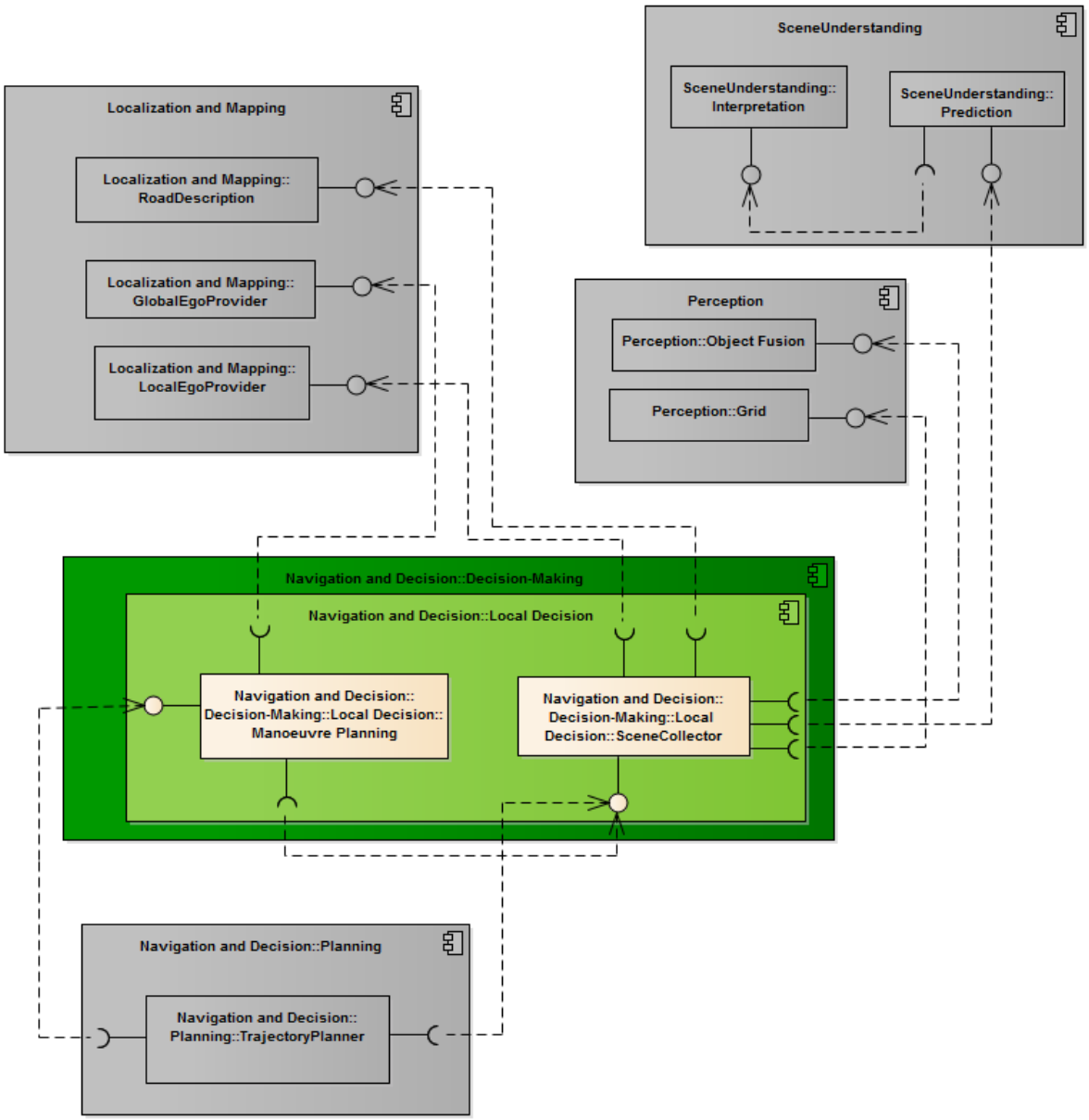

Figure 5: All components involved in computing a strategy to handle local situations .

5 Planning

Currently, two planner modules are used. The first planner generates jerk-optimal trajectories predominantly used in highway and city main roads scenarios. In contrast, the second planner is optimized for free space scenarios, e.g., parking, where the lane direction is not relevant. Both modules are triggered externally by the strategy unit.

5.1 Trajectory Planner

The trajectory planner presented by Werling *et al.*[19] is used to compute collision free trajectories while driving on the road. Planning takes place in Frenet coordinates [1]. The trajectories are jerk-optimal and computed analytically satisfying the side-conditions of a point-mass model. Thus, the approach generates both longitudinal and lateral trajectories $(s(t), d(t))$ per target point. Here, target points are being specified by time, lateral and longitudinal offset limits, and/or velocity. See [19] for more details. Note that both trajectory sets are mutually independent and combined when transforming them back to the real world. The great advantage is that the computation time is very low and that a closed-form solution does exist. However, the trajectory planner does not entail any intelligence on creating the target points. We gained excellent results on highways as the vehicle moves pretty fast in those scenarios and thus most of the combined trajectories are drivable and the invalid assumption of the point-mass model can be neglected.

Of course, this approach clearly has drawbacks as it does not consider the non holonomic mechanics of a real car. As mentioned, this limitation can be safely mitigated in highway scenarios by adding curvature checks after transforming the combined trajectories back to the Euclidean space. The reason is that the longitudinal trajectory makes considerable progress while the lateral distance to the lane varies. However, this assumption does not necessarily hold anymore in low speed city areas. E.g., when passing obstacles standing in the own lane, a significant lateral shift is often required at low speed and thus the vehicle orientation must be considered. It is planned to tackle this problem by treating the reference path as an elastic band which is being pushed away from oncoming obstacles [13]. Curvature constraints can be simply integrated in the gradient descend method performing the path deformation. Hence, the lateral trajectories can be more or less ignored and only the jerk-optimal longitudinal trajectories are executed resulting in smooth and comfortable driving.

Another drawback is the fact that this planner is not capable of planning any cusps, i.e., changes of the driving direction. Consequently, the approach is not suitable for parking. Therefore, we additionally apply a free space planner explained in the next section.

5.1.1 Interfaces

Mission Coordinator The trajectory planner receives its module trigger via this interface and also sends status messages back to the mission coordinator. For instance, if the target address has been reached, the trajectory planner transmits the status *done* and switches to an idle mode.

Maneuver Planning This strategic module provides the target point list sorted by their priority. The planner starts with the target point with the highest priority. The planning procedure is finished as soon as a collision free trajectory connects the current vehicle state with the according target.

Scene Collector All scene data used for collision checking are being accessed by this interface. Moreover, the current ego state is also included. Note, that the scene is already represented in Frenet coordinates. Thus, no further transformation is necessary.

[1] The Frenet frame is defined by the normal vector and the tangent along a reference path. Both vectors are time depending functions.

Trajectory Controller A set of a fixed number of control points is continuously send to the controller. Then the controller selects one of the points as basis to minimize the path deviation and thus the control error.

5.2 Free Space Planner

This planner is used whenever it comes to situations which require to simply stop following the course of the lane, e.g., during parking or u-turns. It is roughly comparable to the hybrid A* planning scheme presented by Dolgov *et al.*[1]. Their method comprises discrete cells where each cell represent a 3D continuous space (x, y, θ). Then a set of so-called motion primitives is defined to explore the search space. Motion primitives are piecewise differentiable maneuvers like *move forward left* or *move straight backward*. Finally, the resulting path is smoothed using a gradient descent approach. Note, that this is a path planner not considering any vehicle dynamics, speed, or acceleration. Therefore, the drivability of the path depends on the current vehicle state and the target velocity.

In contrast, the method used here explores a high dimensional state space encompassing the vehicle pose, velocity, acceleration, and steering angle. The search complexity is handled twofold. A larger number of future vehicle states is being generated per iteration. Dynamic programming is used to keep the number of state candidates manageable. Computational speed is optimized by shifting the state evolution model to a GPU. The latter is mandatory for this algorithm to run on standard computers in real time.

Currently, the free space planner does not have any interfaces to the scene collector or to the maneuver planner but gets all dynamic and static obstacles directly from perception and scene understanding.

However, we assume that the combination of both planners is sufficient to complete all *UP-Drive* scenarios. Eventually, it is desirable to have only one generic path and trajectory planner capable of handling all traffic situations but this is still ongoing research.

5.2.1 Interfaces

Mission Coordinator The interface is the same as for the trajectory planner. See 5.1.1 for more details.

Object Fusion Receives all active dynamic obstacles directly from here.

Grid The grid contains static obstacles considered during planning.

Intention and Prediction Provides the intention of other traffic participants as well as the prediction of their future states.

Trajectory Controller The interface is the same as for the trajectory planner. See 5.1.1 for more details.

Local Ego Provider The free space planner uses local coordinates for planning. Here the odometry frame is used as reference frame. See section 4.1.1 for more details on local ego data.

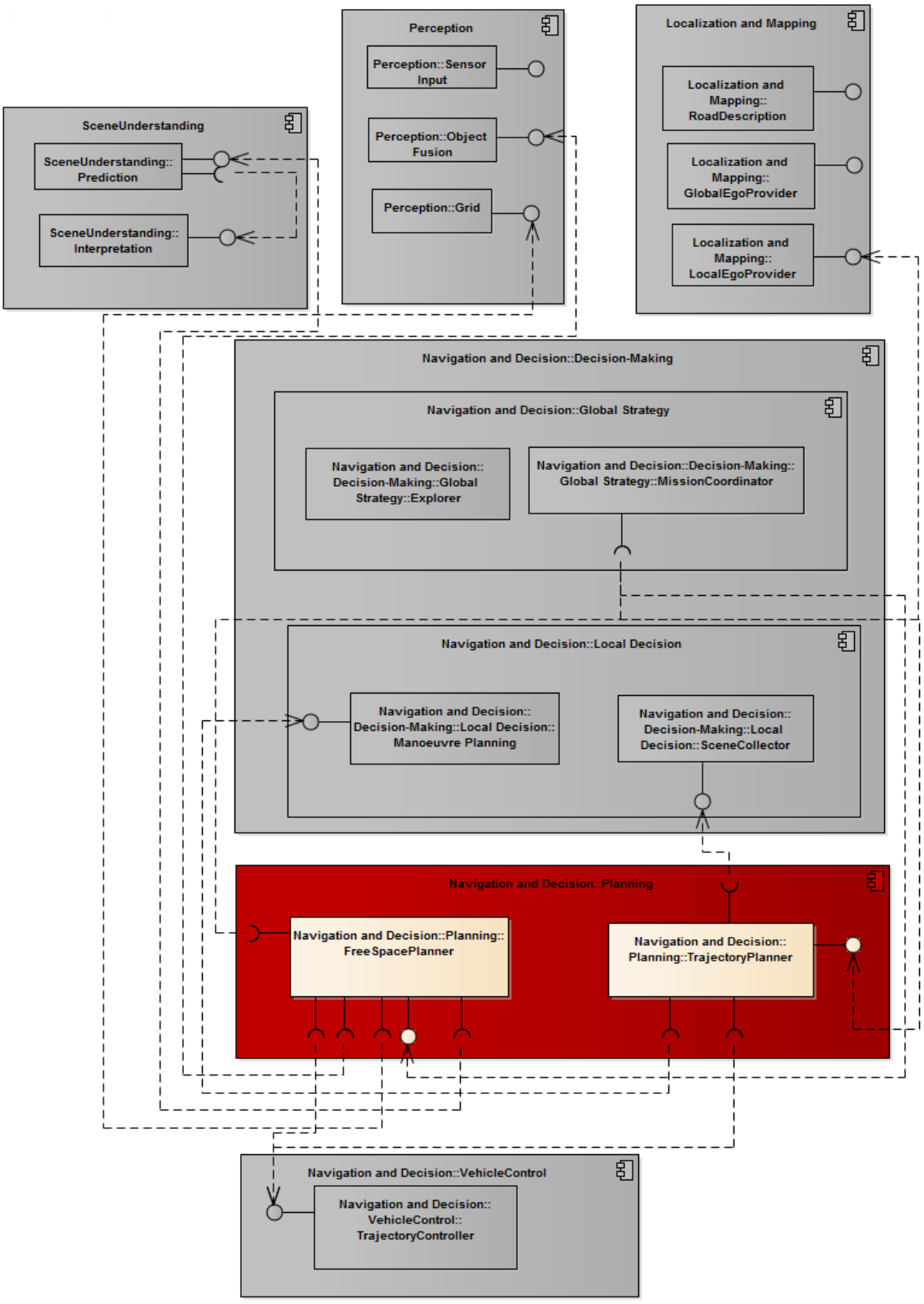

Figure 6: All components involved in computing collision free trajectories resembling human behavior.

6 Vehicle Control

6.1 Trajectory Controller

Similar to the planning stack, we employ two controllers to execute the trajectories generated by the two different planner modules. The control points of the trajectory planner used for lane following are processed by a simple P-controller. The selected control point depends on the distance to the vehicle. Currently, this is a fixed parameter but in the future this parameter will be a function of the vehicle speed.

The execution of the free space planner trajectories is more complex. To this end, the controller of *V-Charge* is used. The following explanation is taken from Furgale *et al.* [5]:

The control parameters for trajectory execution are steering angle and acceleration. To this end, the electric power steering (EPS) interface as well as the ACC interface are employed. The input of the controller is a local reference trajectory. Lateral offset control is performed as follows: given a certain point on the reference trajectory, electric field lines of an electric dipole are simulated guiding the vehicle back to the reference trajectory. The active steering angle results from computing the difference of the vehicle orientation and the direction of the current field line. The active desired velocity is given by considering the maximal velocity on the parking lot and some physical restrictions. A traditional P-controller is applied to control the vehicle velocity via the ACC interface.

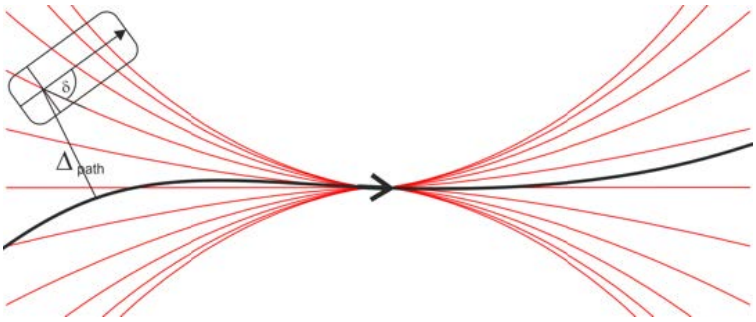

Figure 7: The red lines indicate the simulated electric field lines. δ reflects the difference angle between the vehicle heading and the slope of the field line. Δ_{path} is the current distance to the reference trajectory. Image courtesy of Furgale *et al.*[5].

As we use two controllers, the planners send the desired controller id together with the control points. However, we also intend to fuse both control approaches as we do on the planner side.

6.1.1 Interfaces

Trajectory Planner See 5.1.1 for more details.

Free Space Planner See 5.1.1 for more details.

6.2 Engine Controller

This module is required to activate and to deactivate the vehicle. It is triggered by the mission coordinator at the beginning and at the end of a mission. Depending on the mission goal and vehicle state, a message is send to the engine to switch it on or off.

6.2.1 Interfaces

Mission Coordinator The mission coordinator triggers the engine controller. As soon as the target engine state has been reached, a *done* message is send back.

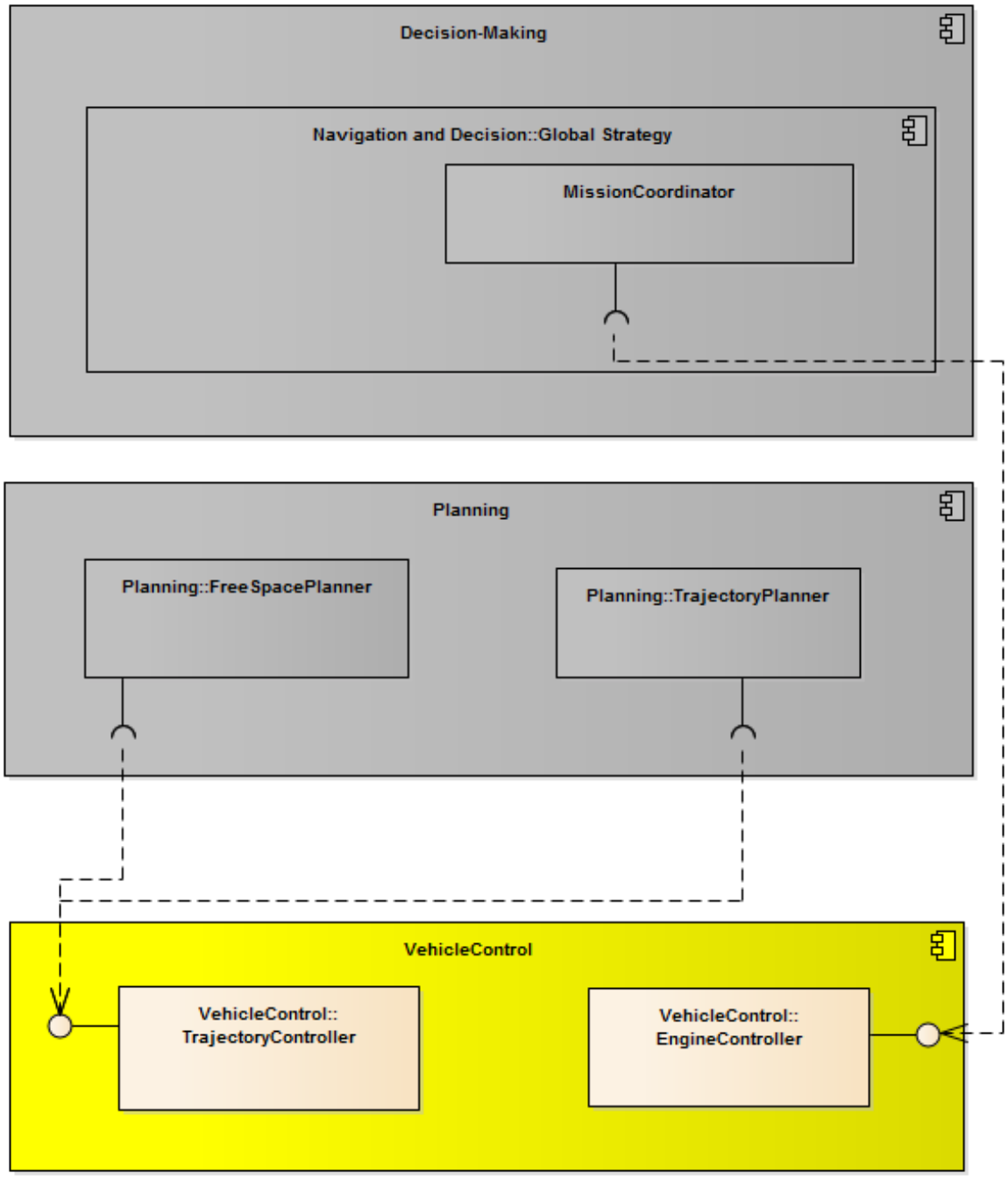

Figure 8: All components involved for trajectory control and vehicle activation/deactivation.

7 Conclusion

In this deliverable an overview of the architecture of the decision-making and navigation workpackage of the *UP-Drive* project has been presented. The architecture has been divided into four layers, namely routing, decision-making, planning and vehicle control. The routing layer is responsible for global and local topological planning based on a-priori known navigation maps. The decision-making layer has been subdivided into to a global strategy and local decision unit. The global part derives a strategy to trigger and control the individual functional modules required to accomplish the mission. In contrast, the local part represents the actual vehicle behavior in dynamic traffic scenarios computing target points for the trajectory planner. Afterwards, the planning stack hosts two trajectory planners using the data generated in the layers on top. Here, the first planner is used in typical traffic scenarios where the higher goal is to follow the road to eventually reach the target address. Predominantly, the second planner is used in parking scenarios. Finally, the vehicle control layer executes the trajectories.

Several research anchors have been identified by analyzing the advantages and drawbacks of the applied methods:

1. The output of WP6, i.e., the intention and prediction of other traffic participants has to be considered in the local decision unit to compute more sophisticated target points for the trajectory planner. This is required in advanced scenarios, e.g., to decide if and where to stop or drive slowly in the presence of oncoming traffic.

2. The trajectory planner proposed by Werling *et al.* [19] does not feature any vehicle model internally. The point-mass model used to generate the longitudinal and lateral trajectories becomes invalid when the vehicle drives slowly (e.g. 10 km/h) and executes sharp maneuvers. Either a strategy needs to be devised to tackle this drawback or another planning method will be investigated.

3. Currently, two trajectory planners are integrated into the system and triggered depending on the traffic scenario. Both approaches have advantages and drawbacks (see sections 5.1 and 5.1. However, the consolidation of both planners is desirable which mitigates the complexity in the system. Furthermore, the resulting planner is much more powerful as it can handle both highway and inner city scenarios at the same time.

4. As explained in section 6 two controllers are employed executing the generated trajectories. The active controller is being selected by the trajectory planner. According to 3, merging the controllers or replacing both approaches by an alternative method is the long term goal.

However, our priority is to put pressure on 1 and 2 as those items will provide much progress in the overall system capabilities and opens up new options to improve the vehicle behavior. In contrast, 3 and 4 make the architecture more elegant and less error prone but the functional benefit is limited.

References

[1] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. Practical search techniques in path planning for autonomous driving. In *Proceedings of the First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR-08)*, Chicago, USA, June 2008. AAAI.

[2] Sergiu Nedevschi et al. Initial version of requirements definition, system architecture and component specification. Deliverable 1.1, Project UP-Drive consortium, May 2016.

[3] W. Derendarz et al. Project UP-Drive, Description of Work, Technical Annex, December 2015.

[4] Andrei Furda and Ljubo B. Vlacic. Enabling safe autonomous driving in real-world city traffic using multiple criteria decision making. *IEEE Intell. Transport. Syst. Mag.*, 3(1):4–17, 2011.

[5] Paul Furgale, Ulrich Schwesinger, Martin Rufli, Wojciech Derendarz, Hugo Grimmett, Peter Mühlfellner, Stefan Wonneberger, Julian Timpner Stephan Rottmann, Bo Li, Bastian Schmidt, Thien Nghia Nguyen, Elena Cardarelli, Stefano Cattani, Stefan Brüning, Sven Horstmann, Martin Stellmacher, Holger Mielenz, Kevin Köser, Markus Beermann, Christian Häne, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, René Iser, Rudolph Triebel, Ingmar Posner, Paul Newman, Lars Wolf, Marc Pollefeys, Stefan Brosig, Jan Effertz, Cédric Pradalier, and Roland Siegwart. Toward Automated Driving in Cities using Close-to-Market Sensors, an Overview of the V-Charge Project. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 809–816, Gold Coast, Australia, 23–26 June 2013.

[6] Sören Kammel, Julius Ziegler, Benjamin Pitzer, Moritz Werling, Tobias Gindele, Daniel Jagszent, Joachim Schröder, Michael Thuy, Matthias Goebl, Felix von Hundelshausen, Oliver Pink, Christian Frese, and Christoph Stiller. Team annieway's autonomous system for the DARPA urban challenge 2007. In *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic, George Air Force Base, Victorville, California, USA*, pages 359–391, 2009.

[7] Isaac Miller, Mark Campbell, Dan Huttenlocher, Aaron Nathan, Frank-Robert Kline, Pete Moran, Noah Zych, Brian Schimpf, Sergei Lupashin, Ephrahim Garcia, Jason Catlin, Mike Kurdziel, and Hikaru Fujishima. *Team Cornell's Skynet: Robust Perception and Planning in an Urban Environment*, pages 257–304. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[8] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun. Junior: The stanford entry in the urban challenge. *Journal of Field Robotics*, 2008.

[9] Benjamin J. Patz, Yiannis Papelis, Remo Pillat, Gary Stein, and Don Harper. *A Practical Approach to Robotic Design for the DARPA Urban Challenge*, pages 305–358. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[10] Adaptive - automated driving. `https://www.adaptive-ip.eu/`, 2014-2016. European Commission funded project, FP7. Accessed: 2016-06-17.

[11] Interactive - accident avoidance by active intervention for intelligent vehicles. `http://www.interactive-ip.eu/`, 2009-2013. European Commission funded project, FP7. Accessed: 2016-07-07.

[12] Automated valet parking and charging for e-mobility (V-Charge). `http://www.v-charge.eu/`, 2011-2015. European Commission funded project, FP7. Accessed: 2016-03-18.

[13] S. Quinlan and O. Khatib. Elastic bands: connecting path planning and control. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, 1993.

[14] Fred W. Rauskolb, Kai Berger, Christian Lipski, Marcus A. Magnor, Karsten Cornelsen, Jan Effertz, Thomas Form, Fabian Graefe, Sebastian Ohl, Walter Schumacher, Jörn-Marten Wille, Peter Hecker, Tobias Nothdurft, Michael Doering, Kai Homeier, Johannes Morgenroth, Lars C. Wolf, Christian Basarke, Christian Berger, Tim Gülke, Felix Klose, and Bernhard Rumpe. Caroline: An autonomously driving vehicle for urban environments. In *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic, George Air Force Base, Victorville, California, USA*, pages 441–508, 2009.

[15] Charles Reinholtz, Dennis Hong, Al Wicks, Andrew Bacha, Cheryl Bauman, Ruel Faruque, Michael Fleming, Chris Terwelp, Thomas Alberi, David Anderson, Stephen Cacciola, Patrick Currier, Aaron Dalton, Jesse Farmer, Jesse Hurdus, Shawn Kimmel, Peter King, Andrew Taylor, David Van Covern, and Mike Webster. *Odin: Team VictorTango's Entry in the DARPA Urban Challenge*, pages 125–162. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[16] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Stanley: The robot that won the darpa grand challenge: Research articles. *J. Robot. Syst.*, 23(9):661–692, September 2006.

[17] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, M. N. Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas M. Howard, Sascha Kolski, Alonzo Kelly, Maxim Likhachev, Matt McNaughton, Nick Miller, Kevin Peterson, Brian Pilnick, Raj Rajkumar, Paul Rybski, Bryan Salesky, Young-Woo Seo, Sanjiv Singh, Jarrod Snider, Anthony Stentz, William "Red" Whittaker, Ziv Wolkowicki, Jason Ziglar, Hong Bae, Thomas Brown, Daniel Demitrish, Bakhtiar Litkouhi, Jim Nickolaou, Varsha Sadekar, Wende Zhang, Joshua Struble, Michael Taylor, Michael Darms, and Dave Ferguson. Autonomous driving in urban environments: Boss and the urban challenge. *J. Field Robot.*, 25(8):425–466, August 2008.

[18] Deliverable 4.3 - motion planning and vehicle control. `http://www.v-charge.eu/deliverables/D4.3.pdf`, 2011-2015. European Commission funded project, FP7. Accessed: 2016-03-18.

[19] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a frenét frame. In *ICRA*, pages 987–993. IEEE, 2010.