Staying in lane on highways with EB robinos
Highly automated driving (HAD) raises the complexity within vehicles tremendously due to many different components that need to be coordinated and combined. A change in current state-of-the-art development procedures is needed to manage this increased complexity. As a result, the use of a software framework that is based on a standardized architecture with open interfaces can reduce this complexity, but also cost and effort. This leads to higher competitiveness and, vice versa, the delivery of better products to the end consumer. The industry needs to shift its focus and also think about best practices in terms of functional software rather than only in terms of hardware architectures.

**EB robinos**

The automotive industry still faces the challenge of how to coordinate and combine separate parts, so that they cover all needs – from functionality to efficiency as well as safety and security. A high degree of complexity results in the need to incorporate elements such as sensors, fusion, function, and control components as well as actuators from the perspective of both hardware and software. EB robinos, an application layer software framework with open interfaces and software modules for highly automated driving, is especially designed to reduce the complexity of highly automated driving and application development. It enables appropriate combination and interaction of all the related elements that form an automated driving system, from development to mass production.

![Figure 1: The EB robinos architecture](image-url)
Staying in lane on highways with EB robinos

EB robinos architecture

EB robinos comprises a software framework based on an architecture with open interfaces and software modules for highly automated driving.

The software architecture, depicted in Figure 1, follows a classical robotic architecture based on the sense-plan-act principle outlined by Boyd, 1976. First, the sensor data is converted into abstracted data structures to render subsequent layers independently of the individual sensor type. It is, thus, easier to replace sensors or enhance the sensor setup with new sensors. This sensor data is processed by different modules in the following sensor data fusion step. This layer consists, for example, of an object hypothesis fusion, a grid fusion module, or a highly accurate positioning module. The results of the sensor data fusion process are shown in function-specific views. These views summarize the fused sensor data to provide a specialized view for the functions that follow. The simplest form has only one functional view for all behaviors. However, this often results in a view that becomes unusable because it contains a lot of information that is not needed by many of the behaviors. The architecture therefore provides different views for different behaviors. This also enables the developer to tune the view to the special needs of a behavior. For example, an emergency brake behavior needs the object hypothesis immediately after the initial detection (e.g. to pre-charge brakes), whereas an adaptive cruise control may require a more stable object hypothesis, since rapid reaction is not so much of an issue. Another advantage of different views is the ability to adapt the amount of data to different system architectures as the performance of communication buses may differ.

The HAD functions are provided by the center part of the architecture. This architecture splits the overall system into small behaviors, each for a specialized situation. This enables the developer to concentrate on a specific task. Moreover, the system can be enhanced over time by the addition of more and more behaviors alongside the original ones. A need to moderate the access to actuators comes up, if multiple vehicle control behaviors are used. This need is fulfilled by situative behavior arbitration. It collects the demands of all behaviors for controlling the car and decides which one or which set of behaviors is allowed to access the actuators.

These commands are then executed by the components of the motion management or the HMI management layers. The system applies vehicle-independent control and optimization algorithms in these layers to execute the requested commands. These vehicle-independent commands are then converted into vehicle/actuator specific commands by the abstraction layer on the right-hand side.

Staying in lane on highways with EB robinos

One of the most important basic functions while driving on highways is to stay in the own lane and adapt the speed according to the vehicles in front. However, if this function can be accomplished with lateral and longitudinal control, one big step towards an automatic highway application is made.

Function definition

In this example the application executes the following function definition:

The driver engages the function while driving on a highway. The vehicle monitors its surroundings and detects other vehicles and lane markings. Afterwards, the vehicle will adapt its speed according to speed limits or to preceding vehicles. Moreover, the vehicle is steered according to the detected lane markings (short term) and the road network of the navigation system (mid-term). Intersections, exits etc. are not handled by the system.

Realization

The function described in previous definition is realized by using a subset of the EB robinos software components, the EB Assist Lane Marking detection, the EB Assist Map Information Toolbox, and the EB Assist Electronic Horizon Reconstructor Toolbox running in the EB Assist ADTF framework.

There are technical and legislative requirements that need to be met:

- The test vehicle is equipped with high precision sensors (e.g. high definition RADAR or LIDAR) which are accessible by a car PC
- The test vehicle sensors’ and actuators’ data is converted to the EB robinos data structures in the abstraction layers as depicted in Figure 2
- The vehicle is driving at a reasonable speed (not very high but also not very low)
Emergency maneuvers are subject to the safety driver. The test vehicle is equipped with a head-up-display to implement a driver-in-the-loop setup and comply with legislative rules.

As depicted in Figure 2, some components of the EB robinos software architecture are used to realize the stay-in-lane-function. The basis for realizing this application is the positioning of the vehicle. Therefore, a (d)GPS position and in-vehicle-sensors (e.g. tire-sensors) are used to estimate the vehicle's position very precisely. Now the system can use the data of the EB Assist Lane Marking Detection to detect lane boundaries. The output consists of the curvature and the width of our current lane. This data is drawn with a sensor model into the Grid Fusion module. The module runs a Dempster-Shafer based grid fusion (Wu et.al. 2002). It represents the vehicles surrounding as cells containing a mass value for drivable, not drivable, and unknown. As the vehicle should stay in its own lane, lane markings are marked not drivable. Hence, the vehicle is put into some kind of cage with an opening at the end. The data of the Grid Fusion is the input for the Path Planning module. It plans from the current vehicle position to some destination. As the path planning is mostly used for short term planning, this destination needs to move along the road of the highway while the vehicle is driving. To determine this target, we use the EB Assist Map Information Toolbox (MIT) and the EB Assist Electronic Horizon Reconstructor Toolbox to get information about the road network. The MIT provides a full automotive grade navigation system which also provides an electronic horizon in the ADASISv2 format. This electronic horizon is reconstructed be the EB Assist Electronic Horizon Reconstructor Toolbox and the current road is used to predict a position a few hundred meters in front.

Once the path to follow the highway has been found, the speed of the vehicle needs to be adapted according to the speed limit or to other vehicles. The speed limits provide an upper limit for the vehicle's target speed. It can be retrieved from the data of the Electronic Horizon Reconstructor very easily. Adapting the target speed according to the preceding vehicle is a bigger challenge as it is not possible to set the target speed to the other vehicles speed. First, it is necessary to decide whether this vehicle is driving in a lane on the vehicle’s path and the distance is small enough to care about it. This is realized by using the planned path from the path planning module. If the preceding vehicle is driving on the same path as the test vehicle, it is a potential object to follow. If the distance is small enough and there is no other object with a smaller distance, this is the target vehicle. The intelligent driver model (Treiber et.al. 2000) is used to adapt the target speed. It adjusts the speed according to the time gap to the target vehicle in a way that is comfortable for the passengers.

Finally, there is a path to follow and a speed value for the test vehicle. These values could be the input for a lateral and a longitudinal controller. However, as the vehicle will be tested on public roads without special precautions, these input values will be displayed to the driver in a head-up display. The safety driver’s will therefore decide whether or not too follow these inputs.
To realize this application, we used “Valerie”, one of EB’s highly automated driving (HAD) test vehicles (Figure 3). This is a regular production vehicles additionally equipped with a high precision dGPS receiver, surround view cameras (not used for this application), ultrasonic sensors, and IBEO Lux LIDAR sensors. Moreover, it also has actuators for disabled people that provides fully drive by wire features to the control PC.

The data from the lane marking detection module is fed into the grid fusion module. As you can see in Figure 4, the sensor model for the lane markings is updating the grid with a very low certainty. Therefore, in the edge of the detection rage it is the markings are very light. However, they are updated repeatedly and the values become very certain within a reasonable distance to the test vehicle.

The grid data is then used to plan the path to the destination derived from the data of the EB Assist Map Information Toolbox and the EB Assist Electronic Horizon Reconstructor Toolbox (see Figure 5, left). Depicted of the right part of Figure 5, a planned path is shown. It starts at the vehicles position, follows the defile of the lane markings in the grid, and moves a little to the left as the certainty of the lane markings are not significant enough that the vehicle should not pass the borders. The path is definitely adjusted as the lane markings become clearer.
As the direction in which to drive has now been determined, the speed is adjusted according to other vehicles. In this example, the object hypotheses generated by the IBEO lux sensor is used directly. Figure 6 shows an example where the test vehicle has detected a preceding car and has selected it as the target vehicle.

Figure 6: Detected vehicle (red dot) to adapt speed

All the necessary input data for the stay-in-lane-function is now complete and the results can be presented to the driver. Figure 7 shows the simple head-up display. It shows acceleration as a horizontal bar. If it is aligned to the circle in the middle, the driver drives the intended velocity. The bar will move down to deaccelerate and upwards to accelerate. The vertical bar is used to present the lateral actuating variable. If it is aligned to the circle, the driver should hold the steering wheel as it is. Moving to the right means steering to the right side and moving to the left means the other way around.

Figure 7: Head-up display used to show the actuating variables to the driver

This technical paper shows how EB robinos can be used to implement a stay-in-lane on highways application. After a short introduction into EB robinos, the function definition of the stay-in-lane example is described and the basic algorithms and components of the application are described. Finally, one of EB’s HAD test vehicles is introduced, and the implementation of the stay-in-lane-application in this vehicle is described (see Figure 8).

Figure 8: Overall application driving on a highway

EB robinos accelerates the HAD development by applying ready to use software components. HAD development is made easier by the correct combination and configuration of the available software components with well-defined interfaces. The EB robinos approach enables car manufacturers to focus on the development of end consumer driving experience and HAD functions.

Summary

Bibliography

Staying in lane on highways with EB robinos

Author:
Dr.-Ing. Sebastian Ohl
Senior Expert Driver Assistance
About EB Automotive

Elektrobit (EB) is an award-winning and visionary global supplier of embedded and connected software solutions and services for the automotive industry. A leader in automotive software with over 25 years serving the industry, EB’s software powers over 70 million vehicles and offers flexible, innovative solutions for connected car infrastructure, human machine interface (HMI) technologies, navigation, driver assistance, electronic control units (ECUs), and software engineering services. EB is a wholly owned subsidiary of Continental.