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 standardized architecture with open interfaces can reduce this complexity, but also cost and effort. This leads to higher competitiveness and, in turn, the delivery of better functions 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 architecture.

**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 up to efficiency to safety and security. A high degree of complexity results from 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 architecture with open interfaces and software modules for highly automated driving, is especially designed to reduce complexity of highly automated driving and application development. It enables the appropriate combination and interaction of all the related elements that form an automated driving system, from development to mass production.
EB robinos architecture

EB robinos comprises a functional architecture with open interfaces and software modules for highly automated driving – from development to mass production.

The software architecture follows a classical robotic architecture based on the sense-decide-act-principle outlined by Boyd, 1976. The sensor data first 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. 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 function view for all functions. However, this often results in a view that becomes unusable because it contains a lot of information that is not needed by all the functions. The architecture therefore provides for different views for different functions. This also enables the developer to tune the view to the special needs of a function. For example, an emergency brake function needs the object hypothesis immediately after the initial detection (e.g. to pre-charge the 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 where 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 many smaller functions, 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 functions along with the original ones. A need to moderate this access is created when multiple vehicle control functions are used. This need is fulfilled by situative behavior arbitration. This collects the demands for controlling the car from all the functions 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.

EB robinos Positioning

A classical navigation system positioning function provides the absolute position in a global coordinate system such as the WGS84 geodetic reference system. This positioning function is suitable for navigation applications in order to determine the current vehicle position on the map database and to perform tasks such as route guidance or electronic horizon compilation. (An electronic horizon distributes information about the course of the road in front of the vehicle to predictive driver assistance features.) Reliable position and movement information is essential for highly automated driving (HAD). High accuracy and a smooth trajectory are essential for the safe execution of lane change functions or parking maneuvers, for example. But position and movement information can also be used differently by other applications. An automated valet parking application (Automated Valet Parking with EB robinos) only needs the absolute position in a global coordinate system to locate the correct parking spot. However, smooth and accurate tracking of the relative movement of the vehicle since the start of the parking maneuver is crucial for the parking procedure. The requirements of this application are better met by a local coordinate system.

EB robinos Positioning addresses the needs of both these kinds of application, as it provides position information in a global and in a local coordinate system simultaneously.

Global versus local position

EB robinos Positioning takes the input from several sensors to obtain a fused position. The different sensor types can be grouped into three categories:

1. Localization sensors: These sensors provide information about the absolute position in a global coordinate system. The most prominent example of this is GNSS (Global Navigation Satellite System, with GPS being the best-known).
2. Exteroceptive sensors: These sensors use the surroundings of the vehicle to measure, e.g. its movement relative to detected features. Examples of such sensors are cameras, RADAR, or LIDAR devices.

3. Interoceptive sensors: These sensors measure the movement of the vehicle from within. In other words, no environment information is required, unlike the other two sensor categories. Examples of such sensors are gyroscopes, accelerometers, or odometers.

The relative movement of the vehicle can already be calculated by using interoceptive sensors only. EB robinos Positioning utilizes interoceptive sensor data to provide a local position that is not dependent on the availability of sensor data from the other two sensor categories. Applications like automated valet parking make use of this kind of position output, as mentioned above. The advantage of the local coordinate system position data is that it describes smooth motion relatively accurately without the typical correctional jumps of incoming GNSS sensor measurements. Furthermore, the data is available immediately on start-up with no need to wait for a GNSS fix, for example. Local position output is actually available even without any GNSS/localization sensor input.

The following table provides an overview of the position data available in the local coordinate system:

<table>
<thead>
<tr>
<th>Position</th>
<th>Latitude, longitude, and altitude on the WGS84 ellipsoid, with estimated accuracy information.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velocity</td>
<td>Velocity vector in east, north, and up-directions, with estimated accuracy information.</td>
</tr>
<tr>
<td>Acceleration</td>
<td>Acceleration vector in east, north, and up-directions, with estimated accuracy information.</td>
</tr>
<tr>
<td>Orientation</td>
<td>Roll, pitch, and yaw angles, with estimated accuracy information.</td>
</tr>
<tr>
<td>Rotation Rates</td>
<td>Roll, pitch, and yaw rates, with estimated accuracy information.</td>
</tr>
</tbody>
</table>

Table 2: Position data in the global coordinate system

EB robinos Positioning can provide an absolute position when a localization sensor is available. This type of position information is referred to a global coordinate system and enables navigation-like applications. While the local position inherently drifts over time with the integration of sensor measurements, the global position has the advantage that it is constantly corrected by the localization sensor.

The following table shows the position data available in the global coordinate system:

<table>
<thead>
<tr>
<th>Position</th>
<th>Latitude, longitude, and altitude on the WGS84 ellipsoid, with estimated accuracy information.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velocity</td>
<td>Velocity vector in east, north, and up-directions, with estimated accuracy information.</td>
</tr>
<tr>
<td>Acceleration</td>
<td>Acceleration vector in east, north, and up-directions, with estimated accuracy information.</td>
</tr>
<tr>
<td>Orientation</td>
<td>Roll, pitch, and yaw angles, with estimated accuracy information.</td>
</tr>
<tr>
<td>Rotation Rates</td>
<td>Roll, pitch, and yaw rates, with estimated accuracy information.</td>
</tr>
</tbody>
</table>

Table 1: Position data in the local coordinate system

The screenshot below shows the outcome of simultaneous local and global positioning. Google Earth is used for visualization:

Figure 2: Test drive in a parking garage at Nuremberg Airport, Germany, visualized using Google Earth. Global position output (red track) compared to local position output in 3D (yellow track, embedded into global frame) and local position output in 2D (violet track, embedded into global frame). The green check marks are GNSS input positions.
EB robinos Positioning

Corrections that are made by using the GNSS input can be seen in the ground track of the absolute global position shown in red in Figure 2. The corrections are at two locations in particular: one on the roof of the parking garage, and one when leaving it (marked by the white arrows). These corrections do not appear in the ground track of the local position output (violet). However, the slight drift in the local position over time is clearly visible in the track after leaving the parking garage.

Sensor calibration

Although interoceptive sensors respond to specific physical quantities, the response is not usually reported as a value in physical units. Mathematical transformations are necessary before the data can be used to estimate the position. The applied sensors are calibrated for this purpose. Most of the applied sensors have two significant parameters that describe the relationship between the sensor measurement and its value in the corresponding physical units:

- **Scale**: This calibration parameter describes the change in the sensor measurement when the measured physical value changes by one unit.

- **Bias**: This calibration parameter describes the sensor measurement when the measured physical unit value is zero.

These parameters do not remain constant, either in the short term (e.g., they are often temperature dependent) or over the life cycle of the sensor. Additionally, the integrating nature of the fusion process means that small deviations in these parameters lead to large errors in the fused output. For high accuracy applications, all sensor calibration parameters must therefore be estimated carefully and updated continuously while the system is running.

The classical approach of using one and the same Kalman filter to estimate the sensor parameters together with the position data (e.g., see [Lutz 2008]) occasionally shows surprising behavior. The Kalman filter tracks the interdependence between the sensor parameters and the position data. Thus, the Kalman filter will also adapt the position data during the correction step when the sensor parameters are determined. The correction includes correcting for accumulated position uncertainty that results from past uncertainty in the sensor parameters. This behavior is excellent when the best possible position estimation for a single point in time is required. The disadvantage is that smoothness and plausibility can be compromised when a sequence of position estimations is interpreted as a trajectory. The screenshot below (visualized in Google Earth) shows this effect:

![Figure 3: Example of mutual dependency of position and calibration parameters when estimated together. The timestamps are in milliseconds. (Test drive with no GNSS input near Erlangen, Germany, visualized using Google Earth.)](image)

The position shown in Figure 3 clearly jumps by several hundred meters in just 200 milliseconds although the vehicle was standing still at the time. This effect is caused by using one filter to estimate the yaw gyro bias together with the position. The estimated position accuracy becomes very low after the vehicle is driven for about an hour without an update step in the Kalman filter (the covariances are high due to the accumulation of process noise). In this scenario, the difference between the initially estimated and the real yaw gyro bias is significant. The Kalman filter performs an adjustment of the estimated yaw gyro bias when stationary by accepting new information from the outside. The new information affects the position estimate because of the low position accuracy and the coupled nature of the estimation, which results in the behavior shown.

EB robinos Positioning therefore keeps the sensor calibration component separate from the position estimation component to prevent such effects. Figure 4 shows this as a component model:

![Figure 4: Component model of EB robinos Positioning core components](image)
The raw sensor measurements are input to the SensorCalibration component, which estimates the sensor calibration parameters and outputs calibrated sensor measurements. These measurements are fed into separate local and global position estimation components.

The SensorCalibration component needs to adapt to the set of sensors in the system, since the number of different sensors and their possible combinations is huge. A configuration component provides the information that is necessary for adaption, such as which sensors are available and what their nominal calibration parameters are (used as the initial values for the online calibration), and the rate at which they are available for measurements.

As well as splitting the functions, this scheme allows EB robinos Positioning to run purely as a sensor calibration component. Sensor calibration and position estimation can even be run in different contexts and on different ECUs.

**Accuracy monitoring**

The accuracy of the estimated position output is one important factor that ensures the quality and reliability of a positioning component. Accuracy in this context means the observed total deviation of the estimated position output from the true position. The sources of deviations are many; some are systematic and others are of a probabilistic nature. Each sensor exhibits a set of impairments with regard to measurement accuracy, which range from measurement noise to a non-ideal scale, biased values, cross-sensitivity, non-linear scaling effects, and many others. Measurements (especially GNSS input) can be affected by error conditions that are hard to detect, such as multipath effects. Sampling rates and delays in communication or measurement are factors that are also crucial to the positioning accuracy.

As well as the sensor measurements, the fusion itself introduces deviations into the output. The movement model is always described ideally. There will be differences therefore between the model and the real movement of the vehicle. This situation causes some maneuver-dependent perturbations in the accuracy of the fused output. Analytical approaches to predicting the expected accuracy of the positioning component are not meaningful because they require many assumptions to be made about the sensors and the upcoming driving maneuvers.

EB robinos Positioning therefore determines the accuracy empirically. A reference system with high-grade sensors providing good accuracy is used as the source of the true position. Figure 5 shows the highly accurate reference system mounted in a test car.

The test car is also equipped with an in-car PC running EB Assist ADTF (Automotive Data and Time-Triggered Framework). This framework is able to record sensor and reference data using the CAN interfaces of the reference system and of the vehicle. Test drives throughout Europe have been performed with this vehicle to record data for many different test scenarios in a list of representative use cases that had been collected in advance. The test data is used mainly to quantify the accuracy, and also for development. Figure 6 shows a screenshot of the project to replay recorded test data with EB Assist ADTF for development purposes: The violet blocks are the EB robinos Positioning sensor calibration and position estimation filters. The surrounding blocks are needed to play back the recorded test drives and for visualization purposes. EB additionally developed a set of scripts to extract the test data from a full test suite and pass it through the EB robinos Positioning components offline. The estimated position output is compared to the corresponding reference and the deviations are aggregated into several statistical views. A statistical report is generated from this. Figure 7 shows some examples.
The plots on the left show the results of comparing the horizontal position and the yaw angle of the global positioning output against the reference. The plots on the right show the corresponding results for the local positioning output with a free-wheeling interval of ten seconds. The term free-wheeling is defined as follows: The local position is embedded into the global reference frame at a point in time. Fusion is performed for a certain time interval, and then the result is compared to the reference position after the same time interval. The accuracy of the full system cannot be expressed by a single number. Besides the absolute position error, other aspects need to be evaluated. These aspects include the yaw angle, vehicle movement, and longitudinal/lateral errors. The generated report includes plots that show the results for specific driving maneuvers such as braking or driving through curves, as some applications are interested in these special situations.

As already mentioned in the introduction to this section, all accuracy measurements must be seen in the context of the sensors used. The generated report therefore also contains a section that describes the sensor properties. When the original test drive data was used this section describes the properties of the highly accurate reference system sensors. The automotive sensors typically used for the mass market have a lower measurement quality. EB robinos Positioning can also create reports with artificially degraded sensor measurements of the test drives in order to predict the position accuracy for other potential sets of sensors.
Additional features

EB robinos Positioning provides some additional features. The data from the sensors arrives at the positioning module with different communication and measurement delays because the sensors are typically distributed throughout the vehicle. All the sensor measurements must be arranged in chronological order of the time when the measurement was performed for the fusion to be consistent. An optional sorting component provides this service.

The extrapolation component is another important feature. This component estimates a position in the near future based on the current vehicle movement. This position can be used to compensate for delays in the fusion itself and in the application uses the output. The extrapolation component can also be used to estimate the position of the vehicle in the past. This feature is useful for exteroceptive sensors that have a high computation delay between performing the measurement and the time when the measurement result is available, for example. Sensed objects can be associated with their positions by reconstructing the vehicle’s position at the moment of measurement with the aid of the extrapolation component.

EB robinos Positioning provides global and local position outputs for different types of application, as described in this EB tech paper. The output parameters are described in Table 1 and Table 2. Smoothness in the local output trajectory is achieved by separating the sensor calibration from the position estimation. This EB tech paper also presents an empirical approach to measuring and monitoring the accuracy of the output that compares the generated output with a highly accurate reference positioning system, and creates a report from this evaluation.

EB robinos Positioning accelerates 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.

Bibliography

www.eb-robinos.com


Authors:

Evgeni Dobrev
Elektrobit Automotive GmbH

Martin Fotta
Elektrobit Automotive GmbH

Florian Janßen
Elektrobit Automotive GmbH

Fabian Kirsch
Elektrobit Automotive GmbH

Martin Schauer
Elektrobit Automotive GmbH
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.