Verification of Driver Assistance Systems in the Vehicle and in the Laboratory

Driver assistance systems acquire the vehicle’s environment via a wide variety of sensors. Warnings to the driver or (semi-)autonomous interventions in the driving situation are always made based on correct results of the object recognition algorithms. This article addresses the typical challenges that arise in verifying object data and testing the image processing algorithm. The XCP standard enables the necessary high data throughput in measurement and calibration.

Behind the wheel, humans acquire information about their environment via their sensory organs – specifically their eyes and ears. Signal processing in the brain interprets the collected information, decisions are made, and actions are initiated. Decisions might include whether a space on the side of the road is large enough for parking or whether the distance to the car ahead needs to be adjusted. Driver assistance systems (Advanced Driver Assistance Systems or "ADAS") support the driver in making these decisions, thereby enhancing safety and improving comfort and convenience as well as economy.

Access to Sensor and Algorithm Data
Driver assistance systems must be able to reliably detect the environment as a type of "attentive passenger". Radar, ultrasonic and video sensors are very often used to provide information to ECUs on the driving situation or the vehicle's environment. Complex algorithms process the sensor data to detect objects such as road signs, parking vehicles, other participants in traffic, etc., and they initiate actions. To verify the sensor system, it may be sufficient to simply measure the results of the algorithm and compare them to reality. An example here is the distance measuring radar of an Adaptive Cruise Control system: The sensor detects objects by return reflections of the radar beam. The ECU supplies range distance information for each object as coordinates. In this case, it is not necessary to acquire all of the radar reflections in the sensor. However, all input variables of the algorithm must be measured if the data is being logged for later stimulation in the laboratory, for example. In this case, over 100,000 signals with a data rate of several megabytes per second would not be atypical.

Image processing ECUs with video sensors are used for road sign detection systems or lane-keeping assistants. An algorithm analyzes the video images and detects road signs or lane markings. One typical requirement for data pro-
processing in the ECU is a high level of microcontroller performance. On the other hand, whether the sensor data originates from a video or radar system has little impact on measurement instrumentation requirements – a high-performance solution is essential for transporting the measurement data. In evaluating and optimizing the algorithms, the measurement instrumentation must be able to acquire all of the algorithm's input and output variables and all necessary intermediate variables within the algorithm without incurring additional controller load (Figure 1).

Serial bus systems such as CAN and FlexRay run into their performance limits in terms of the necessary data throughput rates. Therefore, controller-specific interfaces such as Nexus, DAP or Aurora are used to transport the large quantities of measurement data. It makes sense to rely on established and proven standards to avoid having to develop a separate solution for each technical measurement task. The VX1000 measurement and calibration hardware from Vector is ideal for this; it transfers the data from the controller interface to a base module via a small PC-board (plug-on device or POD), where it converts it to the standardized XCP on Ethernet; it then transfers the data stream to the PC at a high throughput rate [1].

Validating Sensor Data with Reality

The ECU's object detection results must now be verified against reality. Is the distance to the vehicle ahead on the road actually 45.5 meters? To compare the sensor data with reality, it is first necessary to acquire that reality. A camera, which is independent of the sensor system, records the driving situation. Developers can now quickly and reliably verify the object detection algorithms of their ECUs by comparing the objects detected by the ECU with the video image. The CANape Option Driver Assistance measurement and calibration software from Vector is used to overlay the object data on the video image. This lets developers determine exactly where something was detected and whether what was detected makes sense. In Figure 2, an "X" can be seen in the image at each point representing data obtained by the sensor. Coordinates detected by the sensor, such as the distance ahead and angle to the side, are converted to pixel coordinates of the video image in the PC.

Approving and Optimizing Algorithms

If deviations occur while comparing detected objects with reality, then the algorithm needs to be optimized. This is done by modifying the calibration parameters of the system, and it requires that the calibration parameters be defined in the code such that they are located in RAM at runtime and can be changed by a write access. The mechanisms of the XCP measurement and calibration protocol [2] are available for calibrating these parameters. At runtime, the developer modifies the parameter values and gets immediate feedback on the effects. XCP is not limited to use in the ECU. For example, the algorithm could also be run as a virtual ECU in the form of a DLL on the PC. Calibrations and measurements are also made over XCP – which makes the PC a rapid prototyping platform.

What is the most convenient way to incorporate an XCP driver in a DLL? How are the input data linked to the DLL? In the case of a Simulink based development, the "Simulink Coder" from MathWorks is used to generate the code for different target platforms from the model. The CANape tool from Vector might be specified as such a target platform. In the process of generating the code for CANape, an XCP driver is automatically integrated. At the end of this process, there is a DLL with an XCP driver and an ECU description file in A2L format. Both are integrated in CANape, and the input and output ports of the DLL are linked to real data. At the measurement start, CANape transmits the measured sensor data as an input vector to the algorithm, and the virtual ECU computes the results. The calibration process in the ECU is a high level of microcontroller performance. On the other hand, whether the sensor data originates from a video or radar system has little impact on measurement instrumentation requirements – a high-performance solution is essential for transporting the measurement data. In evaluating and optimizing the algorithms, the measurement instrumentation must be able to acquire all of the algorithm's input and output variables and all necessary intermediate variables within the algorithm without incurring additional controller load (Figure 1).

Serial bus systems such as CAN and FlexRay run into their performance limits in terms of the necessary data throughput rates. Therefore, controller-specific interfaces such as Nexus, DAP or Aurora are used to transport the large quantities of measurement data. It makes sense to rely on established and proven standards to avoid having to develop a separate solution for each technical measurement task. The VX1000 measurement and calibration hardware from Vector is ideal for this; it transfers the data from the controller interface to a base module via a small PC-board (plug-on device or POD), where it converts it to the standardized XCP on Ethernet; it then transfers the data stream to the PC at a high throughput rate [1].

Validating Sensor Data with Reality

The ECU’s object detection results must now be verified against reality. Is the distance to the vehicle ahead on the road actually 45.5 meters? To compare the sensor data with reality, it is first necessary to acquire that reality. A camera, which is independent of the sensor system, records the driving situation. Developers can now quickly and reliably verify the object detection algorithms of their ECUs by comparing the objects detected by the ECU with the video image. The CANape Option Driver Assistance measurement and calibration software from Vector is used to overlay the object data on the video image. This lets developers determine exactly where something was detected and whether what was detected makes sense. In Figure 2, an "X" can be seen in the image at each point representing data obtained by the sensor. Coordinates detected by the sensor, such as the distance ahead and angle to the side, are converted to pixel coordinates of the video image in the PC.

Approving and Optimizing Algorithms

If deviations occur while comparing detected objects with reality, then the algorithm needs to be optimized. This is done by modifying the calibration parameters of the system, and it requires that the calibration parameters be defined in the code such that they are located in RAM at runtime and can be changed by a write access. The mechanisms of the XCP measurement and calibration protocol [2] are available for calibrating these parameters. At runtime, the developer modifies the parameter values and gets immediate feedback on the effects. XCP is not limited to use in the ECU. For example, the algorithm could also be run as a virtual ECU in the form of a DLL on the PC. Calibrations and measurements are also made over XCP – which makes the PC a rapid prototyping platform.

What is the most convenient way to incorporate an XCP driver in a DLL? How are the input data linked to the DLL? In the case of a Simulink based development, the "Simulink Coder" from MathWorks is used to generate the code for different target platforms from the model. The CANape tool from Vector might be specified as such a target platform. In the process of generating the code for CANape, an XCP driver is automatically integrated. At the end of this process, there is a DLL with an XCP driver and an ECU description file in A2L format. Both are integrated in CANape, and the input and output ports of the DLL are linked to real data. At the measurement start, CANape transmits the measured sensor data as an input vector to the algorithm, and the virtual ECU computes the results. The calibration process in the ECU is a high level of microcontroller performance. On the other hand, whether the sensor data originates from a video or radar system has little impact on measurement instrumentation requirements – a high-performance solution is essential for transporting the measurement data. In evaluating and optimizing the algorithms, the measurement instrumentation must be able to acquire all of the algorithm's input and output variables and all necessary intermediate variables within the algorithm without incurring additional controller load (Figure 1).

Serial bus systems such as CAN and FlexRay run into their performance limits in terms of the necessary data throughput rates. Therefore, controller-specific interfaces such as Nexus, DAP or Aurora are used to transport the large quantities of measurement data. It makes sense to rely on established and proven standards to avoid having to develop a separate solution for each technical measurement task. The VX1000 measurement and calibration hardware from Vector is ideal for this; it transfers the data from the controller interface to a base module via a small PC-board (plug-on device or POD), where it converts it to the standardized XCP on Ethernet; it then transfers the data stream to the PC at a high throughput rate [1].

Validating Sensor Data with Reality

The ECU’s object detection results must now be verified against reality. Is the distance to the vehicle ahead on the road actually 45.5 meters? To compare the sensor data with reality, it is first necessary to acquire that reality. A camera, which is independent of the sensor system, records the driving situation. Developers can now quickly and reliably verify the object detection algorithms of their ECUs by comparing the objects detected by the ECU with the video image. The CANape Option Driver Assistance measurement and calibration software from Vector is used to overlay the object data on the video image. This lets developers determine exactly where something was detected and whether what was detected makes sense. In Figure 2, an "X" can be seen in the image at each point representing data obtained by the sensor. Coordinates detected by the sensor, such as the distance ahead and angle to the side, are converted to pixel coordinates of the video image in the PC.
In a virtual ECU, stimulation involves streaming a logged video or signals from measurement files to an input port in CANape. In real ECUs, the physical interfaces must be considered. In video systems, for example, the video sensor signals can be routed to a monitor on which a logged traffic situation is running. The ECU is always stimulated in the same way by using the same videos or signals – and this assures reproducibility of the data. Any changes in the behavior of the algorithm are then exclusively a result of calibration and not of changed input vectors. In both virtual and real ECUs, stimulation is not limited to feeding data to inputs; necessary states and preconditions can also be set over XCP.

**Stimulation with Sensor Data**

Developers of sensor systems are confronted with two problems:

- Meaningful, realistic data from a sensor is often only available in the vehicle; the necessary environment is lacking in the laboratory.
- Achieving reproducibility of sensor data requires tremendous effort.

For these reasons, stimulation of ECUs with previously logged sensor data is a key component in development – whether it involves a real or virtual ECU. The data may be written directly to the ECU’s memory, circumventing the inputs, and the VX1000 System provides the necessary bandwidth. Or the data may be transported into the ECU via its sensor inputs (Figure 3).

In a virtual ECU, stimulation involves streaming a logged video or signals from measurement files to an input port in CANape. In real ECUs, the physical interfaces must be considered. In video systems, for example, the video sensor signals can be routed to a monitor on which a logged traffic situation is running. The ECU is always stimulated in the same way by using the same videos or signals – and this assures reproducibility of the data. Any changes in the behavior of the algorithm are then exclusively a result of calibration and not of changed input vectors. In both virtual and real ECUs, stimulation is not limited to feeding data to inputs; necessary states and preconditions can also be set over XCP.

**Summary**

Optimal calibration of ECUs requires a great deal of effort. Measurement and calibration tools communicate with the ECUs, and this makes code instrumentation unnecessary. Processes are defined for generating A2L description files and much more. However, all of these activities are kept parameters are optimized in the same way as in a real ECU. A C++ project supplied with CANape leads to the same result as manually written code.
independent of the ECU’s tasks. XCP is a standardized solution here; it is well-suited for all types of ECUs. Although driver assistance systems may involve special requirements in terms of data volume and performance, the use of existing tools based on XCP – such as CANape and devices of the VX1000 product line-up – is a convenient solution for ADAS ECUs too. They represent a natural step in the advanced development of existing solutions that can be seamlessly integrated into existing development processes – from the support of video data to the use of a rapid prototyping platform to develop image processing algorithms.

Translation of a German publication in ATZ elektronik, 10/2014

Image rights:
Vector Informatik GmbH

Links:
Vector Informatik GmbH: www.vector.com
Product information CANape:
www.vector.com/vi_canape_en.html
Product information CANape Option Driver Assistance:
www.vector.com/vi_canape_en.html
Product information VX1000:
www.vector.com/vi_vx1000_en.html

Andreas Patzer
graduated with a major in Electrical Engineering at the Technical University of Karlsruhe. Focal points of his studies were measurement and control engineering as well as information and industrial engineering. In 2003, he moved to Vector Informatik GmbH in Stuttgart, where he is team leader for Customer Relations and Services in the Measurement & Calibration product line.