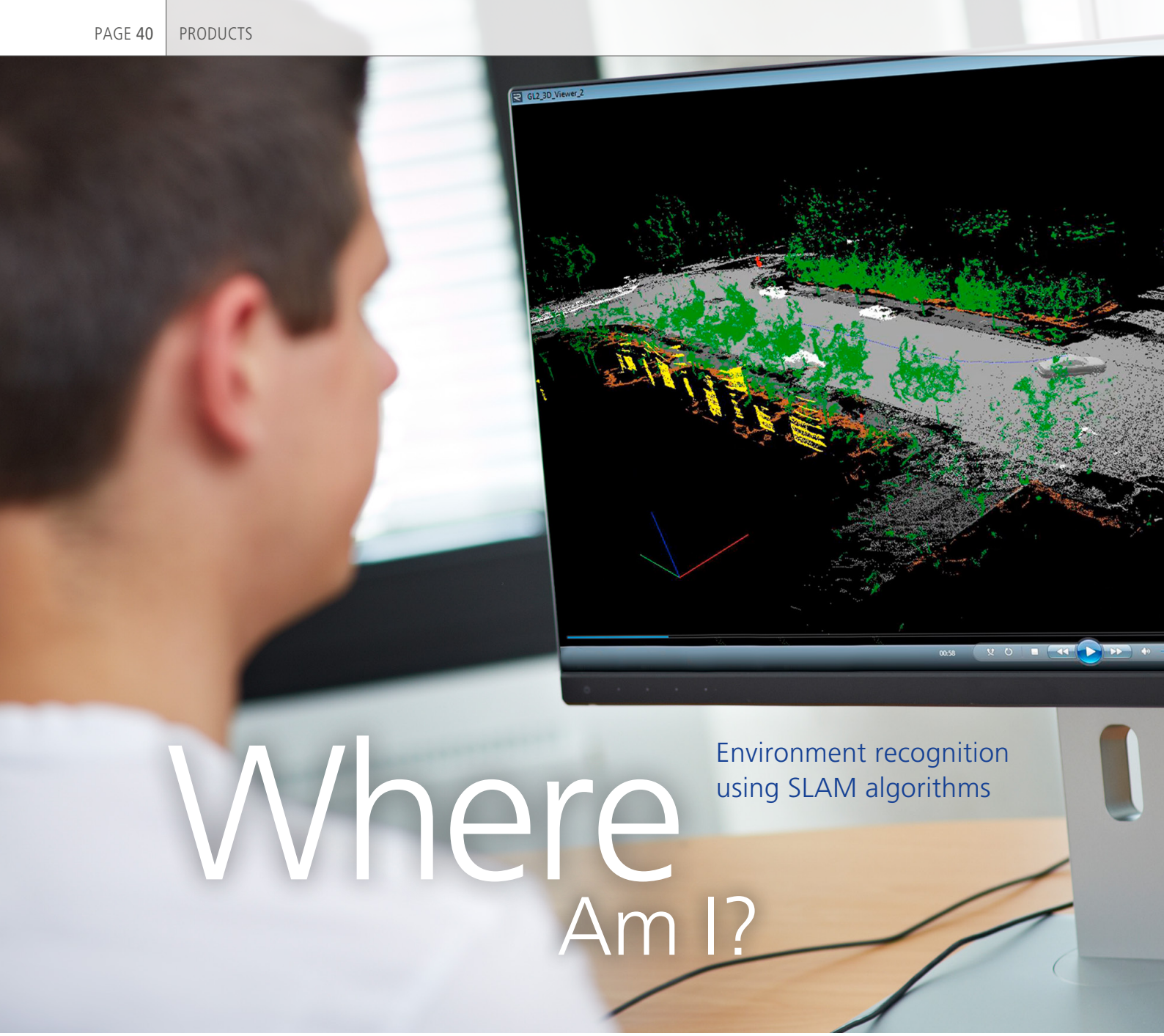
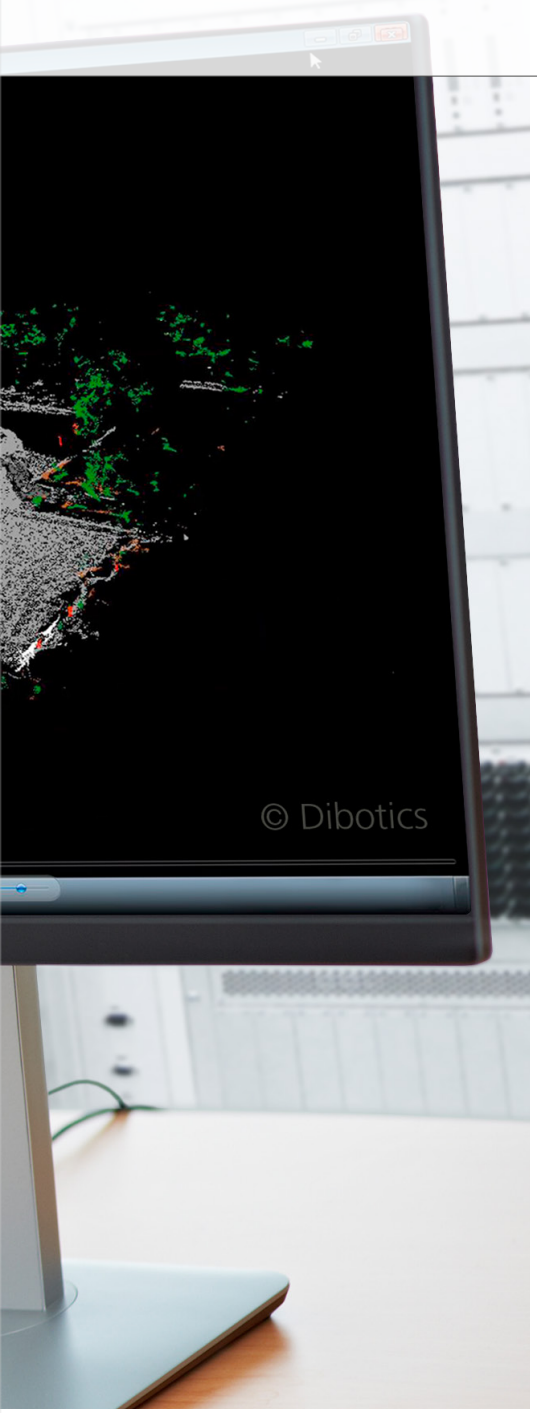


Where Am I?

Environment recognition using SLAM algorithms

To make the vision of autonomous driving a reality, it is important to always know the vehicle position within its environment at any given time. This is also required if no detailed map or satellite navigation is available. In these cases, SLAM algorithms offer a smart solution.

Knowing the exact position of a vehicle in its environment is an important prerequisite for autonomous driving. The most common solution is using precise maps and satellite or inertial navigation systems (inertial measurement units, IMUs) as well as navigation with dead reckoning (location recognition using the direction of movement and velocity). However, for the foreseeable future only a patchy network of detailed maps will be available and the precision of satellite-assisted positioning is also often not sufficient. Simultaneous localization and mapping (SLAM) algorithms offer a solution to this problem. These algorithms use data from environment sensors, such as lidar, to generate a 3-D map of the entire vehicle environment



and locate the vehicle in this environment. The more data is available on the current environment, for example, from rough map data that can be compared to and aligned with the captured 3-D data, the more precisely the algorithm will work.

Using SLAM Algorithms in RTMaps

Augmented LiDAR 3D SLAM algorithms from Dibotics are available in a preconfigured component library for the multisensor development environment RTMaps (Real-Time Multisensor applications, refer to the info box "Profile: RTMaps from Intempora"). RTMaps is integrated in the dSPACE tool chain. SLAM algorithms are perception and localization algorithms based entirely on sensor data. They require neither sensor fusion nor odometry (position estimation based on the propulsion system, for example, wheel rotations). Based on the data of only one lidar sensor, the SLAM algorithms generate a 3-D model in real time by combining a large number of data sets. The 3-D model can then be viewed from different perspectives and also contains a classification of the detected objects (figure 1). At the same time, the vehicle is located in this environment. Afterwards, the environment model forms the basis for subsequent steps in the development of applications for automated driving, for example, trajectory planning and motion control. A simple example for using the SLAM library in RTMaps is replaying sensor data that was captured during test drives. On the simple and intuitive principle of block-based development, the required components are first added to a diagram from the component library in RTMaps via drag & drop. Afterwards, the components are parameterized on the basis of dialogs and in a last step connected as desired via a data link. The application can then be executed and tested simply at the click of a button. For this, the raw data saved to the storage medium is replayed via a replay block and transferred to the SLAM and segmentation algorithms. The algorithm outputs are graphically displayed by means of 3-D visualization blocks. The perspective of the 3-D model can be changed during the operation using the mouse. Thanks to the multithreading capabilities and the efficient memory management of RTMaps, algorithm execution is highly performant despite the algorithms' complexity.

Developing Functions for Autonomous Driving in the Vehicle

For fast prototyping and testing the algorithms in the vehicle, the application must be connected to real environment sensors. This is done by simply replacing the replay block in the RTMaps diagram by a component for capturing sensor data. The development environment provides various interfaces to connect cameras, radar, lidar, and vehicle buses. It also issues time stamps to provide time-correlated capturing, processing, and replay of sensor data. For the development of functions for autonomous driving, information from the generated 3-D environment model is often processed by using specific algorithms, for example, for situation analysis or trajectory planning. These algorithms can be integrated in RTMaps as self-defined components in C++, Python, or even Simulink® code. When integrating self-defined algorithms, the developer can use SDK Wizards in Windows® and Linux as well as numerous code examples for support. The required data exchange with the control algorithms that are executed on the in-vehicle development plat-

>>

RTMaps supports developers with all important steps for creating algorithms for autonomous driving and with the development on embedded platforms.

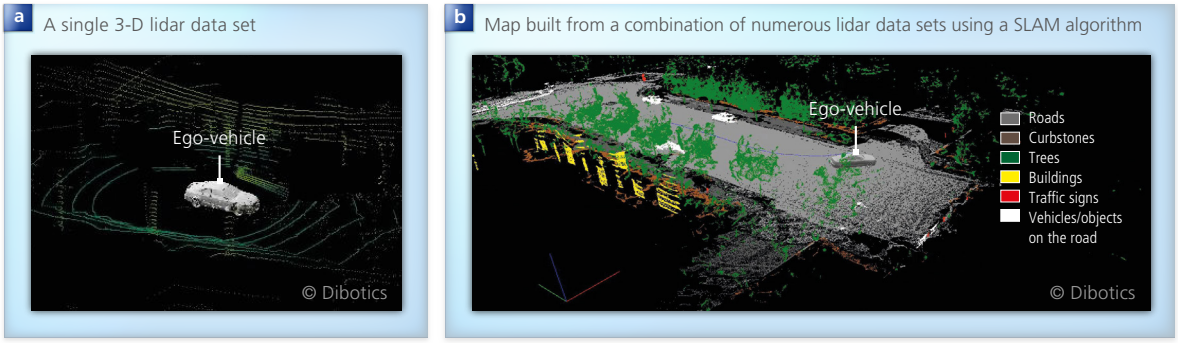


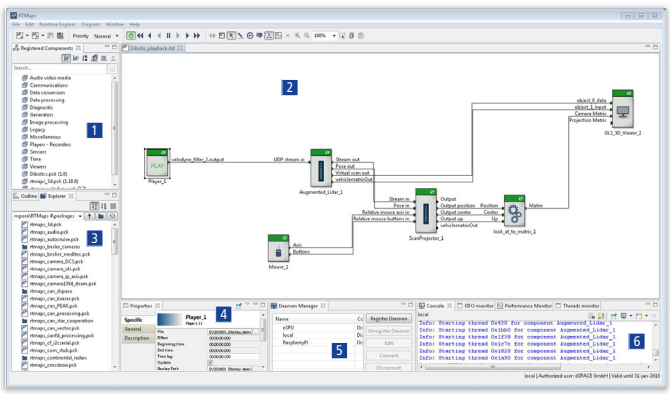
Figure 1: Comparing an individual, unprocessed lidar point cloud (a) and the 3-D map built from numerous lidar data sets (b) demonstrates the effectiveness of SLAM algorithms.

form dSPACE MicroAutoBox can directly be done, for example, for motion control. A special component library helps with the seamless integration of RTMaps in the dSPACE tool chain and ensures a time-synchronous data transfer using dSPACE real-time platforms.

Working Continuously – from the PC to the Embedded Platform

The available resources of a development platform, such as computing power or hardware acceleration, can influence the run-time behavior of complex algorithms. Therefore, developers often have the desire to develop on the target platform as early as possible. Regardless of which platform they choose: Thanks to the convenient tool support of RTMaps, they can always work in their familiar work environment on the PC. The Remote Studio Connector (figure 3) enables them to process and execute RTMaps diagrams via a TCP/IP connection directly from the desired platform without having to attach a keyboard, mouse, or monitor. The environment also automatically makes the working environment more convenient because the developers can always work in the component libraries that are suitable for the selected system and they can load and save the applications directly from the data system of the selected platform. Moreover, the parameterization options also enable loading and starting the desired application in the run-time environment immediately when the target platform is switched on. By adding support for embedded platforms to RTMaps Studio, the prototyping of functions for autonomous driving reaches an unprecedented quality, because the developers can simply and conveniently work on the embedded platform from the familiar environment of their PC. ■

Figure 2: The user interface of RTMaps. The modular development environment allows for easily processing a large variety of multisensor applications. To do this, comprehensive component libraries are available



1 **Component library –** Displays the registered/available component models in the RTMaps environment. These component models are categorized by function (sensors, viewers, image processing, etc.).

2 **RTMaps diagram –** Used to create application diagrams. The elements used in this example are a block for replaying lidar data, Dibotic components for SLAM and segmentation, mouse control for setting the perspective, and the 3-D visualization.

3 **Explorer and Outline –** Explorer: Offers a view of the local directory on the computer. It generally shows a directory with packages that can be registered in RTMaps. Outline: This view shows the entire diagram space for an overview and quick navigation.

4 **Properties –** Lets the user set the different parameters for the current diagram.

5 **Daemon Manager –** Lets the user connect to embedded platforms and open, process, and execute remote diagrams.

6 **Console –** Provides information on various subjects, such as commands, component messages, warnings, and errors.

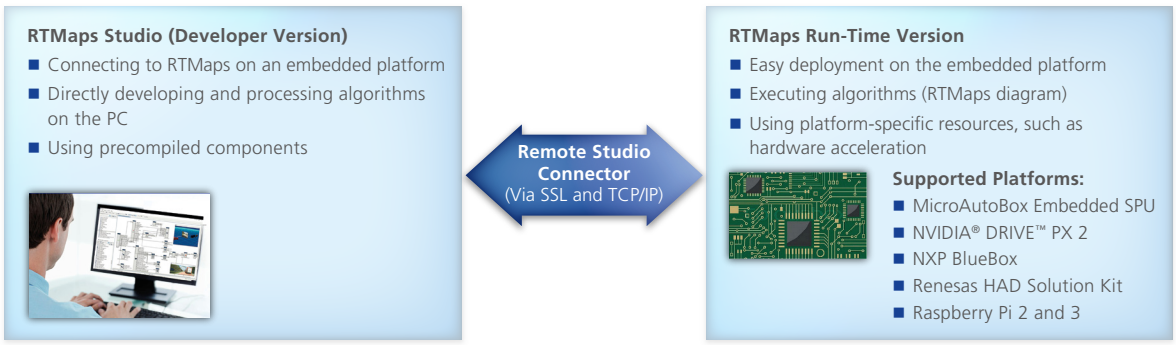
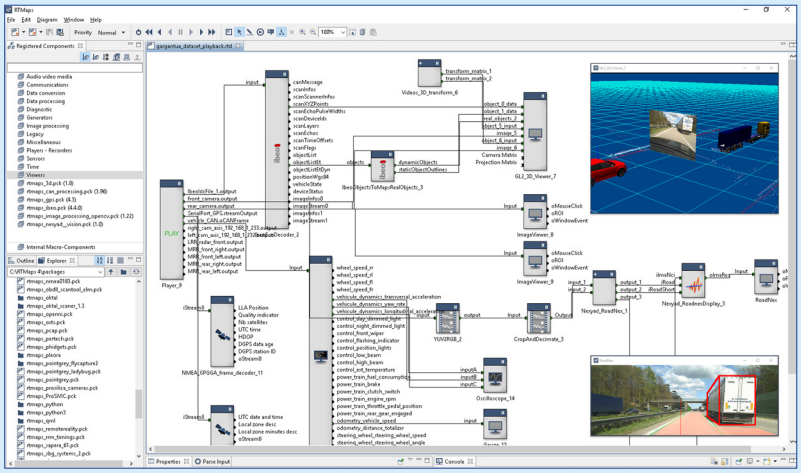


Figure 3: RTMaps with Remote Studio Connector gives developers access to the world of embedded prototyping on various embedded platforms without having to leave their usual work environment on the PC.

Profile: RTMaps from Intempora



Powerful Development Environment for Multisensor Applications

RTMaps from Intempora, which dSPACE has been distributing since 2016, is a modular software development and run-time environment for complex multisensor applications that play an important role in certain areas, such as autonomous driving, robotics, and the aerospace industry. Comprehensive component libraries let the developers accurately capture, synchronize, and process data from various sensors (camera, radar, and lidar). User-specific components can be developed in C++, Python, or as Simulink code and easily integrated. Even complex algorithms, such as OpenCV, Augmented LiDAR 3D SLAM from Dibotics and the support of NVIDIA DriveWorks is available. RTMaps offers unique multithreading capabilities and efficient memory management on PC- and ARM-based platforms.

Seamless Integration in the dSPACE Tool Chain

To integrate RTMaps in the dSPACE tool chain, dSPACE offers an interface blockset that is specifically designed for real-time systems and the PC-based simulation platform VEOS. ControlDesk can also be easily connected with RTMaps to parameterize the developed applications, for example.

NEW: Prototyping Directly on Embedded Platforms

RTMaps with Remote Studio Connector now allows for even faster prototyping of applications on embedded platforms for highly automated driving. Perception and fusion algorithms are therefore developed directly and conveniently on the target platform from the familiar RTMaps development environment on the remote PC, without having to connect mouse, keyboard, and monitor to the embedded platform.