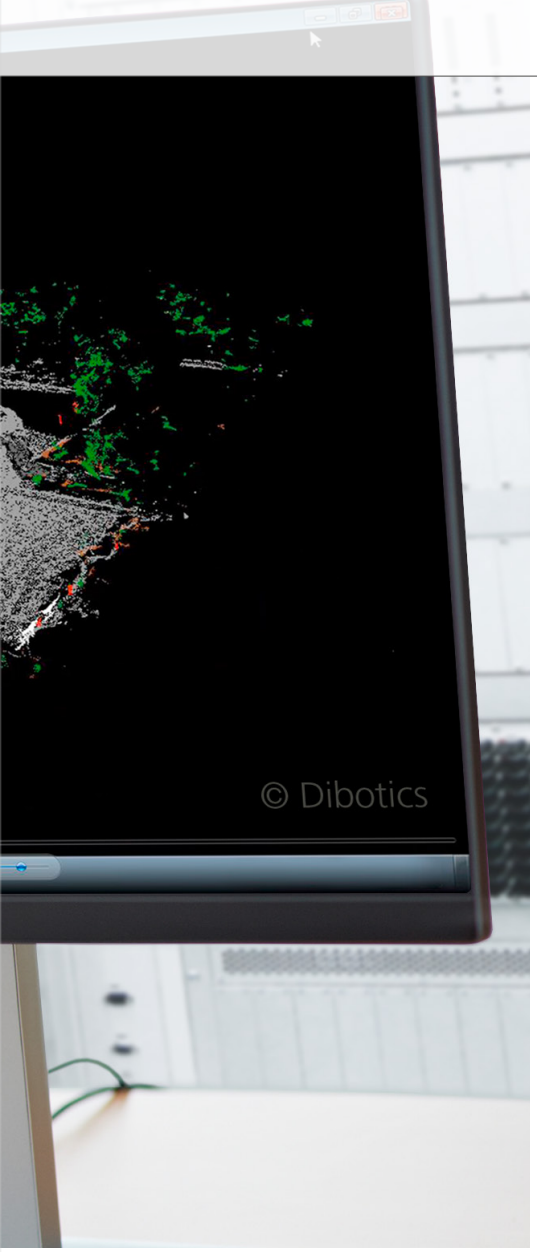

# Wo bin ich?

## Umfelderkennung mit Hilfe von SLAM-Algorithmen

Um die Vision vom autonomen Fahren Realität werden zu lassen, muss unter anderem auch die Fahrzeugposition innerhalb der Umgebung jederzeit bekannt sein. Das gilt auch dann, wenn weder detaillierte Karten noch exakte Satellitennavigation verfügbar sind. In solchen Fällen bieten SLAM-Algorithmen eine elegante Lösung.

Die Kenntnis der exakten Position eines Fahrzeugs in seiner Umgebung ist eine wichtige Voraussetzung für das autonome Fahren. Die naheliegende Lösung besteht in der Nutzung genauer Karten und Satelliten- bzw. Trägheitsnavigationssystemen (Inertial Measurement Units, IMUs) sowie Koppelnavigation (Ortsbestimmung mit Hilfe von Bewegungsrichtung und Geschwindigkeit – „dead reckoning"). Allerdings wird auf absehbare Zeit nur ein lückenhaftes Netz aus detaillierten Karten verfügbar sein, und auch die Genauigkeit von satellitengestützter Positionierung reicht oft nicht aus. Eine Lösung für dieses Problem ist die Verwendung sogenannter SLAM (Simultaneous Localisation And Mapping)-Algorithmen. Hierbei wird in Echtzeit mit Hilfe von Umfeldsensoren, zum Beispiel Lidar, eine 3D-Karte der gesamten Fahrzeugumgebung generiert und das Fahrzeug darin verortet. Das gelingt umso genauer, je mehr Daten zu der aktuellen Umgebung bereits vorliegen, zum Beispiel in Form grober Karten, die man dann mit den erfassten 3D-Daten vergleichen und in Übereinstimmung bringen kann.




## Einsatz von SLAM-Algorithmen in RTMaps

Für die Multisensor-Entwicklungsumgebung RTMaps (Real-time Multisensor Applications, siehe Infobox „Kurzportrait: RTMaps von Intempora“), die dSPACE in seine Werkzeugkette integriert hat, sind die Augmented-Lidar-3D-SLAM-Algorithmen der Firma Dibotics innerhalb einer vorkonfigurierten Komponentenbibliothek verfügbar. Hierbei handelt es sich um Perzeptions- und Lokalisierungsalgorithmen, die ausschließlich auf Sensordaten basieren. SLAM-Algorithmen benötigen weder Sensorfusion noch Odometrie (Positionsschätzung auf Basis des Vortriebsystems, zum Beispiel der Radumdrehungen). Allein mit Hilfe der Daten eines einzigen Lidarsensors wird, Schritt für Schritt und durch Kombination vieler Datensätze, in Echtzeit ein 3D-Modell der Umgebung generiert. Dieses lässt sich dann auch aus verschiedenen Blickwinkeln betrachten und enthält außerdem eine Klassifizierung der erkannten Objekte (Abbildung 1). Gleichzeitig wird das Fahrzeug in dieser Umgebung verortet. Das Umgebungsmodell bildet anschließend die Basis für weitere Schritte bei der Entwicklung von Anwendungen für das automatisierte Fahren, zum Beispiel Trajektorienplanung und Bewegungskontrolle. Ein einfaches Beispiel für die Anwendung der SLAM-Bibliothek in RTMaps ist die Wiedergabe von Sensordaten, die zum Beispiel bei Testfahrten aufgenommen wurden (Abbildung 2). Nach dem einfachen und intuitiven Prinzip blockbasierter Entwicklung werden in RTMaps in wenigen Schritten die benötigten Komponenten zunächst per Drag&Drop aus Komponentenbibliotheken in ein Diagrammfenster eingefügt. Anschließend werden sie dort dialogbasiert parametriert und schließlich auf die gewünschte Weise mittels Datenverbindungen miteinander verknüpft. Die Anwendung lässt sich dann direkt per Knopfdruck ausführen und testen. Dabei werden die auf dem Datenträger gespeicherten Lidar-Rohdaten mit einem Wiedergabeblock abgespielt und den SLAM- und Segmentierungsalgorithmen zugeführt. Die Ausgaben der Algorithmen werden mit Hilfe von 3D-Visualisierungsblöcken grafisch angezeigt. Per Maus lässt sich der Blickwinkel auf das 3D-Modell im laufenden Betrieb ändern. Dank der Multithreading-Fähigkeiten und der effizienten Speicherverwaltung von RTMaps erfolgt die Ausführung der Algorithmen trotz hoher Komplexität sehr performant.

## Entwicklung autonomer Fahrfunktionen im Fahrzeug

Für das schnelle Prototyping und den Test der Algorithmen im Fahrzeug muss die Anwendung an reale Umfeldsensoren angebunden werden. Das geschieht, indem man im RTMaps-Diagramm einfach den Wiedergabeblock durch eine Komponente zur Sensordatenerfassung ersetzt. Die Entwicklungsumgebung bietet hier diverse Schnittstellen zur Anbindung von Kameras, Radar, Lidar oder auch Fahrzeugbussen. Durch Vergabe von Zeitstempeln unterstützt sie auch die zeitkorrelierte Erfassung, Verarbeitung und Wiedergabe von Sensordaten. Bei der Entwicklung von Funktionen für autonomes Fahren werden die Informationen aus dem generierten 3D-Umfeldmodell oft mit Hilfe von speziellen Algorithmen, zum Beispiel zur Situationsanalyse oder Trajektorienplanung, weiterverarbeitet. Diese können in RTMaps in Form von eigenen Komponenten in C++, Python oder auch als Simulink®-Code eingebunden werden. Bei der Integration eigener Algorithmen wird der Entwickler durch SDK Wizards unter Windows® und Linux sowie zahlreiche Code-Beispiele unterstützt. Auch der notwendige Datenaustausch mit Regelungsalgorithmen, die zum Beispiel zur Bewegungskontrolle auf der >>

RTMaps begleitet den Benutzer bei allen wichtigen Schritten auf dem Weg zu Algorithmen für das autonome Fahren und unterstützt die direkte Entwicklung auf Embedded-Plattformen.

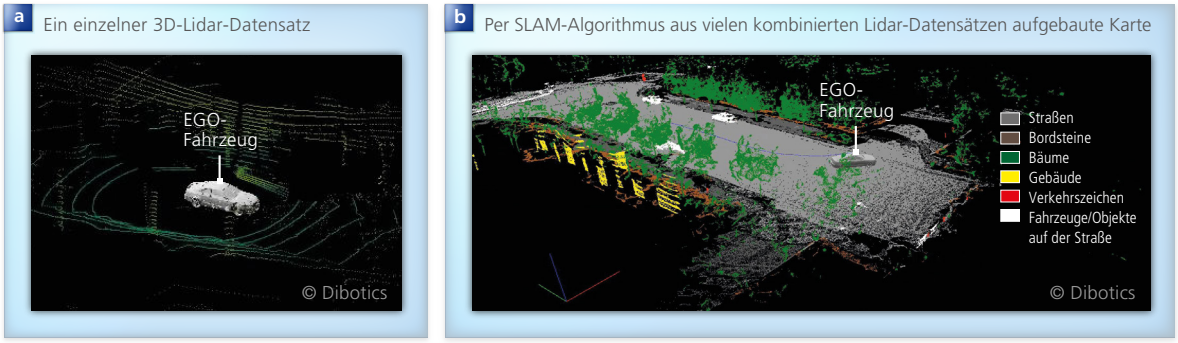


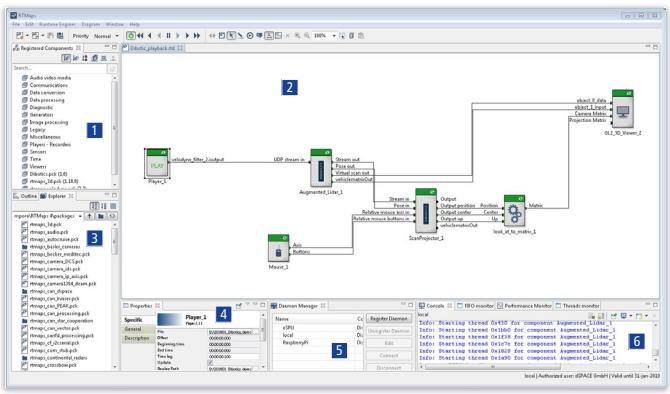

*Abbildung 1: Der Vergleich zwischen einer einzelnen unbearbeiteten Lidar-Punktewolke (a) und der aus vielen Lidar-Datensätzen aufgebauten 3D-Karte (b) verdeutlicht die Leistungsfähigkeit von SLAM-Algorithmen.*

In-Vehicle-Entwicklungsplattform dSPACE MicroAutoBox ausgeführt werden, kann ohne Umwege erfolgen. Eine spezielle Komponentenbibliothek sorgt dabei für nahtlose Integration von RTMaps in die dSPACE Werkzeugkette und gewährleistet eine zeitsynchrone Datenübertragung mit den dSPACE Echtzeitplattformen.

## Durchgängiges Arbeiten – vom PC bis zur Embedded-Plattform

Die verfügbaren Ressourcen einer Entwicklungsplattform wie Rechenleistung oder Hardware-Beschleunigung können das Laufzeitverhalten komplexer Algorithmen beeinflussen. Daher besteht beim Anwender oft der Wunsch, möglichst früh auf der Zielplattform zu entwickeln. Egal für welche Plattform er sich entscheidet: Dank der komfortablen Tool-Unterstützung von RTMaps kann er stets in seiner vertrauten Entwicklungsumgebung vom PC aus arbeiten. Der Remote Studio Connector (Abbildung 3) erlaubt es ihm, über eine TCP/IP-Verbindung RTMaps-Diagramme direkt auf der gewünschten Plattform zu bearbeiten und auszuführen, ohne dabei Tastatur, Maus oder Monitor anschließen zu müssen. Für weiteren Arbeitskomfort sorgt die Entwicklungsumgebung automatisch dadurch, dass stets mit den zum gewählten System passenden Komponentenbibliotheken gearbeitet wird und dass das Laden und Speichern der Anwendung direkt in dem Dateisystem der gewählten Plattform erfolgen kann. Darüber hinaus erlauben diverse Parametrierungsoptionen unter anderem auch, dass bereits beim Einschalten die gewünschte Anwendung auf der Zielplattform in die Laufzeitumgebung geladen und gestartet wird. Durch die Erweiterung von RTMaps Studio um die Unterstützung von Embedded-Plattformen erreicht das Prototyping von Funktionen für das autonome Fahren eine neue Qualität, weil der Entwickler wie gewohnt einfach und bequem vom PC aus direkt auf der Embedded-Plattform seiner Wahl arbeiten kann. ■

*Abbildung 2: Die Benutzeroberfläche von RTMaps. Mit der modularen Entwicklungsumgebung lassen sich vielfältigste Multisensoranwendungen komfortabel bearbeiten. Dazu stehen umfangreiche Komponentenbibliotheken zur Verfügung.*




1 **Komponentenbibliothek –** Zeigt die registrierten/verfügbaren Komponentenmodelle in der RTMaps-Umgebung an. Diese Komponentenmodelle sind nach ihren Funktionen kategorisiert (Sensoren, Viewer, Bildverarbeitung etc.).

2 **RTMaps-Diagramm –** Dient zum Erstellen der Anwendungsdiagramme. Die in dem hier gezeigten Beispiel verwendeten Elemente sind ein Block zur Wiedergabe von Lidar-Daten, Dibotics-Komponenten für SLAM und Segmentierung, Mausteuerung zur Einstellung des Betrachtungswinkels und der 3D-Visualisierung.

3 **Explorer und Outline –** Explorer: Er bietet eine Ansicht eines lokalen Ordners auf dem Computer. Er zeigt in der Regel auf einen Ordner mit Paketen, die in RTMaps registriert werden können. Outline: In dieser Ansicht wird die gesamte Diagrammfläche für eine Übersicht und eine schnelle Navigation dargestellt.

4 **Properties –** Ermöglicht die Einstellung verschiedener Parameter für das aktuelle Diagramm.

5 **Daemon Manager –** Ermöglicht die Verbindung von Embedded-Plattformen sowie das Öffnen, Bearbeiten und Ausführen von Remote-Diagrammen.

6 **Console –** Liefert verschiedene Informationen wie Befehle, Komponentenmeldungen, Warnungen oder Fehler.

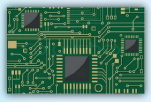
**RTMaps Studio (Developer Version)**

- Verbindung zu RTMaps auf einer Embedded-Plattform
- Direkte Entwicklung und Bearbeitung von Algorithmen vom PC aus
- Verwendung vorkompilierter Komponenten

**Remote Studio Connector** (via SSL und TCP/IP)

**RTMaps Run-Time Version**

- Einfache Bereitstellung auf der Embedded-Plattform
- Ausführung von Algorithmen (RTMaps-Diagramm)
- Nutzung plattformspezifischer Ressourcen wie der Hardware-Beschleunigung

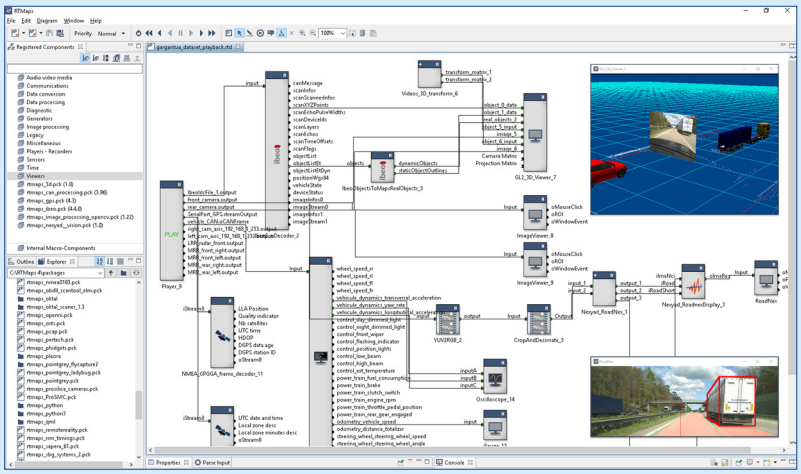


**Unterstützte Plattformen:**

- MicroAutoBox Embedded SPU
- NVIDIA® DRIVE™ PX 2
- NXP BlueBox
- Renesas HAD Solution Kit
- Raspberry Pi 2 und 3

*Abbildung 3: RTMaps mit Remote Studio Connector eröffnet dem Entwickler die Welt zum Embedded Prototyping auf diversen Embedded-Plattformen, ohne die gewohnte Arbeitsumgebung auf seinem PC verlassen zu müssen.*

# Kurzportrait: RTMaps von Intempora



## Leistungsfähige Entwicklungsumgebung für Multisensoranwendungen

RTMaps von Intempora, das dSPACE seit 2016 vertreibt, ist eine modulare Software-Entwicklungs- und Laufzeitumgebung für anspruchsvolle Multisensoranwendungen, die beispielsweise in den Bereichen autonomes Fahren, Robotik oder Luft- und Raumfahrt eine entscheidende Rolle spielen. Umfangreiche Komponentenbibliotheken erlauben dem Anwender, Daten von unterschiedlichen Sensoren (Kamera, Radar, Lidar etc.) präzise zu erfassen, zu synchronisieren und zu verarbeiten. Eigene Komponenten lassen sich in C++, Python oder als Simulink-Code entwickeln und einfach integrieren. Auch komplexe Algorithmen wie OpenCV, Augmented Lidar 3D SLAM von Dibotics und die Unterstützung von NVIDIA DriveWorks sind verfügbar. RTMaps zeichnet sich durch einzigartige Multithreading-Fähigkeiten und effiziente Speicherverwaltung auf PC- und ARM-basierten Plattformen aus.

## Nahtlose Integration in die dSPACE Werkzeugkette

Für die Integration von RTMaps in die dSPACE Werkzeugkette bietet dSPACE ein Schnittstellen-Blockset, das speziell für die Echtzeitsysteme und die PC-basierte Simulationsplattform VEOS ausgelegt ist. Auch ControlDesk lässt sich mühelos mit RTMaps verbinden, zum Beispiel um die entwickelten Anwendungen zu parametrieren.

## NEU: Prototyping direkt auf Embedded-Plattformen

RTMaps mit Remote Studio Connector erlaubt jetzt noch schnelleres Prototyping von Anwendungen auf Embedded-Plattformen für hochautomatisiertes Fahren.
Die Entwicklung von Perzeptions- und Fusionsalgorithmen erfolgt dabei direkt auf der Zielplattform und bequem von der gewohnten RTMaps Entwicklungsumgebung am Remote-PC, ohne Maus, Tastatur und Bildschirm an die Embedded-Plattform anschließen zu müssen.