# Speed is Key to Safety

Developing algorithms for autonomous driving that respond promptly and precisely to the perceived environment

Indiana University – Purdue University Indianapolis (IUPUI) is researching ways to improve road transport safety for autonomous applications by analyzing the benefits of high-speed sensor data processing. RTMaps Embedded and NXP BlueBox are serving as the core real-time execution platform for embedded computing capabilities.

The number of sensors and electronic control units (ECUs) used in autonomous vehicles is growing exponentially, making it critical for automakers to use advanced and efficient solutions to handle numerous tasks with real-time execution performance. Data logging, time stamp synchronization, and processing from various sensors, such as cameras, laser scanners, radars, and global navigation satellite system (GNSS) receivers, have to be completed within milliseconds to ensure the precise position of a vehicle and its safe operation. Students at the Purdue School of Engineering and Technology at Indiana University – Purdue University Indianapolis (IUPUI) recently completed several studies to test the use of a high-speed computational platform for processing sensor data for four different autonomous applications.

## Building the Test Platform

To begin their research, the students set up a test platform that included RTMaps Embedded (version 4.5.0) by Intempora. RTMaps Embedded, distributed by dSPACE, is a software solution for real-time multisensor applications that handles time coherency among numerous software tasks and provides a high bandwidth of raw data streams with powerful real-time execution performance. The RTMaps solution encompasses several independent modules: RTMaps Runtime Engine, RTMaps Studio, RTMaps Component Library and RTMaps SDK (Software Development Kit). "RTMaps Embedded is designed to face and win multisensory challenges," said Professor Mohamed El-Sharkawy, Purdue School of Engineering and Technology. "It provides an efficient and easy-to-use framework for fast and robust developments in areas such as advanced driver assistance systems, automotive vehicles, and robotics. It was easy for us to develop, test, validate, benchmark, and execute applications." The test platform also featured NXP BlueBox, an embedded computing system that allows the vehicle to create a real-time, high-definition 3-D image of its surroundings. Specifically, the students used BlueBox Version 2.0, which incorporates the S32V234 automotive vision and sensor fusion processor, the LS2084A integrated communication processor, and the S32VR27 radar micro controller. "NXP BlueBox provides the performance required to analyze driving environments and assess risk factors as well as automotive reliability to develop self-driving cars, "Professor El-Sharkawy added "BlueBox provides the required performance, func- >>

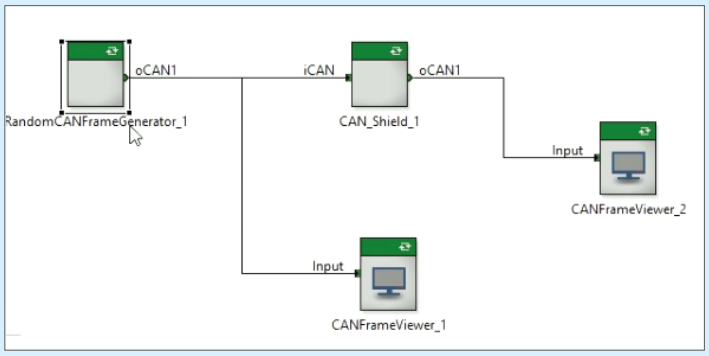



*Figure 1: RTMaps workspace for a user-defined component package.*

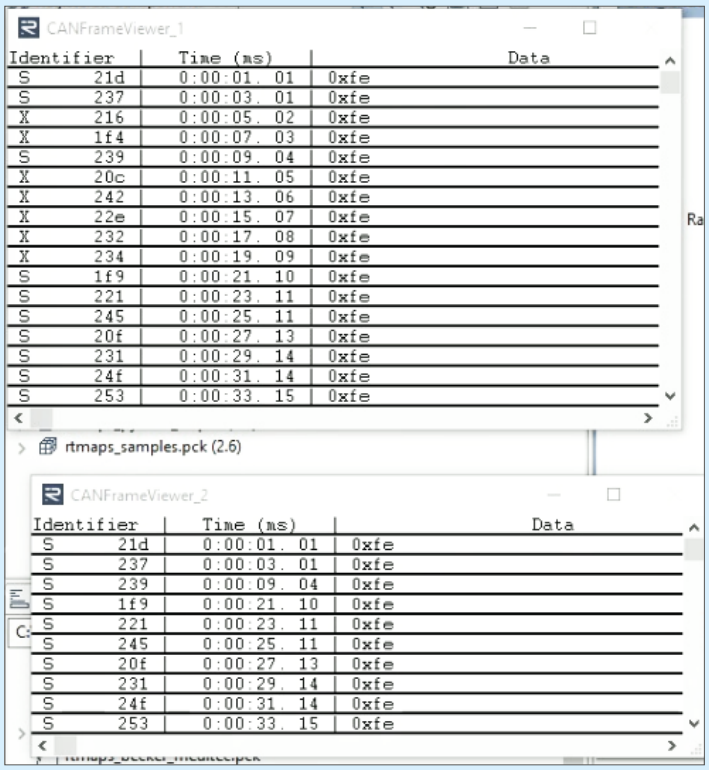



*Figure 2: Display of output using CAN Frame viewer in RTMaps.*

tional safety, and automotive reliability, and it can be integrated with RTMaps."

*Note: RTMaps Embedded is also compatible with the dSPACE MicroAutoBox Embedded SPU – a prototyping platform for functions for automated driving.*

## Testing Five Applications

With their test platform set up, the students set out to study and prototype five applications for autonomous driving:

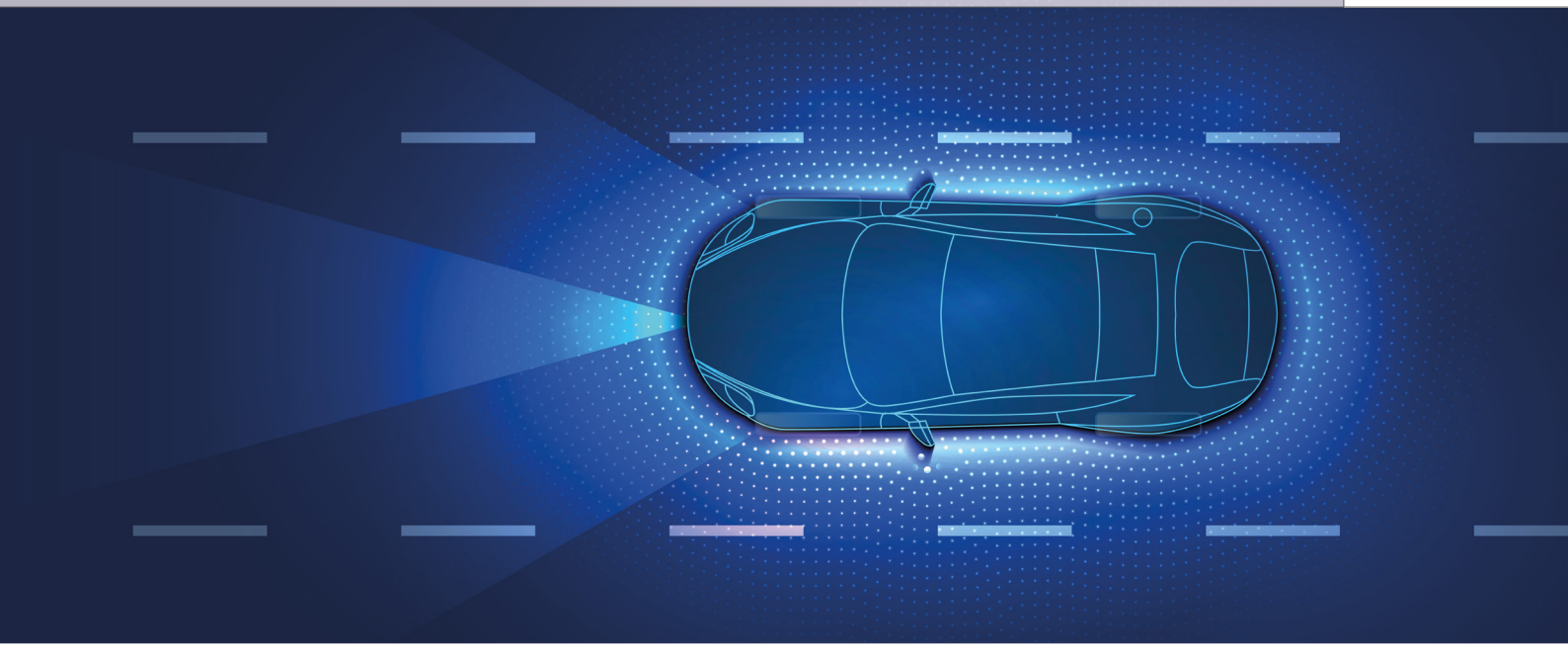
1. Creating a user-defined component package
2. Developing a neural network Python component
3. Pedestrian detection
4. Recording and replaying of real-time scenarios
5. Forward collision warning using SVM classifier

## Application 1: Creating a User-Defined Component Package

The students used the RTMaps Studio graphical development environment and the RTMaps Component Library, and integrated C/C++ and Python code to create a component package. They used RTMaps SDK to import the component package to the RTMaps project workspace. In RTMaps Studio, applications are represented by connected components provided by the RTMaps Component Library. "The Studio tool provides an easy way to set up complex modular applications," said IUPUI engineering student Sreeram Venkitachalam. "Components can simply be dragged to the Studio workspace from the library. The components are used for communication, interface sensors, building an algorithm, and connecting the actuators." Figure 1 illustrates a user-defined component package that was imported into RTMaps Studio. The RandomCANFrameGenerator package was used to generate CAN frames, both standard and extended. The incoming CAN data was filtered so that only the standard frames were displayed in a data viewer, which is a built-in component of RTMaps. Two actions are incorporated for the generator component to control the generation speed of CAN frames. The 'speed up' action doubles the current speed of CAN generation, and the 'speed down' action reduces the speed of the CAN generation by 50%.

## Application 2: Developing a Neural Network Python Component

In their next application project, the students developed a neural network Python component and classified the input image. For this, they used the Python component of the RTMaps Embedded package, which features an editor with syntax coloration to help users develop Python scripts. The



editor can be opened by right-clicking the component module. The Python block of RTMaps generates a class called 'RTMaps_Python', which can be used to reactively or periodically call an input. The core function in the RTMaps_Python class acts as an infinite loop running the main program. Using the Python component of RTMaps Embedded, the students were able to complete two example projects: vehicle detection and traffic sign classification.

### Vehicle Detection

To detect vehicles, an image was implemented using a five-layer convolutional neural network (CNN) with pooling in each layer. The network used two fully connected layers as well as a drop-out layer to prevent overfitting of the network. The model was trained using a regression model and optimized with an Adam optimizer. The skeleton structure of the neural network was added to the RTMaps Python block, and the trained weights for the block were fed to the network as saved data. The network in the Python block checks whether an image contains a vehicle. The RTMaps application is sent to the BlueBox to run on the embedded platform. The RTMaps Python libraries in the BlueBox are modified to include a TensorFlow deep learning library (tflearn) for building the CNN structure.

> "The Studio tool of RTMaps provides us with an easy way to set up complex modular applications."
>
> *Sreeram Venkitachalam, IUPUI engineering student*

### Traffic Sign Classification

In their second example project, the students were able to classify German traffic signs using a TensorFlow model. They used the model to identify the traffic signs, assign each of them to one of 43 classes, and display the meaning of the detected traffic sign in the image. The classification model used approx. 39,000 images to train the model. The TensorFlow model was recreated for testing, and the model weights, saved using the training model, were added to predict the class of the input image. The entire test structure was created in the RTMaps Python component. The trained model and the captured images were sent to the BlueBox, and according to the sign in the image, the output was predicted as a label, indicating the meaning of the traffic sign. >>

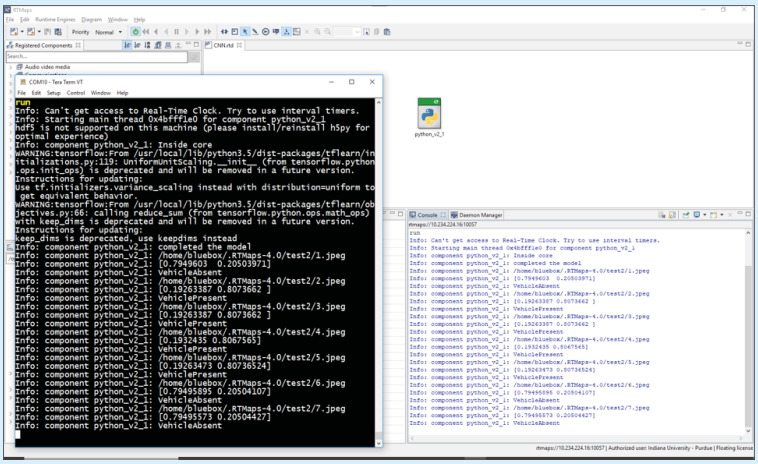



*Figure 3: Output display for vehicle detection based on a convolutional neural network (CNN) using BlueBox.*

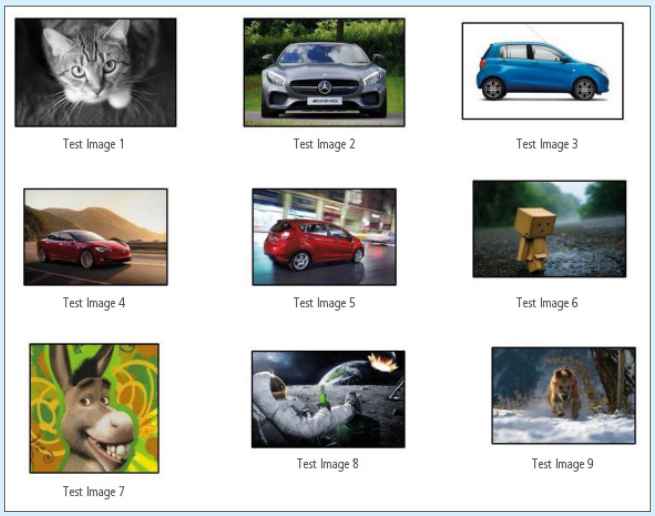



*Figure 4: Input images for testing vehicle detection.*

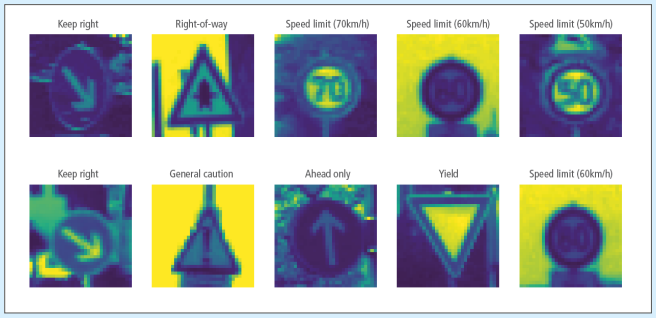



*Figure 5: German traffic signs classified using a TensorFlow model.*

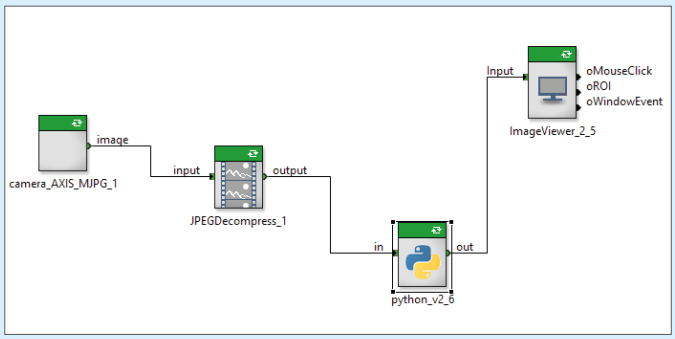



*Figure 6: Diagram of IPCAM-based pedestrian detection.*

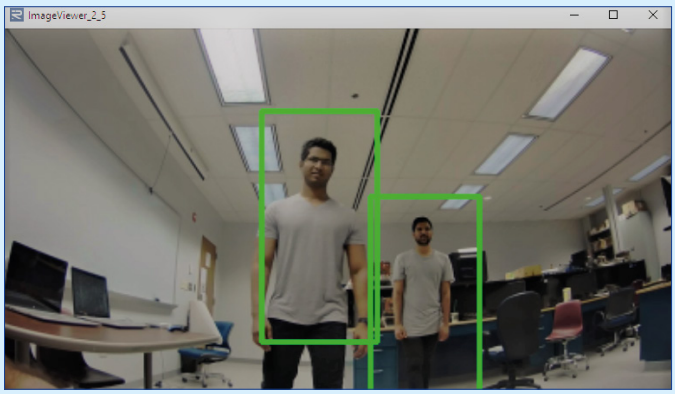



*Figure 7: Bounding boxes are drawn around detected pedestrians.*

> "RTMaps Embedded is designed to face and win multisensor challenges. It provides us with an efficient and easy-to-use framework for fast and robust developments in areas such as advanced driver assistance systems, automotive vehicles, and robotics."
>
> *Professor Mohamed El-Sharkawy, Purdue School of Engineering and Technology*

### Application 3: Detecting Pedestrians

In their next application experiment, the students were able to achieve pedestrian detection by capturing real-time frames with an Axis IP camera (IPCAM). The pedestrian detection model in the Python application of RTMaps detects persons in the input image and draws a bounding box for each one (figure 7). A histogram of oriented gradients (HOG) and a linear supported vector machine (SVM) were used as a pretrained detector. To avoid an overlapping of bounding boxes, non-maxima suppression (NMS) was used. The pedestrian detection model was integrated with the IPCAM and the input images from the IPCAM were fed into the Python application. The detected pedestrians and the related bounding boxes were then displayed in the image viewer of RTMaps.

### Application 4: Recording and Replaying Real-Time Scenarios

In their fourth application project, the students were able to demonstrate the ability of RTMaps to record and play back real-time data. The captured data was saved as a REC file that the students were able to play back. In this example, the IPCAM was integrated into the LS2084A of BlueBox to capture the images and save them to the IPCAM folder as REC files.

### Application 5: Forward Collision Warning Using SVM Classifier

With their final application project, the students set out to implement a forward collision warning system that included a forward-looking automotive radar model and a classifier algorithm. The outputs from the radar model were the velocity/acceleration

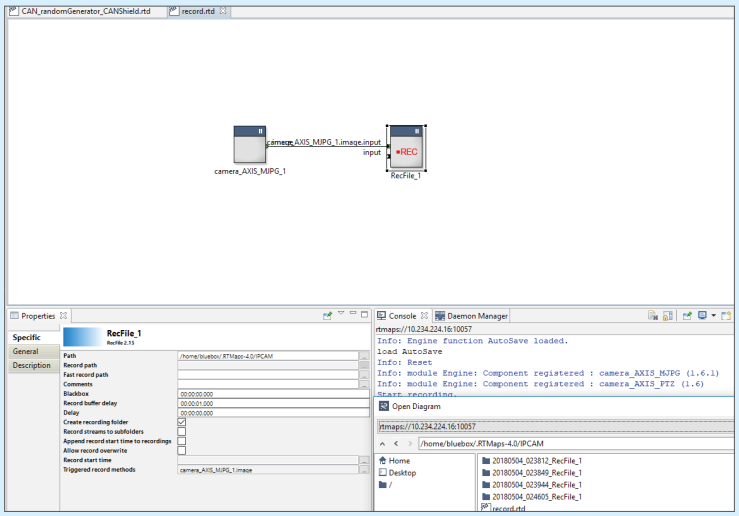



*Figure 8: BlueBox recording example.*

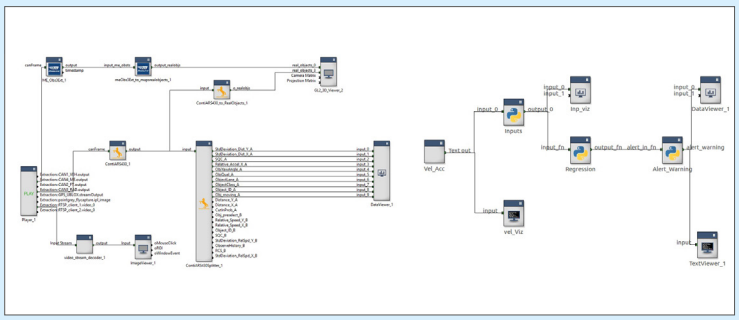



*Figure 9: Diagram of forward collision warning using SVM classifier.*

and separation distance. These outputs were used as inputs to the linear regression and the support vector machine (SVM) classifier to predict the warning range, i.e., the range in which a collision could occur between the ego-vehicle and the leading vehicle.

## Conclusion

The IUPUI engineering students were able to complete various application projects for autonomous driving with the RTMaps Embedded platform and NXP BlueBox. They completed the development and testing of five different application examples to reaffirm the integration feasibility of Intempora RTMaps 4.5.0 with the computational platform BlueBox 2.0. While the data volume and complexity of ADAS algorithms increases, the students benefit from the computing power of RTMaps and BlueBox to ensure that the autonomous vehicle quickly detects and responds to its surrounding traffic environment and driving conditions. ■

*Courtesy of the Indiana University – Purdue University Indianapolis*

## Contributors from the Purdue School of Engineering and Technology:

*Mohamed El-Sharkawy*
*Professor*
*Fulbright Scholar*
*Director of the Internet of Things (IoT) Collaboratory*

*Akash Gaikwad*
*Graduate Student*
*IoT Collaboratory*
*Department of Electrical and Computer Engineering*

*Surya Kollazhi Manghat*
*Graduate Student*
*IoT Collaboratory*
*Department of Electrical and Computer Engineering*

*Sreeram Venkitachalam*
*Graduate Student*
*IoT Collaboratory*
*Department of Electrical and Computer Engineering, Currently Test and Verification Engineer at Aptiv*

*Niranjan Ravi*
*Graduate Student*
*IoT Collaboratory*
*Department of Electrical and Computer Engineering*

*Sree Bala Shruthi Bhamidi*
*Graduate Student*
*IoT Collaboratory*
*Department of Electrical and Computer Engineering*

*Dewant Katare*
*Graduate Student*
*IoT Collaboratory*
*Department of Electrical and Computer Engineering*