Realtime Applications with RTMaps and Bluebox 2.0

Sreeram Venkitachalam, Surya Kollazhi Manghat, Akash Sunil Gaikwad, Niranjan Ravi, Sree Bala
Shruthi Bhamidi and Mohamed El-Sharkawy
Electrical and Computer Engineering
Purdue School of Engineering and Technology, IUPUI

Abstract **- This paper aims to build applications for improving road transport safety using Intempora RTMaps Embedded and NXP Bluebox. Data logging, timestamp synchronization and processing are done in real time using RTMaps multi sensor application and Bluebox development platform. RTMaps runtime engine is the core of any RTMaps based application. It is light weight, multithreaded and easily embeddable in third party applications. Using four simple examples, we extend the RTMaps to accommodate embedded computing capabilities of Bluebox and show that the RTMaps embedded is a good framework for prototyping the multi sensor application of automotive field.**

Keywords**: Intempora, RTMaps, NXP Bluebox, Embedded Targets, Sensor Fusion, CNN/DNN, S32V234, Functional Safety**

I. INTRODUCTION

RTMaps is an asynchronous high-performance platform designed to face and win multisensory challenges and allow taking advantage of efficient and easy to use framework for fast and robust developments in areas such as ADAS, automotive vehicles, robotics etc. It provides an easiest way to develop test validate benchmark and execute the applications. RTMaps is made of several independent modules to be used in different contexts -RTMaps Runtime Engine, RTMaps Studio, The RTMaps Component Library and RTMaps SDK. [1]

RTMaps studio is the graphical development interface. Applications are represented by diagram in the workspace, made of connected components provided by RTMaps component library. The studio provides an easy way to set up the complex modular applications. Components are used by drag and drop from the library. These are used to communication, interface sensors, build an algorithm and connect the actuators. The Runtime Engine takes care of all base services such as component registration, buffer management, time stamping threading and priorities. The Software Development Kit from RTMaps helps to develop a component at will and expands the component library provided.

With tremendous increase of the number of sensors and ECUs, it become critical to automakers to use advanced and efficient solutions to handle numerous tasks with a real time execution performance. RTMaps is a real time multi-sensor application which can process data from various sensors such as cameras, laser scanners, radars and GNSS receivers. The RTMaps embedded V4.5.0 and RTMaps studio connector on the embedded targets make the software run on highly developed computational target and be able to dramatically reduce the development time. RTMaps embedded is available on various targets such as NXP Bluebox. [1]

We make use of RTMaps embedded available on NXP Bluebox and the components already compiled for the embedded platform. NXP Bluebox provides the performance required to analyze driving environments, assess risk factors and automotive reliability to develop self driving cars. The Bluebox2.0 incorporates the S32V234 automotive vision and sensor fusion processor, LS2084A embedded computer processor and S32VR27 radar micro controller. The applications/algorithms developed are deployed on Bluebox using SSL connection. The NXP Bluebox is a development platform that provides required performance, functional safety and automotive reliability, and can be embedded with RTMaps. In this paper we study and prototype multiple applications with RTMaps embedded with Bluebox. [3]

II. APPLICATIONS

A. User Defined Packages

Developing your own components and complete the components library at will, by integrating the code in C/C++ or python. This package generated can make to do any kind of processing according to the need. This is developed in C++ using RTMaps Software Development Kit(SDK). The created package is imported to the RTMaps workspace and used in the project.

"RandomCANFrameGenerator" and "CAN_shield", shown in Fig. 1, here are two packages built by this method and imported to RTMaps studio. The RandomCANFrameGenerator is then used to generate CAN frames, both standard and extended. This incoming CAN data is then filtered out so that the only the Standard frames are displayed. The display output can then be seen on a data viewer, which is an in-built component of RTMaps. We have incorporated 2 actions for the generator component to control the speed of generation of CAN frames. The "Speed up" action doubles the current speed of CAN generation and "Speed down" action reduces the speed of CAN generation by half.

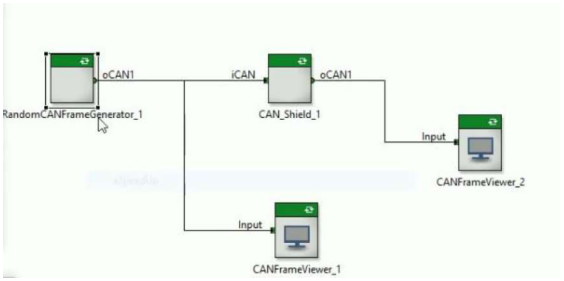

Fig. 1: Workspace with a user defined package

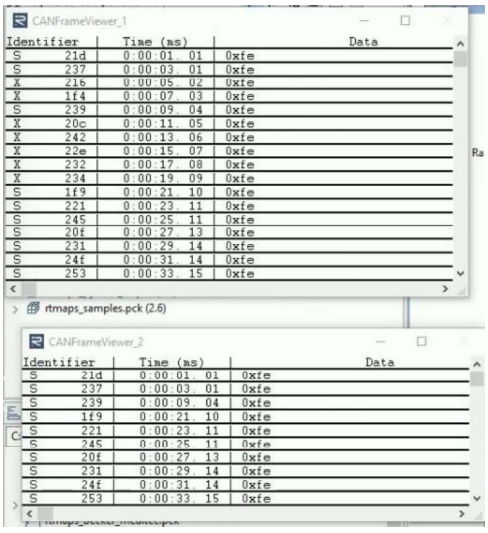

Fig. 2: Display of output using CAN Frame viewer

B. Neural Network with Python component

The Python component of RTMaps has an editor with syntax coloration to help users develop with python. The editor can be opened via right click on the component module. Python component of RTMaps embedded packages is used to develop a neural network and do the classification of the input image. The python block of RTMaps make a class called RTMaps_python. This can be made to call reactively to an input or periodically. The core function inside the RTMaps_python class act as infinite loop and this is where main program is running. The vehicle detection and Traffic sign classification are the two examples implemented in python. [1]

The vehicle presence detection in an image is implemented using a 5-layer Convolutional Neural Network (CNN) with pooling present in each layer. This uses 2 fully connected layers along with a drop out layer which prevents overfitting of the network. The model is trained using Regression model and optimized using Adam optimizer. The skeleton structure of the Neural Network is added in the python block of RTMaps and the trained weights for the same is fed to the network as saved data. This is obtained from previously trained network. This network in the python core block will be responsible to check if a given image has a vehicle in it. The RTMaps application is sent to the Bluebox to run on the embedded platform. The RTMaps python libraries in the Bluebox is modified to include tensor flow deep learning library-tflearn for making the CNN structure.

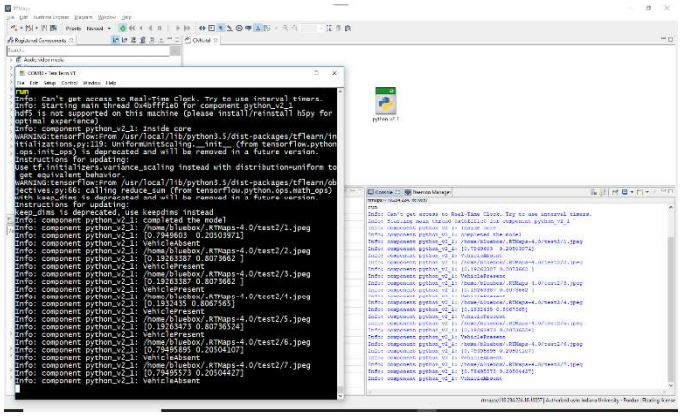

Fig. 3: Output display for the Vehicle Detection based on CNN using the Bluebox

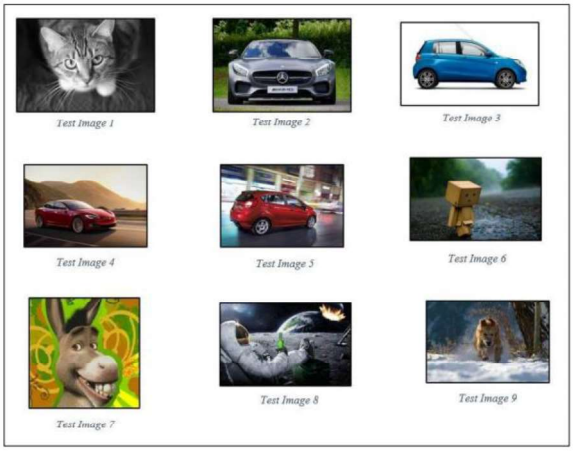

Fig. 4: Input images given for testing vehicle detection

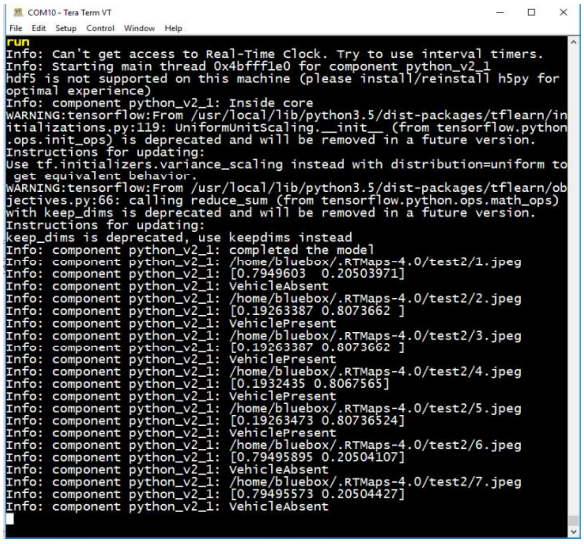

Fig. 5: Output as seen in Console for Vehicle detection.

The second project with RTMaps-Embedded python, is for traffic sign (Germany) classification using tensor flow model. This is made to identify the traffic sign board, classify it in to one of the 43 classes and display the meaning of detected traffic sign board from the image. The classification model used ~39000 images to train the model. The tensor flow model is recreated for testing and the model weights, saved using the training model, are added to predict the class of input image. This entire test structure is done in the RTMaps python component. The trained model and the captured images are sent to the Bluebox and according to the sign board in the image the output is predicted as label, indicating the meaning of the traffic sign.

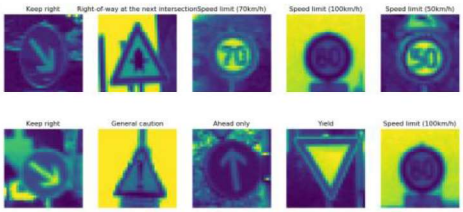

Fig. 6: IPCAM based pedestrian detection

C. Pedestrian Detection using IPCAM

This can capture the real-time frames using Axis IP camera. The pedestrian detection model in the python application of RTMaps will detect the presence of humans in the input image and draw a bounding box for each detected person.

This uses Histogram of Oriented Gradients and Linear supported Vector Machine(SVM) to be a pre-trained detector. The scaling factor of image plays a key role in pedestrian detection. To avoid the overlapping the bounding boxes around the detected pedestrians Non-Maxima Suppression(NMS) is used. This model is integrated with the IPCAM. The input images from the IPCAM are fed in to the python application and the detected pedestrians with bounding boxes are displayed using image viewer of RTMaps.

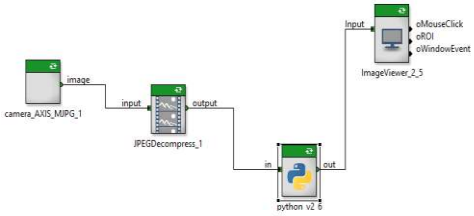

Fig. 7: IPCAM based pedestrian detection

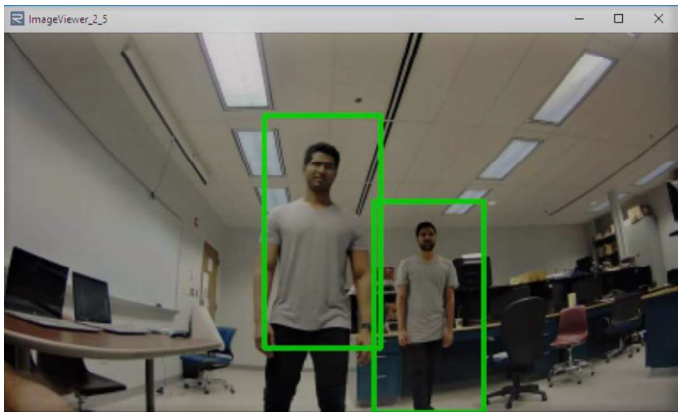

Fig. 8: Output for pedestrian detection

D. Record and Replay of real-time scenarios

RTMaps allow you to record the real time data and play it back when you want to. The captured file will be saved as .rec file. The "PLAY" allow you to replay the file. Here the IPCAM integrated to the LS2084A of Bluebox will capture the images and save it as .rec in the IPCAM folder inside /home/bluebox/RTMAPS4.0.

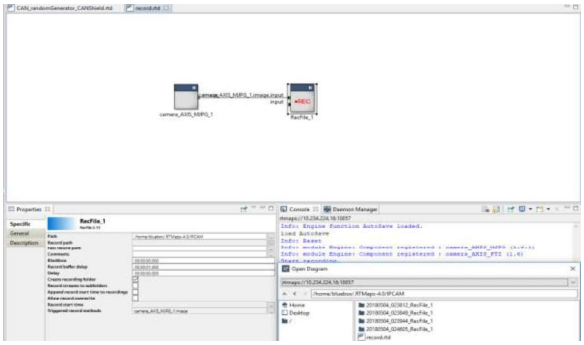

Fig. 9: Bluebox record example

III. CONCLUSION

In this paper, we have presented examples based on RTMaps embedded with Bluebox for autonomous applications along with RTMaps Studio for PC. We clarified the integration feasibility of Intempora RTMaps 4.5.0 with the high computational platform Bluebox2.0. We also added the applications developed to test the feasibility. The applications can be found at the location mentioned in [5] and [6].

IV. REFERENCES

1. Intempora.com. (2018). Intempora - RTMaps - A component-based framework for rapid development of multi-modal applications. [online] Available at: https://intempora.com/ [Accessed 6 Jun. 2018].
2. Intempora.com. (2018). Intempora - RTMaps. [online] Available at: https://intempora.com/products/rtmaps [Accessed 6 Jun. 2018].
3. Nxp.com. (2018). NXP BlueBox Autonomous Driving|NXP. [online] Available at: https://www.nxp.com/products/processors- and-microcontrollers/arm-based-processors-and-mcus/qoriq-layerscape-arm-processors/nxp-bluebox-autonomous-driving-development-platform:BLBX [Accessed 6 Jun. 2018].
4. Rosebrock, A. (2018). Pedestrian Detection OpenCV - PyImageSearch. [online] PyImageSearch. Available at: https://www.pyimagesearch.com/2015/11/09/pedestrian-detection-opencv/ [Accessed 6 Jun. 2018].
5. Sreeram Venkitachalam, Surya kollazhi Manghat, Akash Sunil Gaikwad, Realtime-applications-with-RTMaps-and-Bluebox-2.0, (2018), GitHub repository, https://github.com/sreeramv91/Realtime-applications-with-RTMaps-and-Bluebox-2.0/settings
6. Akash Sunil Gaikwad, Surya kollazhi Manghat, Sreeram Venkitachalam, Realtime-applications-with-RTMaps-and-Bluebox-2.0, (2018), GitHub repository, https://github.com/akashsunilgaikwad/Traffic-Sign-Recognition-using-convolution-neural-network-with-RTMaps-Embedded-and-BlueBox-V2