Autonomous Embedded System Enabled 3-D Object Detector: (with Point Cloud and Camera)

Dewant Katare and Mohamed El-Sharkawy
IoT Collaboratory IUPUI, Department of Electrical and Computer Engineering
Purdue School of Engineering and Technology Indianapolis
dkatare@iupui.edu melshark@iupui.edu

***Abstract*—An Autonomous vehicle or present day smart vehicle is equipped with several ADAS safety features such as Blind Spot Detection, Forward Collision Warning, Lane Departure and Parking Assistance, Surround View System, Vehicular communication System. Recent research utilize deep learning algorithms as a counterfeit for these traditional methods, using optimal sensors. This paper discusses the perception tasks related to autonomous vehicle, specifically the computer-vision approach of 3D object detection and thus proposes a model compatible with embedded system using the RTMaps framework. The proposed model is based on the sensors: camera and Lidar connected to an autonomous embedded system, providing the sensed inputs to the deep learning classifier which on the basis of theses inputs estimates the position and predicts a 3-d bounding box on the physical objects. The Frustum PointNet a contemporary architecture for 3-D object detection is used as base model and is implemented with extended functionality. The architecture is trained and tested on the KITTI dataset and is discussed with the competitive validation precision and accuracy. The Presented model is deployed on the Bluebox 2.0 platform with the RTMaps Embedded framework.**

***Index Terms*—Autonomous vehicles, BLBX2, camera, lidar, KITTI, object detection, RTMaps.**

I. Introduction

The number of sensors and ADAS safety feature in a vehicle has been categorically increased in the past years, the current trend follows incorporation of these sensors with the state-of-the art deep learning architecture based on the sense, think and act model, that can provide the assistance to the driver or replace a driver by providing the highest level of autonomy [5]. The highest level of autonomy in a vehicle can be described as execution of multiple processes, that serves the self driving functionality from a source point to destination point without any input or control to a vehicle from the human. Research shows, this highest level of autonomy is achieved by integrating multiple sensors such as camera, lidar, global navigation satellite system, radar which provides the automotive safety feature or the advanced driver assistance system [5][13][18]. The essential component or sub-system of an autonomous or self-driving vehicle can be specified into four categories: sensing, perception, planning and control. Sensing is acquiring of raw data, or reading the environment around the moving or static vehicle using the above specified sensor. The perception sub-system can be described as accessing the sensed raw value or environment, and creating a model of the environment, the planning sub-system uses the created model and determines on the related future actions. The purpose of control sub-system is to actuate the respective system to follow the actions chosen in the previous step. Localization, Object classification and detection are perception related tasks. Object detection can be further divided into 2D detection or 3D, based on cameras, RGB-D cameras or Lidar respectively. Deep learning based 2D object detection involves drawing of bounding boxes (x, y) around the detected objects in an image or video frame, whereas the 3D detection involves drawing of three dimensional bounding box (x, y, z) on an object, therefore estimating the exact position of the object in the 3D plane. Figure 1 shows an example of expected 3D bounding boxes on the Lidar point cloud and the similar bounding boxes on an image [3]. Deep learning has been widely accepted as an esteemed technique for image-based computer vision because of the development of the state of the art convolutional neural network based architectures [4]. The previous techniques or approach remains the pre-processing of the 3D point clouds data and adopting them into the data structure required for the existing deep learning algorithms, thus providing an output based on the algorithm [6]. Recent researches have proposed to process the Lidar point clouds directly on deep neural network without converting them to any representations. For example, [6] [8] proposed different form of deep net architectures, called

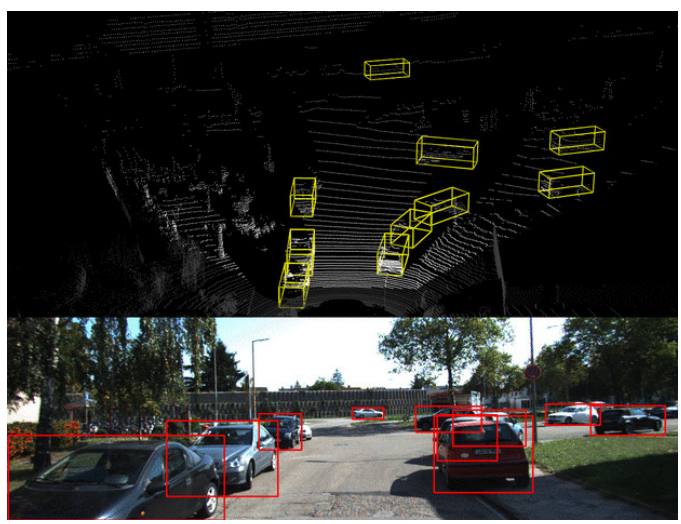

Fig. 1. 3-D vehicle detection visualization w.r.t to the camera and Lidar [3]

This is the author's manuscript of the article published in final edited form as:

Katare, D., & El-Sharkawy, M. (2019). Autonomous Embedded System Enabled 3-D Object Detector: (With Point Cloud and Camera). 2019 IEEE International Conference on Vehicular Electronics and Safety (ICVES), 1–6. https://doi.org/10.1109/ICVES.2019.8906442

as Pointnets and Frustum Pointnets respectively. These deep learning architectures have shown higher performance and have proved as benchmark for 3D perception based detection such as object classification and semantic segmentation. Pointnets architecture [6] proposed by Qi et al. is capable of both classification and semantic segmentation of 3d point clouds by learning the local and global feature vector from the raw point clouds. Zhou et al. presented VoxelNet [7], a deep learning architecture detecting 3D bounding boxes based on reading of Lidar Point clouds, here the lidar point clouds were divided into 3D voxel spaced equally. The architecture successfully detects and gives high performance for the car, cyclist and pedestrians. The most prominent 3D object detector Frustum-Pointnet [8] is presented by Qi et al., which predicts the bounding box on an object based on instance segmentation and the bounding box estimation. A similar method Pointfusion [9] is proposed by Xu et al. which utilizes the Pointnet [6] and ResNet [11] architecture for estimating the frustum and a 2D object detector respectively.

II. 2D Object Detection

The 2D object detection in an autonomous vehicles are primarily based on the single or multiple cameras connected to sense the environment or surrounding of the car. The 2D object detection architecture or algorithm requires the raw image as an input, and outputs the bounding box with the class or label of the detected object as shown in figure 2. In 2D object detection the bounding box is an axis-aligned rectangle, which is almost an exact fit on the position of the multiple objects or classes in that image, here the bounding box can be parameterized as (x_{min}, x_{max}, y_{min}, y_{max}) where (x_{min}, y_{min}) are the pixel coordinates of the bottom-left bounding box corner, and (x_{max}, y_{max}) are the pixel coordinates of the top-right corner. An example of the ground truth 2D bounding boxes from the KITTI dataset [3] is shown in Figure 2, in which the bounding boxes correspond to the class i.e. car, traffic light or motorcyclist. Recent State-of-art 2D object detector is based on the approach where the image is processed on the convolution neural network, extracting the feature map of the entire image. The selected object regions are passed onto these extracted feature map, and mapped onto the region feature vector, which on the basis of the class scores predicts the type of object and proposes the bounding box onto it.

III. 3D Object Detection

The 3D object detection can be based on the RGB-D cameras, radars, lidar or combination of such sensors. Here the deep learning algorithm requires the image with depth information or the lidar point cloud as an input, and outputs the 3D bounding box for the present objects in the line of sight. A 3D bounding box is very close to cubical shape covering the objects region on the sensors line of sight. For automotive applications, a 3D bounding box can be parameterized as (x, y, z, l, w, h, θ). Here the (x, y, z) is the 3D coordinates of the bounding box center, the (l, w, h) is length, width and height respectively of the bounding box, and θ is the

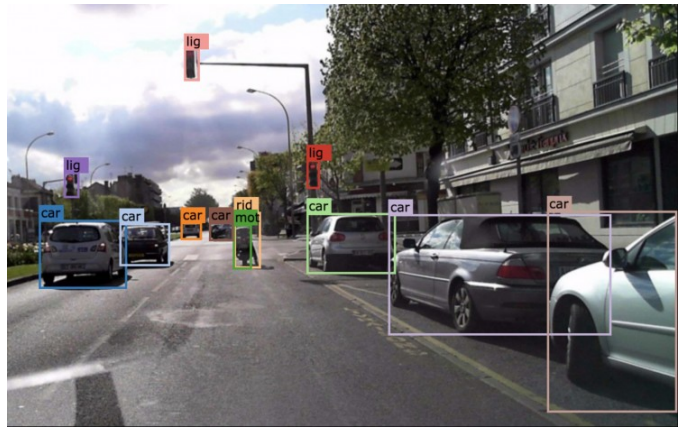

Fig. 2. An example of 2D object detection [source : xsens]

yaw angle of the bounding box. An example of the ground truth 3D bounding boxes for the vehicle detection application is shown in figure 1 from the KITTI dataset [3], where the boxes are visualized both in the Lidar point cloud along with the respective camera image. Most of the statistical or deep learning related algorithms for 3D object detection is however trained and evaluated on the KITTI dataset [3], which contains images and lidar point clouds collected from the forward facing stereo camera and velodyne Lidar.

As mentioned earlier the Frustum Pointnet architecture comprises of three primary steps: frustum proposal, instance segmentation and bounding box estimation. At the frustum proposal step, the 2D proposed region is extruded to extract the corresponding 3D frustum proposal, which contains the entire points of the lidar point cloud, that lies inside the 2D region when it is projected onto the image plane. This frustum proposal point cloud is then passed to the instance segmentation step, where the PointNet segmentation network carries out the binary classification for each point, thus predicting if the point is actual part or feature of the detected object. All true classified points are then passed to the bounding box estimation step, which utilizes the density of these points to estimate the position of the 3D bounding boxes. To estimate the center of the bounding box, the model regresses the residuals related to the segmented point cloud centroid. For the 3d bounding box dimensions and heading angle, a classification-regression hybrid based approach inspired by Mousavian et al. [12] is used.

Proposed Model: The model proposed in this paper uses Frustum Pointnet [8] as the base model. The model contains the Segmentation-Pointnet, Regressed PointNet (T-net), and an additional feature based on the inputs from the camera for the frustum proposal based on the known camera projection matrix. The proposed model uses extracted features from the image, for the position estimation of the 3D bounding boxes. The proposed model adds another block (figure 6) exploiting the global feature from the lidar and output feature vector of the image from the ResNet architecture. As shown in figure 6 the lidar global feature vector is passed as an input to a

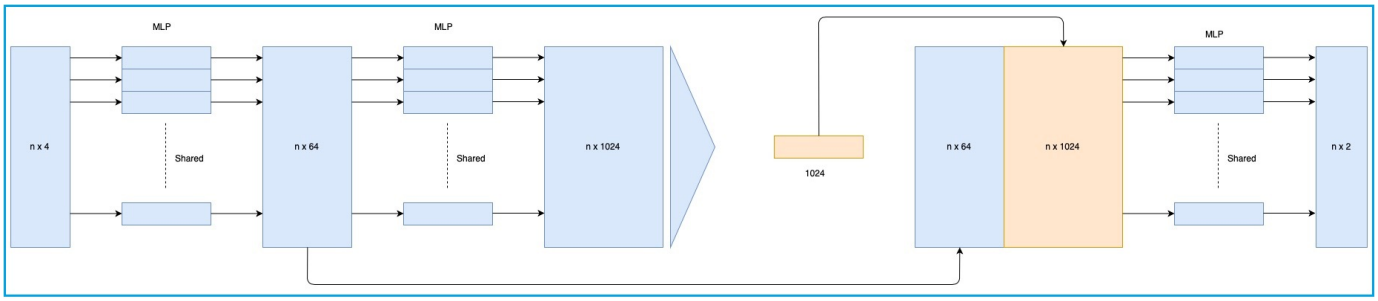

Fig. 3. Instance Segmentation pointnet

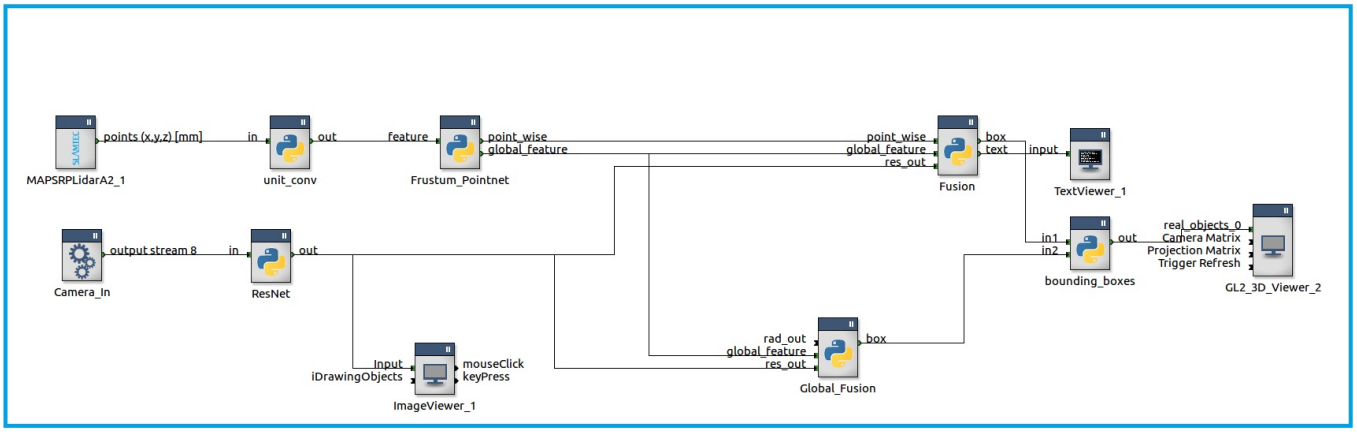

Fig. 4. Proposed model for Bluebox 2.0 in the RTMaps framework for Real time Detection

fully connected network with three layer which outputs the centroid position of the 3D bounding box w.r.t the center Position estimated by T-net. The same lidar global feature is also connected as an input to the image feature vector, which is then passed to the another fully connected network with three layers that outputs the value corresponding to the position estimation of 3D bounding box (length, width, height and angle). The image feature comprising of 2D bounding box described above, is the output of the 2D object detection or classification from the ResNet architecture.

Instance Segmentation Pointnet: The Instance Segmentation pointnet uses the extracted frustum point-cloud as an input and calculates the classification scores for each of the n input points estimating how probable they belong to the class or object considered for the case or scenario. The output classification-score is the measure of the predicted probability for a class among k different classes. For detection of multiple class in a scenario, this network utilizes the feedback from the 2-D detector: ResNet architecture (figure 4) for accurate segmentation as it reduces the task of finding similar point cloud densities or feature, matching the object considered int the scenario. The output from this network combines the global and local feature of the point clouds and segment it on the basis of category containing predefined maps or vector, for previously mentioned k different classes. This network can be considered as a modified or upgraded version of Pointnet. The network architecture is schematically illustrated in Figure 3.

T-Net: The input to T-net or Regressed Pointnet architecture (as shown in fig 5)is the object point cloud but it only utilizes the 3D coordinate points of the each given points. The regressed pointnet uses the segmented point clouds from the previous step and proposes the centroid position (x,y,z) for the chosen point cloud, for the 3D bounding box position. The estimation of the center of the object is performed here and transformation of the bounding boxes is performed to convert the estimated center into origin.

The bounding box estimation network (as shown in figure 6) is based on the regressed pointnet architecture and is similar to T-net with a difference in the last layer where the outputs are bounding boxes on the segmented objects with variables: center, size and yaw angle (c_x, c_y, c_z, l, w, h, θ). For the final prediction of the center and coordinate of the 3D bounding box, the outputs from the bounding box estimation network (bound-box-est-net) and T-net architecture is combined to calculate the absolute center on the segmented point cloud as described below.

$$C = C_{t-net} + C_{bound-box-est-net}$$

IV. Implementation

The proposed model is implemented in Python using the PyTorch framework [1] [16] as it is also available for the

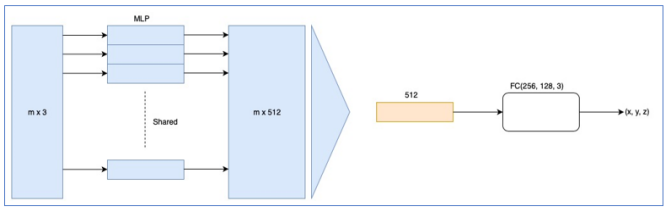

Fig. 5. T-net architecture

autonomous embedded platform using Ubuntu OS for arm devices. The multilayer perceptron is implemented using convolution module (nn.Conv1d) from pytorch. The points in the frustum point-cloud which lies inside the labelled 3D bounding boxes is recognized as the part of the object. This is the similar approach followed in Frustum-pointnet [8] to use the ground truths for the Instance segmentation block. For accessing the Image feature vector ResNet architecture comprising of 34 layers is implemented with torchvision [1][16]. The ResNet architecture uses pre-trained model for the initialization of the weights. The pre-trained model inputs image of size (224 , 224) with 3 channel.

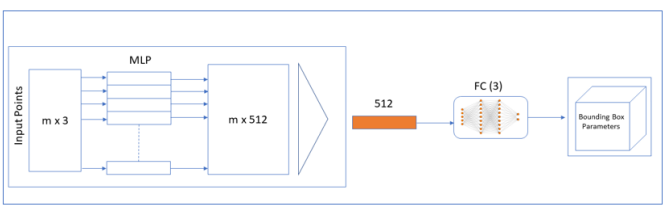

Fig. 6. Bounding box Estimation diagram

Loss Function: The loss function for the proposed model can be described as follows:

$$l = l_{isp} + (l_{t-net} + l_{box-center} + 0.1 l_{corner} + l_{box-size} + l_{clf})$$

here l_{isp} is the log loss corresponding to the output of Instance Segmentation pointnet, $l_{t\text{-}net}$ corresponds to the log loss for the 3D box centroid in the T-Net, $l_{box\text{-}center}$ corresponds to the log loss of the 3D box center for the added model into the existing frustum-pointnet architecture, l_{corner} is corner loss of the bounding box, $l_{box\text{-}size}$ corresponds to the predicted box size regression, l_{clf} corresponds to the log loss of the classification output.

Dataset and Training: As mentioned in the previous section the 3d object detection KITTI dataset [3] is used for training and validation of the architecture. The dataset consists of approximately 7400 training sets and 7500 testing sets. Here each set consists of sample images from forward facing camera and 3d point cloud using Velodyne Lidar. The training set are labelled for 2d and 3d ground truth with rectangle bounding boxes containing min and max values in x, y for classes such as cars, pedestrian, cyclist and the (x, y, z, l, w, h, θ) respectively. For this paper the training and validation ratio is chosen randomly with a split of 70% - 30% which amounts to approx 5100 sets for training approx 2000 sets for testing. The

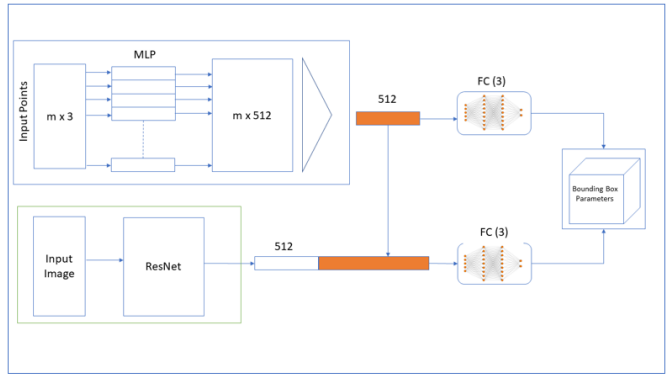

Fig. 7. Proposed Lidar and Camera fused bounding box estimation

proposed model is initially trained on a Ubuntu OS desktop pc with intel i7 processor comprising of 32 GB ram and nvidia GTX 1080 GPU and is finally deployed on NXP Bluebox (an autonomous embedded system) as described in the next section.

Data Augmentation: One of the most important factor to consider during the training of an architecture is the goodness of fit, which corresponds to how accurately an architectures predicted value matches to the ground truth values. This situation generally occurs when the architecture or model learns the noise feature instead of the signal features. For the proposed model the over-fitting is avoided by randomly rotating the labelled data from 3D bounding box and points in the frustum point cloud, by the angle uniformly sampled around the camera's Y-axis.

Optimizer: For optimization of the proposed model, the Adam optimizer is used. The adam optimizer follows an adaptive learning rate by using the first and second order moment of the gradient to update the weights. The starting learning rate of 0.0001 is chosen as random value along with the batch size of 32 frustum point clouds. After every 60 epochs the learning rate is reduced by half of it's previous value. The loss is calculated for the complete validation set on the completion of an epoch, the training is stopped once the validation loss shows constant values without major change for 60-100 epochs.

V. Embedded System Setup

This section presents the information about the complete setup used to implement the proposed model (fig 4) and clarifies the integration feasibility of RTMaps embedded and Bluebox 2.0; a complete System overview is described below.

A. Bluebox

Bluebox is an autonomous embedded platform developed by NXP semiconductor for development of sensor fusion based algorithms and applications. It is an ARM based development platform thus has the capability to function as the central computing unit or as the brain of the car. The development platform comprises three independent systems: LS2084A (a

computation unit), S32V234 (vision processor) and S32R274 (a radar processor).

The *LS2084A* is an embedded based computing processor; it comprises of eight ARM Cortex-A72 cores that provides the capability to handle high computation, network interfaces and high-performance data path [2].

The software enablement on the LS2084A and S32V234 SoC is deployed using the Linux board support package which is built using the Yocto framework. The LS2084A and S32V234 SoC are installed with Ubuntu 16.04 LTS which is a complete, developer-supported system and contains the complete kernel source code, compilers, tool-chains, with ROS kinetic and Docker package [2]. The platform overview is shown in Figure 8.

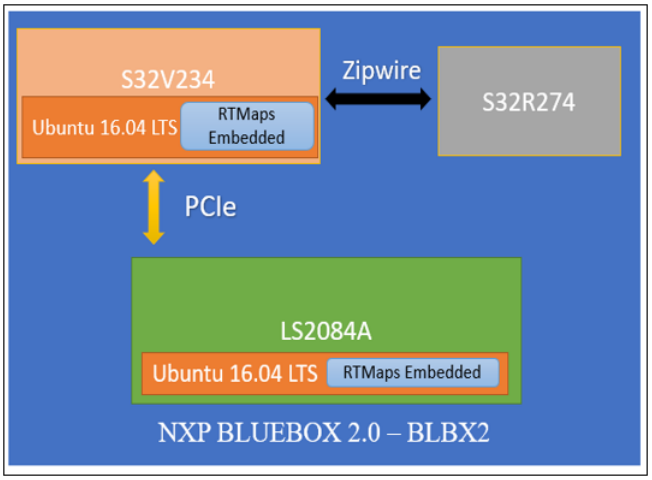

Fig. 8. System overview of NXP BLBX2 with RTMaps

B. RTMaps

RTMaps is designed for the development of multimodal based applications, thus providing the feature of incorporating multiple sensors such as camera, lidar, radar, imu. It has been tested for processing and fusing the data streams in the real-time or even in the post-processing scenarios. The software architecture consists of several independent modules that can be used for different situation and circumstance [10] [13].

RTMaps Component Library: : It consists of the software module which can be easily interfaced with the automotive and other related sensors and packages such as Python, C++, Simulink models and 3-d viewers. responsible for the development of an application [10] [13].

RTMaps Embedded: : It is a framework which comprises of the component library and the runtime engine with the capability of running on an embedded x86 or ARM capable platform such as NXP Bluebox, Raspberry Pi, DSpace MicroAutobox [10] [13].

For this paper the RTMaps embedded v4.5.3 platform is tested with NXP Bluebox, along-with the RTMaps studio operating on a computer which provides the graphical interface for the outputs to be demonstrated as images or video for the testing purpose. The connection between the Computer running RTMaps Remote studio and the Embedded platform can be accessed via a static TCP/IP as shown in Figure 8.

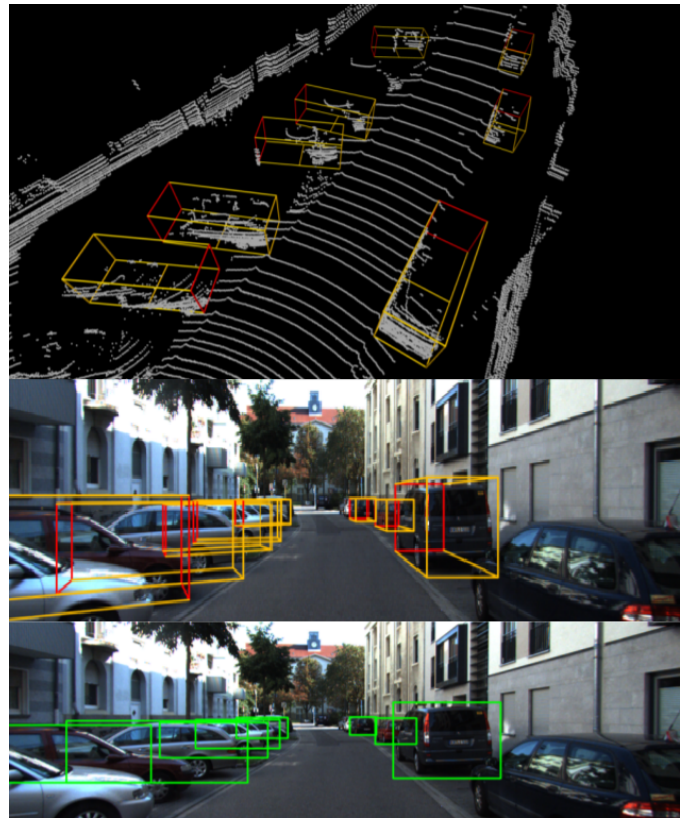

Fig. 9. Output from the 3D bounding boxes predicted on the KITTI dataset

VI. Results

The proposed model as mentioned in section III and IV is trained with KITTI train and KITTI test dataset [3] and evaluated on KITTI val dataset. This section presents the results for the trained and tested data. As shown in table I, the proposed method for Frustum PointNet implementation has performance close to the baseline model. Figure 9 shows the predicted 3D bounding boxes from the lidar point cloud, 3D bounding boxes from the image point of view and finally the 2D bounding boxes on the detected vehicle. Figure 10 and figure 11 shows the validation accuracy and precision score for the KITTI dataset. After 350 - 370 epochs the architecture outputs constant accuracy and precision score with minor change, and relatively does not change after 570-590 epochs. The proposed model performs well in most of the scenarios, however when the view point is at the curve of the street scenario, it predicts false bounding boxes, because of the false angle assumption. The results with respect to the KITTI dataset are shown in table I. As mentioned earlier the training of the architecture is done on desktop pc which results in generation of Pytorch model file. For the real-time deployment this file is transferred to bluebox and is reloaded using the RTMaps python packages for testing with the point cloud and

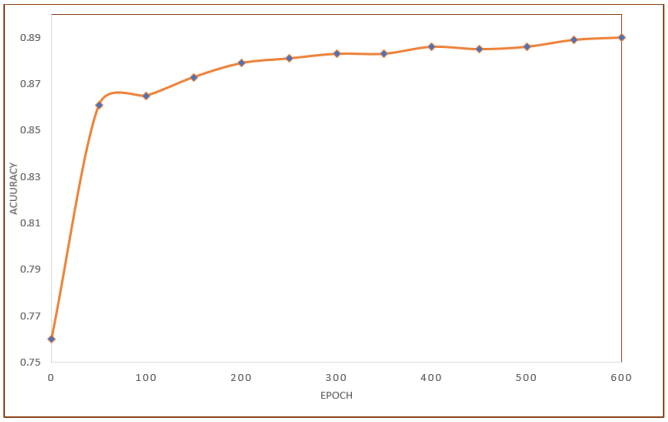

Fig. 10. Validation accuracy curve on KITTI dataset

TABLE I
OUTPUT OF KITTI DATASET ON BASELINE AND MODIFIED ARCHITECTURE

Method	Easy	Moderate	hard
Frustum-Pointnet	83.76	70.92	63.65
Proposed Frustum-Pointnet - KITTI	79.80	65.83	62.71
Proposed Frustum-Pointnet - KITTI (50%)	84.76	69.78	74.57

images from KITTI dataset. The real-time scenario can be tested using the Lidar and camera sensor as per the model proposed in figure 4. The lidar module shown in figure 4

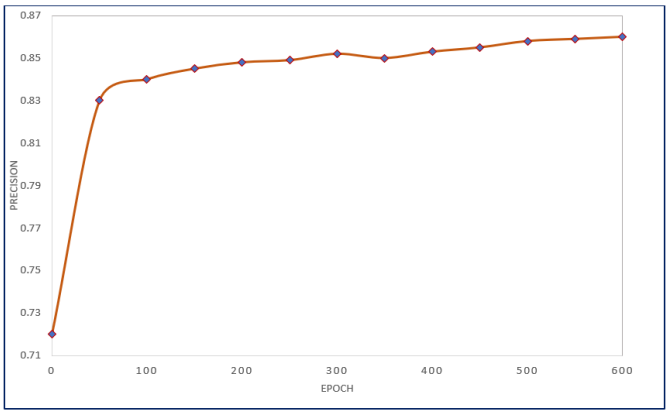

Fig. 11. Validation precision score curve on KITTI dataset

can be replaced with any CAN-Frame supported lidar module [5] available, to use the proposed model as a generic one, in that case the CAN bus component is used to establish the connection with the sensor and then the acquired inputs can be processed using the vendor specific lidar module (as shown in fig 4) which further splits the CAN frame information into desired parameters such as: raw point clouds or x,y,z dimensions and yaw angle. These parameter or sensed inputs is further passed through the proposed algorithm which then based on the input scenario predicts the 3D bounding boxes.

VII. CONCLUSION

This paper proposes an approach to combine the sensed inputs from the Lidar and camera and then process them through Frustum-pointnet architecture with the purpose of predicting 3D bounding boxes on the detected vehicles. The proposed detection model is designed in the RTMaps framework, to avail the functionality of real-time detection. The proposed method can be divided into two parts: First, a Lidar and camera based model implemented from the RTMaps component library which is capable of acquiring the real-time sensor values and Secondly, a deep learning algorithm implemented in the python module of the RTMaps components shown in Figure 4, which on the basis of the input values from the lidar and camera classifies and predict the bounding boxes on the detected vehicle.

REFERENCES

[1] Paszke, Adam, et al. "Automatic differentiation in pytorch." (2017).
[2] NXP Semiconductors,https://www.nxp.com/applications/solutions/ automotive accessed: 2019-03-14.
[3] Geiger, Andreas, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? the kitti vision benchmark suite." 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2012.
[4] Verbickas, Rytis, et al. "SqueezeMap: fast pedestrian detection on a low-power automotive processor using efficient convolutional neural networks." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2017.
[5] S. Kato, S. Tokunaga, Autoware on board: enabling autonomous vehicles with embedded systems, 9th ACM/IEEE Intenational conference on cyber-physical systems, Portugal 2018.
[6] Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.
[7] Zhou, Yin, and Oncel Tuzel. "Voxelnet: End-to-end learning for point cloud based 3d object detection." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.
[8] Qi, Charles R., et al. "Frustum pointnets for 3d object detection from rgb-d data." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.
[9] Xu, Danfei, Dragomir Anguelov, and Ashesh Jain. "Pointfusion: Deep sensor fusion for 3d bounding box estimation." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.
[10] Intempora.com. (2018). Intempora - RTMaps: For multi-sensor applications. https://intempora.com/prducts/rtmaps accessed: 2019-03-14
[11] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
[12] Mousavian, Arsalan, et al. "3d bounding box estimation using deep learning and geometry." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.
[13] Katare, Dewant, and Mohamed El-Sharkawy. "Embedded System Enabled Vehicle Collision Detection: An ANN Classifier." 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC). IEEE, 2019.
[14] Scikit-Learn (2018). "An easy-to-use library for machine learning algorithm development". https://scikit-learn.org/stable/supervised_learning accessed: 2019-03-14.
[15] Kukkala, Vipin Kumar, et al. "Advanced Driver-Assistance Systems: A Path Toward Autonomous Vehicles." IEEE Consumer Electronics Magazine 7.5 (2018): 18-25.
[16] PyTorch, https://pytorch.org/ accessed: 2019-03-14.
[17] Li, Bo, Tianlei Zhang, and Tian Xia. "Vehicle detection from 3d lidar using fully convolutional network." arXiv preprint arXiv:1608.07916 (2016).
[18] Kuutti, Sampo, et al. "A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications." IEEE Internet of Things Journal 5.2 (2018): 829-846.