https://doi.org/10.2352/ISSN.2470-1173.2020.16.AVM-257

LiDAR-Camera Fusion for 3D Object Detection

Darshan Bhanushali, Robert Relyea, Karan Manghi, Abhishek Vashist, Clark Hochgraf, Amlan Ganguly, Andres Kwasinski, Michael E. Kuhl, Raymond Ptucha; Rochester Institute of Technology; Rochester, New York/USA

Abstract

The performance of autonomous agents in both commercial and consumer applications increases along with their situational awareness. Tasks such as obstacle avoidance, agent to agent interaction, and path planning are directly dependent upon their ability to convert sensor readings into scene understanding. Central to this is the ability to detect and recognize objects. Many object detection methodologies operate on a single modality such as vision or LiDAR. Camera-based object detection models benefit from an abundance of feature-rich information for classifying different types of objects. LiDAR-based object detection models use sparse point clouds, where each point contains accurate 3D position of object surfaces. Camera-based methods lack accurate object to lens distance measurements, while LiDAR-based methods lack dense feature-rich details. By utilizing information from both camera and LiDAR sensors, advanced object detection and identification is possible. In this work, we introduce a deep learning framework for fusing these modalities and produce a robust real-time 3D bounding box object detection network. We demonstrate qualitative and quantitative analysis of the proposed fusion model on the popular KITTI dataset.

Introduction

The task of object detection is a fundamental function of environment based systems such as self-driving cars, autonomous robots, and augmented/virtual reality. The world around us is comprised of 3D geometry and sensing and understanding this 3D geometry is necessary for solving many of the problems in intelligent machines such as path planning, obstacle avoidance, and human-computer interaction. Reliability and safety with respect to these autonomous machines is highly dependent on its object detection capability. Considering the criticality of the 3D object detection (3DOD) task, robustness and real-time inference are the two major requirements for object detectors.

Data driven 3D deep learning has evolved greatly in the past few years because of the availability of huge amounts of data from variety of sensors in the form of well-designed datasets. Some major datasets for 3D deep learning include KITTI [1], nuScenes [2], Lyft Level 5 AV Dataset 2019 [3], and Waymo Open Dataset [4]. 3D sensors like RGBD cameras, LiDAR, and RADAR are widely available in the market with year over year improved quality and lower pricing. The *3D object detection (3DOD)* is defined as a task where objects in the scene are identified and localized by estimating their position in 3D space. Historically, most 3D visual computing techniques focus on a single modality (primarily LiDAR or image), thus reaching a limit in terms of accuracy and robustness. LiDAR and camera provide us with different information of the surrounding. LiDAR provides a 3-dimensional point cloud of the surrounding environment, data rich in depth information but lacking color and texture information. Camera provides more detailed texture, color and lighting information in the form of images or video, but lacks depth information. To combine the unique qualities of each modality, fusion based detection methods can be utilized.

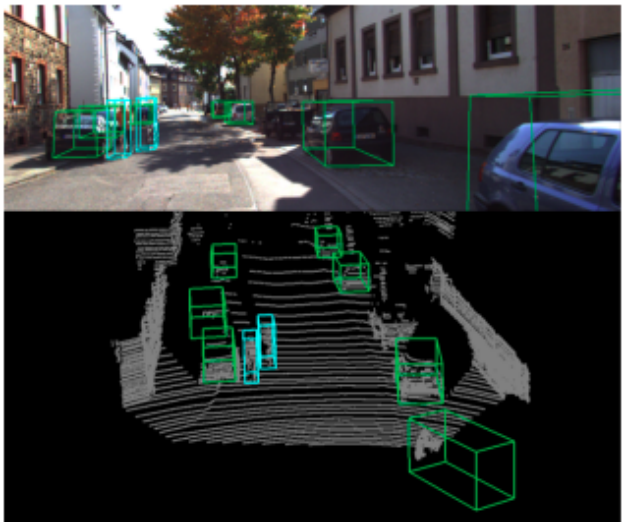


Figure 1. *Sample 3D object detection on the KITTI [1] dataset.*

We focus our efforts on developing a network which can smartly fuse the depth information from LiDAR with the textural information from the camera to achieve simultaneous 3DOD and localization. Specifically, we experiment with two techniques of sensor fusion: early and late fusion. Each fusion strategy has its own properties that pose challenges to deep architecture design while providing the opportunity for novel and efficient solutions. In this work, we discuss several methods to encode sensor data into the neural network and explore various fusion strategies along with key tuning parameters of the resulting fusion network architecture. We introduce a LiDAR-Camera fusion model called *combined fusion network* which provides improved 3D detection over prior methods. Our combined fusion network is an end-to-end learn-able network incorporating both early and late fusion. Figure 1 shows an example of 3D object detection using our combined fusion model on the KITTI dataset.

Related Work

Data driven deep learning based object detectors can be categorised into two categories, namely single-stage detectors and two-stage detectors. In a two-stage detector, the first stage involves a fast and lightweight region proposal network (RPN) that generates a set number of region proposals having high likelihood of existence of an object. These proposals are then passed through the second stage which extracts advanced features and performs

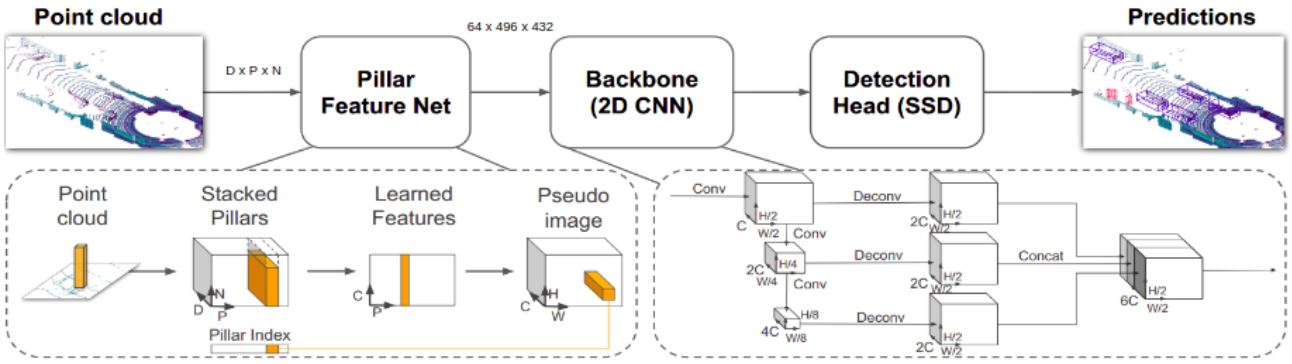


Figure 2. *PointPillar Network Overview [5], D = Number of point features (9), P = Number of points per pillar (100), N = Maximum number of pillars (12000), C = Number of pseudo-image feature maps (64), H, W = Height and Width of the feature maps (496×432).*

classification and regression on the generated proposals for identifying the object category and estimating the object's position in the 3D space. As the potential bounding box candidates can be infinite, RPN narrows down the potential bounding box candidates to a finite number. Contrary to two-stage detectors, single-stage detectors skip the RPN step and run feature extraction and classification on the input image simultaneously. Single-stage detectors are generally faster than two-stage detectors and hence more apt for real-time applications. Depending on the type of sensor data used by the encoder, 3DOD methods are further grouped as camera-only methods, LiDAR-only methods and dual modality methods using sensor fusion.

Mono3D [6] is a two stage camera-only approach. The first stage generates 3D object proposals in three different aspect ratios by leveraging the semantic segmentation of the image, location priors and other contextual features. The obtained proposals are fed to the second stage, consisting of a convolutional neural network (CNN) to generate final labels and oriented 3D bounding boxes. The network inference time is around 0.7 seconds proving to be inadequate for real-time applications.

Eduardo et al.[7] presented an overview of 3D object detection methods, the available datasets, research gaps and future directions. The paper also discusses the pros and cons of fusion based methods compared to LiDAR-only and camera-only methods. Voxelnet [8] first discretizes the point cloud into voxels, applies PointNet to each voxel and then uses a set of 3D convolutional layers on the 4D tensor to consolidate the features across the vertical axis. The inference time of Voxelnet is about 225ms, the majority of which is due to the expensive 3D convolution operations. AVOD [9] uses late fusion for fusing data from both a camera and a LiDAR to perform 3DOD. The point cloud data is converted into image modality by projecting the cloud onto the XY plane to obtain a bird's eye view (BEV) of the vehicle's surrounding. Although this allows a computationally less expensive single modal deep neural network for object detection, it sacrifices 3D geometrical information such as height for each of the points in the point cloud.

MV3D by Chen et al.[10] is another example of a two stage late fusion model. It converts the point cloud data into image modality by using BEV and front view (FV) projection techniques. BEV projection maps are encoded with height, intensity and density data of the points in the point cloud while FV projection maps contain height, distance and intensity information with respect to front view. BEV projection images are used by the region proposal network to generate proposals. FV images and RGB images are passed through separate encoders and the obtained feature vectors are passed into a deep fusion network. The inference time is about 0.36 seconds. F-PC-CNN[11] is an example of a two stage early fusion approach. 2D region proposals are generated from RGB images. A subset of points are selected from the point cloud based on the points which fall under the obtained 2D proposal and are passed into a deep CNN network to obtain oriented 3D boxes. Some methods like F-Pointnet and RoarNet leverage the data from LiDAR and Camera in a sequential fashion instead of direct fusion. We refer to such work as sequential fusion.

Pointpillar Encoder - Baseline

The Pointpillar encoder [5] is a PointNet [12] based LiDAR point cloud encoder. It converts the input 3D point cloud into 2D pseudo BEV projection images by extracting learned features across the Z-axis of the point cloud. The Pointpillar encoder [5] authors demonstrate its effectiveness by developing a lean downstream network to perform 3DOD task using LiDAR data as input. The result is an end-to-end learnable network with an inference time of just 16 ms per point cloud. Its real time performance is due to two factors, the use of multi-box single shot detector (SSD) network and the discretization strategy used for converting the unbounded point cloud data into bounded pillars instead of voxels. It significantly outperforms all other state of the art networks on KITTI 3D object detection benchmark [13] using only LiDAR point cloud data.

The Pointpillar 3D object detector architecture as shown in the Figure 2 consists of three components, Pillar Feature Net (PFN), 2D CNN Backbone and SSD Detection head. First the point cloud is discretized into evenly spaced grid in the $X-Y$ plane into multiple vertical columns also known as pillars. For each non-empty pillar, the maximum number of N points residing in the pillar is randomly sampled in order to achieve uniform distribution of points across the space and reduce computational complexity. If the number of points in a pillar is less than N, zero padding is applied. This results into a dense tensor of shape $(D \times P \times N)$ where P is the number of non-empty pillars per sample and D is the number of features per point. The PFN extracts

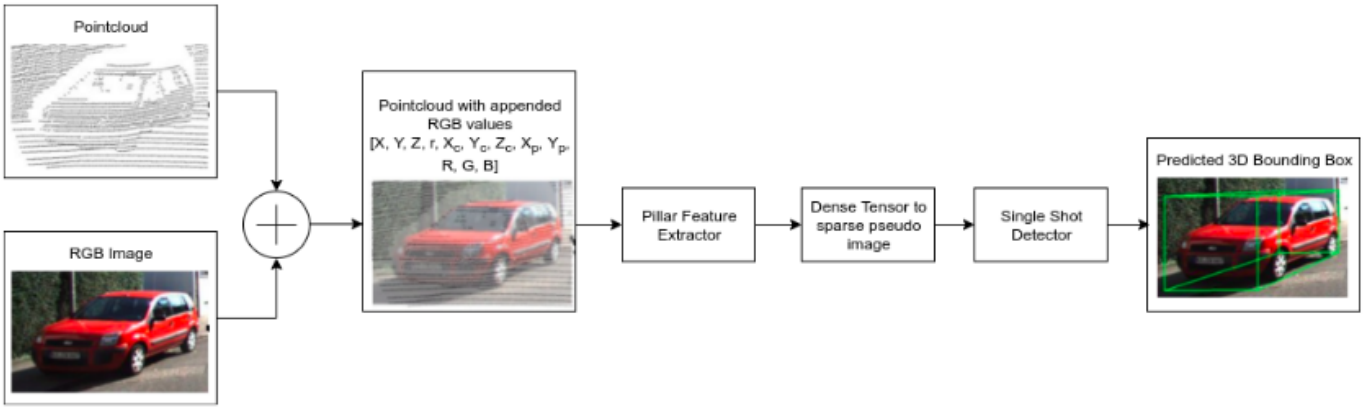


Figure 3. *The structure of the proposed early fusion network.*

Table 1: Network Configuration.

	Car Network	Cyclist-Pedestrian Network
Voxel size	[0.16, 0.16, 4]	[0.16, 0.16, 3]
Max number of pillars per point cloud (P)	12000	12000
Max number of points per pillar (N)	100	100
Default anchor box sizes (w, l, h)	Car : [1.6, 3.9, 1.56]	Cyclist : [0.6, 1.76, 1.73] Pedestrian: [0.6, 0.8, 1.73]

learned features across the vertical columns by applying PointNet on each of these pillars yielding a $C \times P$ feature vector where C is the number of pseudo-image features maps to be generated. This eliminates the need for handcrafted features and thus making the network end-to-end learnable. These learned features are then consolidated based on their pillar index and converted into a pseudo-image like BEV feature maps of size $C \times H \times W$ which are then passed through the 2D CNN backbone. Each feature map is of dimension $H \times W$. The backbone network is a top-down network consisting of series of convolution and deconvolution layers which generate features at different resolutions. The top-down features are then concatenated and used by the SSD head to generate 3DOD predictions.

Proposed Architecture and Implementation

An efficient fusion strategy needs to be implemented to overcome the limitations of single sensor detection networks and leverage the use of information from multiple sensor sources. LiDAR and camera provide us with complementary information for robust 3DOD. Camera captures detailed texture information of the surrounding scene and while LiDAR provides us with depth information.

We are primarily inspired by and modify the architecture of Pointpillar [5] to enable multi-modal data input. The real-time performance of this network was one of the main reason behind choosing PointPillar network for data fusion experiments. As discussed before, the three stage structure of the PointPillar network allows implementation of data fusion strategies. The augmented LiDAR point l has $D = 9$ features in the default PointPillar network,

$$l = [\ x,\ y,\ z,\ r,\ X_c,\ Y_c,\ Z_c,\ X_p,\ Y_p\ ] \tag{1}$$

where $X_c,\ Y_c,\ Z_c$ are offset from the arithmetic mean of all the points in the pillar, X_p and Y_p are the x and y offset from the pillar center (x, y) and r is the reflectance value from LiDAR.

In the sections below, the three fusion strategies, namely, early fusion, late fusion, and combined fusion are discussed. Figure 3, Figure 4 and Figure 6 show the architecture of the three fusion models respectively. All the fusion models use Pointpillar [5] to encode point cloud data into a pseudo-image like feature. These features are passed into the SSD network which performs the 3DOD task and predicts the object classes for the objects present in the scene with their 3D bounding box coordinates. SSD uses the same backbone as [5]. Table 1 shows the configuration setup for the baseline as well as the fusion models. All the weights are initialized with uniform random distributions.

Early Fusion

Early fusion combines the raw data from the two sensors before feeding into the neural network. The Pillar Feature Net (PFN) is the first stage of the PointPillar network. PFN is responsible to extract learned features from the input point cloud across the Z-axis and convert the point cloud to a pseudo-image. This structure enables early fusion by modifying the PFN stage to take an augmented point cloud as input. Figure 3 shows the architecture of the Early fusion model. The input point cloud is augmented with corresponding RGB values obtained from the image. To obtain the RGB value for a particular point, the 3D point is firstly projected onto the image frame using the rotation and projection matrices provided in the KITTI development kit and then the point l is augmented with the RGB value of the pixel location where the point overlaps on the image frame.

A point $x = (x, y, z, 1)^T$ in LiDAR coordinate frame to a point $y = (u, v, 1)^T$ in the i^{th} camera image frame is given as

$$y = P_{rect} * R_{rect} * x \tag{2}$$

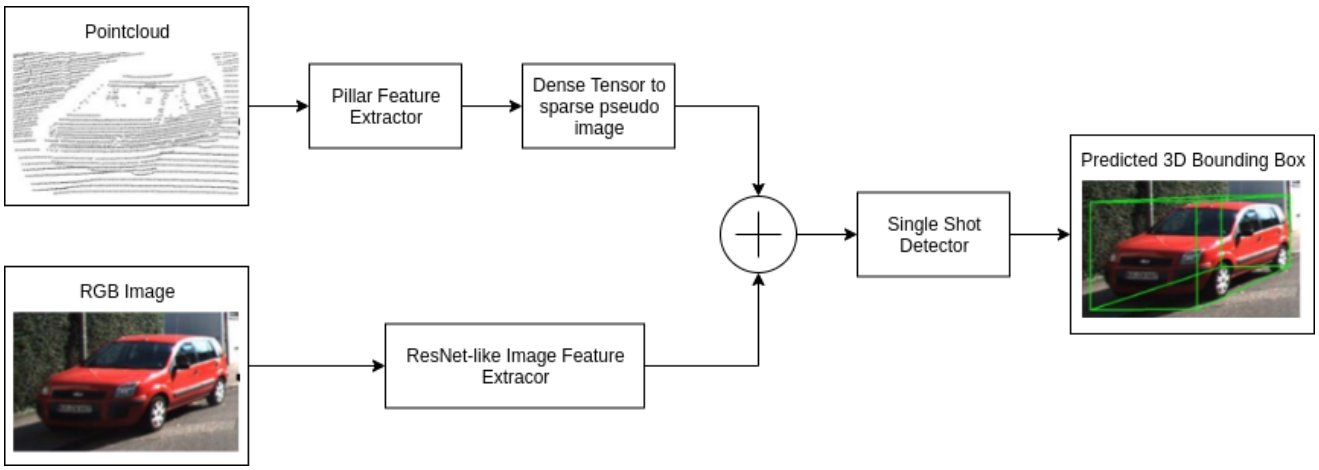


Figure 4. *The structure of the proposed network showing late fusion of the two modalities.*

where P_{rect} is the rectified projection matrix, and R_{rect} is the rectified rotation matrix.

Additionally, a mean filter of kernel size 5×5 is convolved across the image before overlaying the point cloud onto the image. This provides a normalization effect for the RGB augmentation. Experiments showed that a 5×5 kernel gives the best result. The final augmented LiDAR point l has $D = 12$ features:

$$l = [\ x,\ y,\ z,\ r,\ X_c,\ Y_c,\ Z_c,\ X_p,\ Y_p\ ,\ R,\ G,\ B\ ]. \quad (3)$$

The obtained augmented point cloud is then passed into the network to perform 3DOD.

Late Fusion

Late fusion in deep networks is defined whereby the higher dimensional features from the individual sensor encoder networks are combined into a joint feature vector. This joint feature vector is then used by the object detector network to predict 3D object detection parameters. Late fusion is popular for fusing data of different dimensions such as image, point cloud and analog data, as vector representations are easily concatenated with one another. Late fusion allows for the use of specialized encoding of each modality for independent processing of raw sensor data. This freedom can potentially generate more reliable vector representations of the input data. This strategy assigns weights to the inputs automatically during the training phase without the need for any manual tuning. Unlike the early fusion model, late fusion can perform 3DOD even if the data from one of the two modalities is missing.

Figure 4 shows the architecture of the late fusion model. The first branch consists of the Pointpillar encoder [5] which encodes the raw point cloud into 64 image-like feature maps each of size 496×432. The second branch consists of a variant of ResNet [14] to extract features from the RGB images. 128 feature maps each of size 28×28 are generated after the second block of the ResNet18 architecture (Figure 5) and are up-scaled to 496×432 using bilinear interpolation. These up-scaled feature maps are concatenated with the 64 feature maps obtained from the point cloud data and passed through a SSD network to generate 3D bounding boxes and predict the object class.

Figure 5. *ResNet CNN architecture for image feature extraction.*

Combined Fusion

Our combined fusion model incorporates both late fusion and early fusion in the same network. Figure 6 shows the architecture of the combined fusion model where early fusion is performed in the first branch, noted as "EF" and late fusion is noted as "LF". The first branch consisting of the PFN takes the augmented point cloud with RGB values as input and generates 64 BEV pseudo-images. The second branch is a ResNet [14] based feature extractor for the RGB image. The feature maps are concatenated as per the late fusion model and passed through the SSD network to output 3D object scores and bounding boxes.

Loss

In the context of 3D object detection task, loss is comprised of two parts, the classification loss for object category mismatch and the localization loss for bounding box offset prediction. The model loss is the weighted sum of the localization loss (SmoothL1) and the classification confidence loss (Focal loss). Focal loss L_{cls} is defined as:

$$L_{cls} = -\alpha * (1 - p^{\alpha})^{\gamma} * log(p^{\alpha}) \quad (4)$$

where p^{α} is the class probability of an anchor.

The SmoothL1 loss is used for bounding box regression in most of the object detection networks like Fast RCNN [15], Faster RCNN [16], and SSD [17]. It is observed that the SmoothL1 loss is less sensitive to outliers than L2. The smoothL1 loss L_{loc} is defined as:

$$L_{cls} = \begin{cases} |x|\ if\ |x| > \alpha \\ \frac{1}{|\alpha|} x^2\ if\ |x| <= \alpha \end{cases} \quad (5)$$

where α is a hyper parameter and is usually taken as 1. Combin-

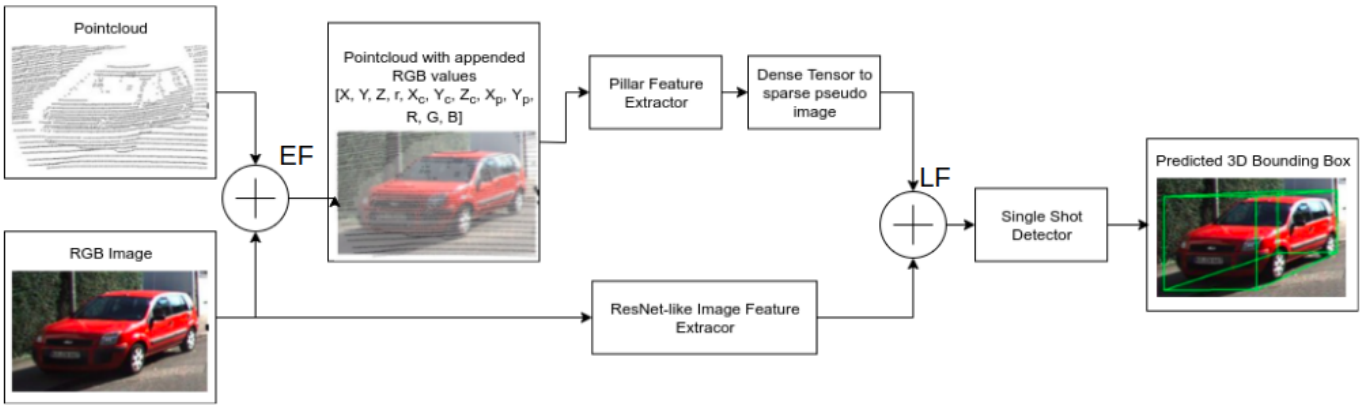


Figure 6. *The structure of the proposed combined fusion network. EF = Early Fusion, LF = Late Fusion.*

ing (4) and (5), the total loss is:

$$L = \frac{1}{N_pos}(\beta_c ls * L_c ls + \beta_l oc * L_l oc) \tag{6}$$

where N_pos is the number of positive anchors, $\beta_l oc$ = 2, and $\beta_c ls$ = 1.

The Adam optimiser is used for the optimization of the loss function during training. The initial learning rate is set as 2×10^{-4} and the decay factor is set as 0.85 every 15 epochs. The network is trained for 220 epochs with a batch size of 2.

Evaluation Metrics

Intersection over Union (IoU) and Average Precision (AP) are commonly used to evaluate the performance of object detectors. IoU is defined as the area of intersection of the predicted and the ground truth box divided by the joint area occupied by both boxes:

$$IoU = \frac{(area\ of\ overlap)}{(area\ of\ union)}. \tag{7}$$

According to the norms defined by the KITTI benchmark, a detected box is considered as a true positive (TP) if its IoU with the groundtruth annotation is greater than the threshold of 0.7 for "Car" class and 0.5 for "Pedestrian" and "Cyclist" class. If the IoU is less than the threshold it is considered as a false positive (FP). A false negative (FN) is defined when the network fails to generate a box for a real object in the scene. Precision is defined as the ratio between the number of detected TPs and the total number of detections:

$$Precision = \frac{TP}{TP+FP}. \tag{8}$$

Recall is defined as the ratio between the number of TPs and all the ground truth boxes:

$$Recall = \frac{TP}{TP+FN}. \tag{9}$$

As per the KITTI Benchmark specifications, average precision (AP) is defined as the mean of the precision values at each of eleven defined recall values depending on the eleven different confidence thresholds. The maximum precision value corresponding to the given recall value is used for calculating the mean.

$$AP = \frac{1}{11}\sum_{r=0}^{10} \max\left[p\left(\frac{r}{10}\right)\right] \tag{10}$$

where $p(\cdot)$ is the calculated precision corresponding the particular recall value.

Table 2: Inference time per point cloud.

Method	Inference Time (ms)
Baseline (LiDAR only)	16
Early fusion	32
Late fusion	34
Combined Fusion	34

Results

Dataset

All the training and evaluations are done using the KITTI object detection benchmark dataset [1]. The KITTI dataset consists of six hours of real-world road traffic scenarios recorded at 10Hz-100Hz using a variety of sensors. It has about 200k 3D object annotations for image data from four HD cameras as well as corresponding point cloud data from a Velodyne LiDAR. The cameras, laser scanner and localization system are all calibrated and synchronized. Each scene has up to 15 cars and 30 pedestrians. The data is split into 7481 training samples and 7518 testing samples. The ground truth labels for the test set are not publicly available. Evaluation on the test data can be obtained by submitting the prediction results to the online benchmark server. We follow the training set split provided by MV3D [10] which is commonly used by others. The training set is split into a new training set of 3712 samples and a validation set of 3769 samples. Depending on the amount of truncation and occlusion in the scene, each annotation label is marked as either "fully visible", "partly occluded" or a "largely occluded" target. The annotations are further divided into three groups considering the detection difficulty for that particular annotation as "Easy", "Moderate" or "Hard". For our experimentation, we use the left camera image and the

Table 3: 3D detection benchmark scores on the KITTI validation set (3769 examples), trained on the KITTI training set (3712 examples).

Method	mAP (Mod.)	Car (IOU=0.5)			Pedestrian (IOU=0.5)			Cyclist (IOU=0.5)		
		Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
Baseline	58.03	81.49	67.89	65.99	60.80	56.31	50.42	70.18	50.04	47.07
Early Fusion	60.30	84.15	70.34	**66.12**	62.79	**58.42**	51.22	72.43	52.16	**47.83**
Late Fusion	57.20	81.23	66.42	63.11	59.84	56.33	50.12	69.13	48.86	46.02
Combined Fusion	**60.44**	**84.67**	**70.84**	64.88	**63.14**	58.13	**51.88**	**72.44**	**52.34**	47.17

Table 4: BEV detection benchmark scores on the KITTI validation set (3769 examples), trained on the KITTI training set (3712 examples).

Method	mAP (Mod.)	Car (IOU=0.5)			Pedestrian (IOU=0.5)			Cyclist (IOU=0.5)		
		Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
Baseline	66.67	90.36	82.16	81.85	69.57	64.15	58.12	73.82	53.70	49.97
Early Fusion	70.12	**92.85**	86.09	83.22	**74.45**	67.90	60.20	**75.60**	**56.38**	**51.34**
Late Fusion	66.59	90.20	80.96	81.03	70.32	63.92	56.44	73.12	54.89	49.13
Combined Fusion	**70.59**	92.15	**87.42**	**83.42**	74.33	**68.12**	**60.43**	73.55	56.24	50.88

corresponding cropped point cloud from the Velodyne LiDAR.

Fusion Results

Out of the seven classes defined in the KITTI dataset; "Car", "Pedestrian", and "Cyclist" are the three dominant classes. We train and evaluate our models on these three classes. We evaluate the models using the official KITTI evaluation detection metrics for BEV and 3D detection. Each annotation in the KITTI dataset is categorised as either "Easy", "Moderate" or "Hard" depending on the difficulty level of detecting that particular object in the scene. Results for 3D object detection are shown in Table 3 and BEV object detection task are shown in Table 4.

Results show that Early fusion performs better than late fusion for both tasks and combined fusion model gives the best results. Compared to the Baseline LiDAR only method, we see an improvement of about 2%-4% in the early fusion model. Similar results are shown for the late fusion model. As indicated in Table 2, inference run-time of the three fusion models is between 32ms to 34ms, making them suitable for real-time applications. All inference times are measured on an Alienware laptop with an Intel i7 CPU and 1080ti GPU.

Conclusion

In this paper, the PointPillar architecture is extended to incorporate both image pixel data along with LiDAR point cloud data. We experiment and evaluate different LiDAR-Camera fusion strategies for the task of 3D object detection. The network is trained in an end-to-end manner and does not require any hand crafted features. The network takes inputs from two sources, LiDAR and camera and outputs 3D bounding box coordinates and the class label of the detected object.

The results show that early fusion performs better than late fusion and combined fusion gives the best results. The inference time for the three models ranges between 32ms to 34ms, suitable for real-time applications.

The introduced network is scalable to more input modalities such as BEV projection images, infrared images, and optical flow images. Recent advances in imaging RADARs such as 80 GHz RADARs can optionally output point cloud type data with an additional velocity field. It is anticipated that such RADAR data can be fused with LiDAR or image data to further increase the accuracy and robustness of the proposed model.

Acknowledgments

This research is sponsored in part by a grant from Toyota Material Handling of North America.

References

[1] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013. [Online]. Available: https://doi.org/10.1177/0278364913491297

[2] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," *CoRR*, vol. abs/1903.11027, 2019. [Online]. Available: http://arxiv.org/abs/1903.11027

[3] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet, "Lyft level 5 av dataset 2019," urlhttps://level5.lyft.com/dataset/, 2019.

[4] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," 2019.

[5] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019, pp. 12 689–12 697.

[6] T. He and S. Soatto, "Mono3d++: Monocular 3d vehicle detection with two-scale 3d hypotheses and task priors," 2019.

[7] E. Arnold, O. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "A survey on 3d object detection methods for autonomous driving applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, pp. 1–14, 01 2019.

[8] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," *CoRR*, vol. abs/1711.06396, 2017. [Online]. Available: http://arxiv.org/abs/1711.06396

[9] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3d proposal generation and object detection from view aggregation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 1–8.

[10] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 6526–6534.

[11] X. Du, M. H. Ang, S. Karaman, and D. Rus, "A general pipeline for 3d detection of vehicles," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 3194–3200.

[12] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 77–85.

[13] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, June 2012, pp. 3354–3361.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[15] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 1440–1448.

[16] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, June 2017.

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *Lecture Notes in Computer Science*, p. 21–37, 2016. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2

Author Biography

Darshan Bhanushali is a second-year computer engineering M.S. student at the Rochester Institute of Technology. He obtained his Bachelor of Engineering in Electronics from the University of Mumbai in 2016. His research interests include computer vision, robot autonomy, and sensor modelling.

Robert Relyea is a second-year computer engineering M.S. student at the Rochester Institute of Technology. He obtained his B.S. in computer engineering at the Rochester Institute of Technology in 2018. His research focus includes applications of computer vision and deep learning techniques to robot perception and localization.

Karan Manghi is a third-year computer science M.S student at the Rochester Institute of Technology. He studied B.S in information technology at the University of Mumbai and received his degree in 2017. He is most interested in mobile robot path planning and navigation, computer vision and algorithm design for navigation of a mobile robot.

Abhishek Vashist is currently pursuing his Ph.D. degree in Department of Computer Engineering at Rochester Institute of Technology, Rochester, NY, USA. He received his M.S. degree in Electrical Engineering from Rochester Institute of Technology, Rochester, NY, USA in 2017 and B.Tech. degree from ABES Engineering College, India in 2014. His research interest includes machine learning, 60 GHz mmWave, and computer architecture.

Clark Hochgraf is an Associate Professor in the Department of Electrical, Computer, and Telecommunications Engineering Technology at Rochester Institute of Technology. He earned his Ph.D. in Electrical Engineering from the University of Wisconsin-Madison in 1997. He was a principal engineer at Visteon, held research positions at Westinghouse and General Motors, and has 12 US patents. He is a certified master trainer in Educational Mobile Computing. His areas of research interest include intelligent vehicles, power conversion, and education of engineers.

Amlan Ganguly is currently an Associate Professor in the Department of Computer Engineering at Rochester Institute of Technology, Rochester, NY, USA. He received his PhD and MS degrees from Washington State University, USA and BTech from Indian Institute of Technology, Kharagpur, India in 2010, 2007 and 2005 respectively. His research interests are in robust and scalable intra-chip and inter-chip interconnection architectures and novel datacenter networks with emerging technologies such as wireless interconnects. He is an Associate Editor for the Elsevier Journal of Sustainable Computing Systems (SUSCOM) and the MDPI Journal of Low Power Electronics and Applications (JLPEA). He is a member of the Technical Program Committee of several conferences such as International Green and Sustainable Computing (IGSC) and International Network-on-Chip Symposium (NOCS). He is a member of IEEE.

Andres Kwasinski received in 1992 his diploma in Electrical Engineering from the Buenos Aires Institute of Technology, and the M.S. and Ph.D. degrees in Electrical and Computer Engineering from the University of Maryland at College Park, in 2000 and 2004, respectively. He is currently a Professor at the Department of Computer Engineering, Rochester Institute of Technology. He is a Senior member of IEEE, Chief Editor of IEEE SigPort and Area Editor of the IEEE Signal Processing Magazine.

Michael E. Kuhl is a Professor in the Department of Industrial and Systems Engineering at Rochester Institute of Technology. He earned his Ph.D. in Industrial Engineering from North Carolina State University in 1997. His areas of research interest including simulation modeling and analysis, the design and development of autonomous material handling systems, and application of simulation to supply chain, healthcare, and cyber security systems.

Raymond Ptucha is an Assistant Professor in Computer Engineering and the Director of the Machine Intelligence Laboratory at the Rochester Institute of Technology. His research includes machine learning, computer vision, and robotics, with a specialization in deep learning. Ray was a research scientist with the Eastman Kodak Company where he worked on computational

imaging algorithms and was awarded 31 U.S. patents. He earned a Ph.D. in computer science from RIT in 2013. Ray was awarded an NSF Graduate Research Fellowship in 2010 and his Ph.D. research earned the 2014 Best RIT Doctoral Dissertation Award. Ray is a passionate supporter of STEM education, an NVIDIA-certified Deep Learning Institute instructor, and the Chair of the Rochester area IEEE Signal Processing Society.

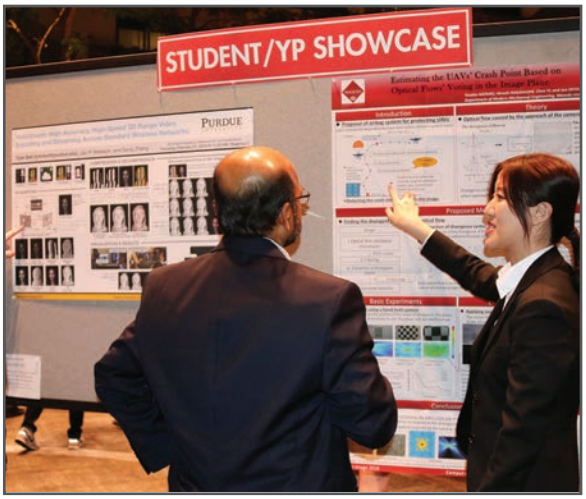
JOIN US AT THE NEXT EI!

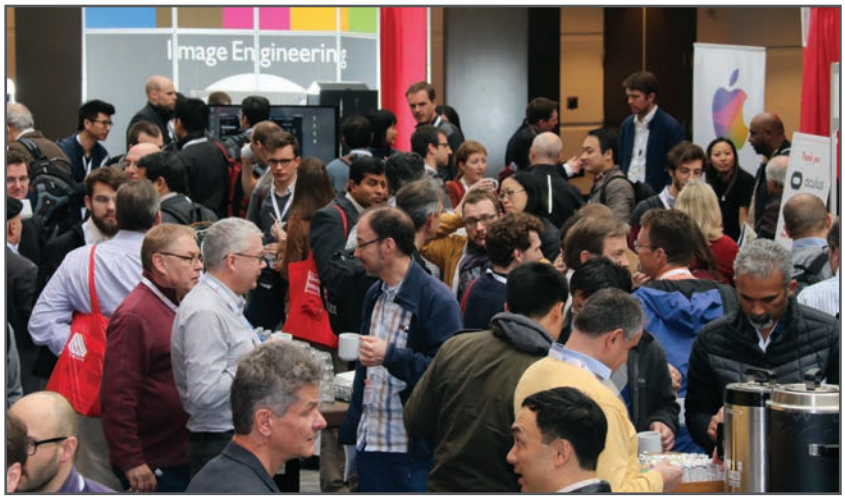
IS&T International Symposium on

Electronic Imaging

SCIENCE AND TECHNOLOGY

Imaging across applications . . . Where industry and academia meet!





• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •
• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •

www.electronicimaging.org

IS&T
imaging.org