# Autonomous Driving Perception Pipeline: Implementation of 3D Object Detection with Camera and LiDAR data

Bernard Lange
Department of Aeronautics & Astronautics
blange@stanford.edu

## Abstract

*Autonomous driving is heavily relying on 3D object detection to provide safe-critical information required in other modules of the vehicle, such as prediction, and planning. As part of this project, we have re-implemented popular PointFusion architecture which performs 3D object detection. One of the strengths of this approach is no need for domain expert knowledge. We have compared different types of fusion networks, which fuse information from the image and point cloud encoders, and proposed a novel way to integrate adversarial training and map information into the pipeline. We trained the architecture on the Nuscenes dataset and achieved satisfactory performance.*

## 1. Introduction

Autonomous driving, aka driving without any human intervention, promises to improve the safety on the road, optimize the traffic flow, and provide transportation opportunities to all. It has motivated the work of many companies, researchers, and students, including myself. "Traditional" autonomous vehicle prototype contains a range of different sensor modalities each of them providing different types of representation of the environment, namely cameras (RGB representation), lidars (dense point cloud representation), and Radars (sparse point cloud representation). Example of the vehicle is shown in Figure 1. Measurements from these sensors are then passed through a robotic stack to execute a safe and mission-driven maneuver. Such a robotic stack consists of the following modules: object detection, localization, sensor fusion, state estimation, prediction, planning, and control to say the least. They all are running in parallel and continually feed themselves information.

As part of my research, we focus on processing the point clouds into the occupancy grids representations and experimenting with environment prediction, scene generation, and occlusion inference approaches. When we have used information from other modules of the stack (e.g. object detection), we have always used it in a black-box fashion. Even though we am familiar with the general idea of how they function, we have never implemented them ourselves. Hence, as part of this project, we am re-implementing the perception part of the autonomous vehicle stack, namely 3D Object Detection, fine-tuning it, and extending it to improve the performance. The implementation is based on the PointFusion [15] architecture, due to its very general design and no need for domain expert-knowledge. The considered extensions are the incorporation of road layouts, and adversarial training. Prior information about the road layouts should be informative to the detection process and an adversarial component should help "weed out" unrealistic predictions.

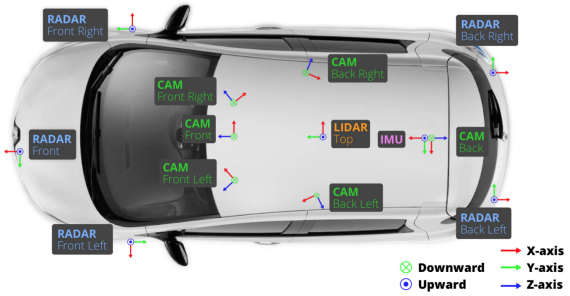


Figure 1. Example of the vehicle setup used for collecting Nuscenes dataset. Image credits: Nuscenes[1].

The key contributions of this project are as follows:

- implemented a PointFusion pipeline,
- compared different networks to fuse point cloud and image information,
- introduced adversarial extension of the PointFusion training pipeline.

Our implementations are trained and evaluated on the Nuscenes dataset [1]. The code is available at https://github.com/BenQLange/Pointfusion.pytorch.

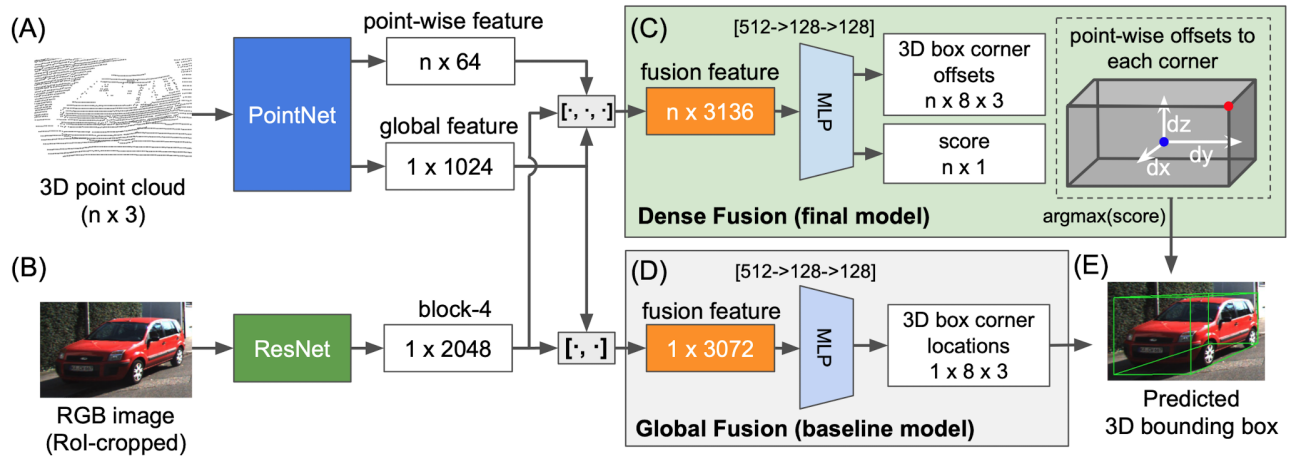


Figure 2. PointFusion 3D object detection pipeline: (A) Point cloud representation encoder (PointNet) which receives point cloud points inside the 2D bounding box. (B) Image encoder (ResNet-50) which is provided a cropped 2D image detected by the 2D object detector. (C) Improved Fusion Network approach where for each point cloud, the locations offset to the corners is predicted. (D) General Fusion Network where for each crop we predict one set of corner locations. Image credits: PointFusion [15].

## 2. Related Work

There has been a considerable amount of work done in the 3D object detection field. Majority of them builds upon the work done in 2D Object Detection field. Most popular image processing backbones are VGG [12], ResNet [4], and Inception [13]. These backbones are then used to build meta-architectures, such us Mask R-CNN [3], YOLO [9], and SSD [6].

In the 3D object detection pipeline, the most popular approaches are using camera and lidar sensor modalities to provide accurate detections. Lidar point clouds are encoded with networks like PointNet [8]. Similarly to the 2D object detection, output from the image and point cloud encoders are used in 3d object detection meta-architectures such as PointPillars [5], PointRCNN [11], PointFusion [15], and SecondNet [16]. There is also a considerable amount of work done on stereo 3d object detection and even monocular 3d object detection which are beyond the scope of this project.

In this project, we are using Mask-RCNN to detect 2D regions of interest from images, and ResNet-50 to encode cropped images. Information from those models is then used in the PointFusion pipeline to predict the 3D bounding box of the detected objects.

## 3. Approach

### 3.1. 3D Object Detection Task

In the 3D object detection task, the objective is to identify the location of the 3D box which is represented by its 8 corners. The network attempts to directly regress the locations of the corners of the 3D bounding box in a frame by optimizing the traditional for this task loss:

$$L_{3D} = \sum_i smoothL1(\mathbf{x}_i^*, \mathbf{x}_i) + L_{stn}$$

where $\mathbf{x}^*$ is ground truth, $\mathbf{x}$ is the prediction, and $L_{stn}$ is the orthogonality regularization introduced by [8]. However, as noted by [15], there is a large variance in predictions targets depending on the 3D object position in the scene. Hence, for each point cloud point, they predict the locations offset to the corners of the box following the above loss. This objective is used in this project.

### 3.2. 3D Object Detection Pipeline

The 3D object detection pipeline is based on the PointFusion [15], which performs the detection process using images from cameras and point cloud representations provided by lidar sensors. The framework is visualized in Figure 2. First, we propose regions of interest using a 2D object detector on an image representation of the environment. For the respective regions, we extract cropped images and point cloud points within the region. The extracted representations are then independently encoded with the point cloud encoder, and image encoder networks. Both representations are concatenated and passed through a fusion network that outputs 3D box representations. As part of this project, we are re-implementing the point cloud encoder PointNet and fusion network, and using a pre-trained 2D object detector and image encoder. Each of the sub-modules is described below.

**2D Object Detector**: The authors of [15] used off-the-shelf Faster-RCNN [10] implementation. We are using an off-the-shelf Mask R-CNN [3]. It is a newer and readily available architecture. The implementation and trained

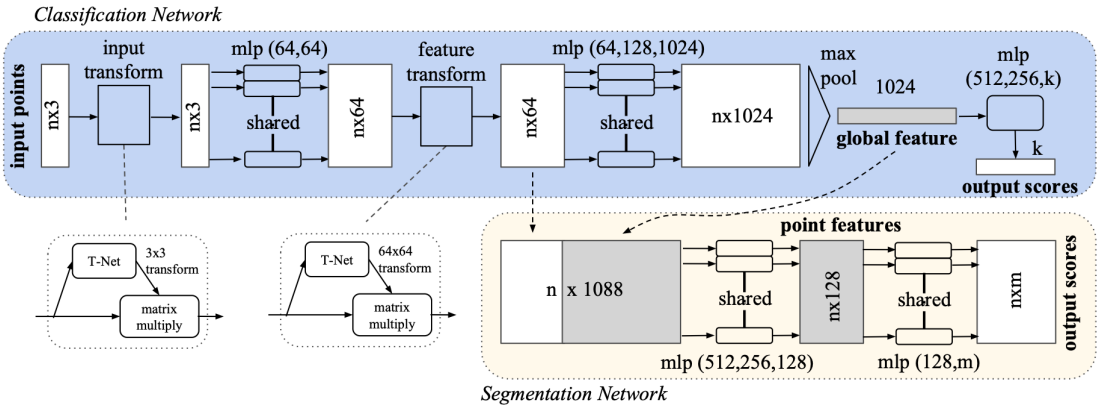


Figure 3. PointNet pointcloud encoder pipeline. Image credits: PointNet [8]

weights are provided by the PyTorch team and available at PyTorch Mask RCNN. More information about the architecture can be found in [3].

**Point Cloud Representation Encoder:** PointNet is an efficient approach to encode lidar point cloud without using 3D voxel grids or series of images [8]. It is visualized in Figure 3. It uses a symmetric function to satisfy the permutation invariance property of the point cloud set. Hence, its input is in $\mathbb{R}^{n\times3\times3}$, where n is the number of points in the cropped scene. It outputs a point-wise feature in $\mathbb{R}^{n\times64}$ and a single global feature in $\mathbb{R}^{1\times1024}$. It is re-implemented as part of this project. As a reference, we are comparing it with a popular in a community implementation pointnet.pytorch [14].

**Image Encoder:** I am using a ResNet-50 pre-trained on the ImageNet task [2]. It is a popular architecture that takes in image in $\mathbb{R}^{3\times H\times W}$. As output, I take a final residual block averaged across the feature map in $\mathbb{R}^{1\times2048}$, as described in PointNet paper [8]. We are using an implemented and pre-trained network by PyTorch team.

**Fusion Network:** It stacks the encodings provided by image and point cloud encoders, provides them to a multi-layer perceptron, and outputs a 3D bounding box for each object [15]. There are two types of fusion networks:

- **Global Fusion Network:** It consists of 4 fully connected layers with ReLU activations between each layer and outputs bounding box corners. Setup is visualized in Figure 2D.

- **Dense Fusion Network:** Network architecture is identical to global fusion network. Instead of predicting the bounding box corners, it predicts an offset of each 3D point to the 3D bounding box corners. Hence, the network is "agnostic to the spatial extent of a scene" [15]. To make it work, we need an additional predicted score that determines how important each 3D point is. Setup is visualized in Figure 2C. We also optimize a modified loss:

$$L_{3D} = \frac{1}{N}\sum_i (L^i_{offset}\cdot c_i - w\cdot log(c_i) + L_{stn}$$

where $c_i$ is a predicted score, and $w$ is a weight factor which is set to 0.1.

**Extensions:** We also consider extensions, such as are the incorporation of road layouts, and adversarial training. Road layout information is a useful prior for human drivers when searching for potential oncoming vehicles. Adding map information of the environment could potentially boost the performance of the detector. The addition of adversarial loss based on the predicted 3D bounding box, image, and map information might help us reject unrealistic predictions (e.g. vehicles printed on walls/banners or out-of-scale predictions). We add a discriminator, which takes in an image, road layout, and bounding box. Its task is to classify whether the bounding boxes are ground truth or our predicted ones.

$$L = L_{3D} + L_{DIS}$$

## 4. Experiments

This is an educational project, where we want to get a respectful performance on a 3D object detection task rather than match SOTA results. We are comparing dense fusion, global fusion, and dense fusion + adversarial training. Our implementation is public and available at Pointfusion.pytorch. There are still many issues to address in this package, and we will be improving this repository in the future.

### 4.1. Dataset

We use a Nuscenes dataset which consists of 1000 20s-long scenes annotated at 2Hz. It contains information from

Figure 4. Two samples from the Nuscenes dataset. Top 3 rows: different camera views around the vehicle. Bottom row: Radar detection (left) and lidar point cloud projected onto 2D plane (right). Lidar and radar are visualized on top of the road layout information.

Figure 5. Predicted bounding boxes from PointFusion with Dense Fusion network on the left and ground truth on the right. [15].

the following sensor modalities: 6 cameras, 1 lidar, 5 radar, GPS, and IMU, and road layout information. Example samples are visualized in Figure 4. The total size of the dataset is close to 500 GB and careful pre-processing is required. [1] provides a high quality package called nuscenes-devkit. However, its memory requirements and overall slow responsiveness prohibit its use during training. Hence, instead, we are saving each dataset sample required for a single forward pass in a hickle file. A sample is then loaded by the dataloader in an online fashion leading to more than 100x speed up. This approach, however, requires in the worst-case scenario 1TB of storage ($2 \times 500$ GB) during pre-processing. Hence, the experiments are conducted on the $30\%$ of the entire dataset due to storage constraints.

### 4.2. Metrics

To evaluate the performance of our approach, we are using 3D object detection Average Precision (AP).

### 4.3. Training Setup

We are using PyTorch 1.10.2 [7]. The models are trained on the research workstation with AMD Ryzen 3960X, Nvidia Titan RTX 24GB GPU, and 64 GB RAM. I am not aiming to replicate the SOTA results, but I would like to achieve reasonable performance and evaluate whether any extensions to the framework improve the performance or accelerates convergence. The train-val-test dataset splits are 0.8-0.1-0.1, respectively. We use Adam optimizer with a learning rate set to 0.001 and the batch size is 512. We optimize the model over 100 epochs with 50 batch samples per epoch, i.e. $50 \times 512$ samples per epoch. We save a model every 5 epochs and use early stopping based on the validation set. The training loss plots for each model are visualized in Figure 6. Unfortunately, the adversarial training, due to unknown reasons, plateaus very early leading to poor detection performance. None of the models have over-fitted.

### 4.4. Qualitative Results

The example results of our 3D object detection pipeline are visualized in Figure 5. We can see that our models perform well when the object is unoccluded and close to the ego vehicle. It struggles when the vehicle is far away, occluded, or present at the edge of the camera plane.

### 4.5. Quantitative Results

We compare our model in terms of the Average Precision metric evaluated on our test set in Table 1. We also compare models to the reported values of the baselines in the nuscenes paper [1]. Results of PointPillars and Orthographic Feature Transform are evaluated on the different division of the nuscenes dataset. It is provided just as a reference.

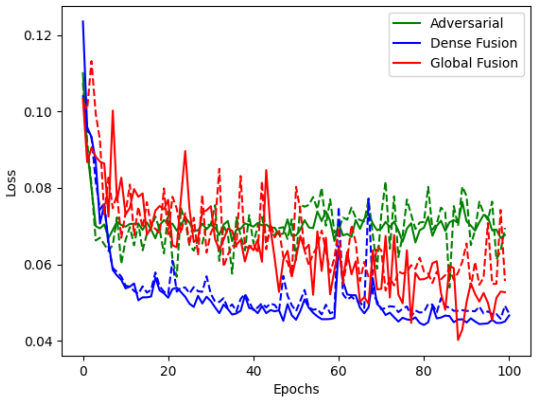


Figure 6. Plot of training (solid) and validation losses (dashed).

Table 1. Quantitative Results. Results of PointPillars and Orthographic Feature Transform are evaluated on the different division of the nuscenes dataset.

| Models | AP |
|---|---|
| Dense Fusion [15] | 8% |
| Global Fusion [15] | 7% |
| Adversarial Approach | 2% |
| PointPillars [5] | 20% |
| Orthographic Feature Transform | 13% |

## 5. Conclusions

As part of this project, we have implemented PointFusion and achieved satisfactory performance on the Nuscenes Dataset. The results suggest that Dense Fusion set outperformed Global Fusion. We are capable of detecting objects which are present clearly in the scene and close to the ego vehicle. As expected, we have not been able to replicate the SOTA performance. However, I believe this implementation is accurate and in the future, we will be able to get closer to the advertised performance. We have also introduced adversarial training. I believe, that adversarial training is the way to improve 3d object detection even further. Though, I am not sure if the approach we have recommended is the way to go.

There are several key weaknesses of this approach which we would like to improve in the future. One of the key weaknesses of the PointNet approach is the lack of inclusion of point cloud information when identifying the regions of interest provided to the image and point cloud encoders. I believe the incorporation of the 3D information at this stage could reduce the number of false positives (e.g. vehicle printed on the wall) and potentially make the approach more robust to unknown or rare objects appearing on the road. Another weakness is in how we are identifying the regions of interest. We are detecting the objects on the camera images independently, not stitched together. Hence,

when the vehicle is in-between frames, it is not detected correctly.

## References


[1] 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. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[3] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[7] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[8] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[10] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[11] S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 770–779, 2019.

[12] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[13] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[14] F. Xia. Pointnet.pytorch, https://github.com/fxia22/ pointnet.pytorch.

[15] D. Xu, D. Anguelov, and A. Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 244–253, 2018.

[16] Y. Yan, Y. Mao, and B. Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.