# Robust 3D Object Detection for Autonomous Vehicles using Sensor Fusion
## *Final Report*

Mohan Rao Divate Kodandarama
University of Wisconsin-Madison
Madison, WI
divatekodand@wisc.edu

Aditya Rungta
University of Wisconsin-Madison
Madison, WI
arungta@wisc.edu

## Abstract

*Accurate detection of objects in 3D is a central problem in autonomous navigation. Although, there are some works in this area, most of them suffer from one of these problems - poor accuracy at far off range and poor accuracy on minority classes. In this work, we try to address some of these by fusing multi-sensor (Image and LiDAR) information. More formally, we augment the true point cloud data obtained form a lidar sensor, with pseudo point-cloud data generated from a combination of corresponding monocular image and range view of the true lidar data.*

## 1. Problem Definition & Motivation

3D object detection is a fundamental problem in the domain of self-driving cars. It's very important to have an accurate solution to this problem for the future of self-driving cars and hence, extensive research is going on in this area in order to be as close to perfection as possible. We have tried to make some improvements upon some of the existing works.

In our work, we intend to explore the viability of augmenting the point cloud with pseudo-LiDAR data generated form monocular images. More formally, the problem is to predict accurate depth map from monocular images and project the points in 3d to augment the point cloud.

[13] utilizes a similar approch to augment the point cloud with Pseudo-LiDAR points. However, they make use of stereo images to generate Pseudo-LiDAR data.

## 2. Related Work

[9] is one of the first papers that proposed an architecture for processing point cloud data. It processes each point in the point cloud independently and uses a symmetric operation like max or sum to derive a global descriptor of the point cloud. [10] improves [9] by utilizing hierarchical features and weight sharing (inspired by CNN architecture). [14] further improves [9] by modelling the relationship between the points in a point cloud using graph convolutions. Our approach uses [12] which internally uses [10] to compute pointcloud features.

[16] is one of the first papers that proposed an end-to-end learnable system for 3D object detections using point cloud data. It divides the 3D world into voxels, computes voxel-wise features and employs 3D convolutional network as a detection head .However, expensive 3D convolution operations resulted in very high inference latency. [6] refines the architecture of [16] by eliminating 3D convolution operations and utilizing better loss function [7].

[12] takes a two stage approach and is inspired by the highly successful RCNN framework for 2D object detection. The first stage of the network, segments the foreground and background points and also output per point bounding box. The second stage filters the overlapping proposal by using region based pooling and then refining the proposal in canonical coordinates. We use [12] as a detection head in our end-to-end architecture. We consider this because of its low latency and high performance.

[8] uses LiDAR data as range view images. It uses a fully convolutional network to predict a multimodal distribution over 3D boxes for each point and then it uses a clustering algorithm to fuse the predicted distributions.

## 3. Approach

Our fusion based approach for 3D Object Detection is outlined below.

1. **Estimate per pixel depth from monocular images(Depth map prediction) -** Our depth prediction network is based on [4]. However, as shown in Figure 1 our architecture utilizes both input monocular image and point cloud data. The encoder network consists of two resnet18 based feature pyramid encoders one for encoding Image data and the other for encoding lidar data. We project all the lidar points on to the image plane to obtain a range view representation of the lidar

data that can be fed into the encoder network. Image features and LiDAR features are fused using simple elementwise addition operation.

The decoder is a dual of the each branch of the encode and predicts per pixel depth. Similar to [11], we use skip connections between the layer of encoder and decoder.

Since it is very difficult to obtain ground truth data for depth of every pixel, we formulate the problem as a self-supervised learning problem where we minimize the photometric re-projection error between two images captured at adjacent time frames at training time.

Figure 2 shows an example output depthmap generated using our augmented model.

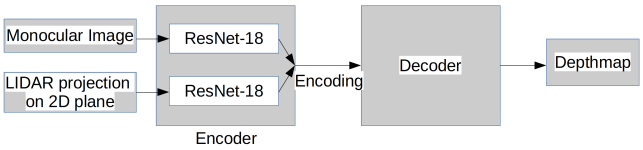



Figure 1. Architecture of our model for predicting the depthmap

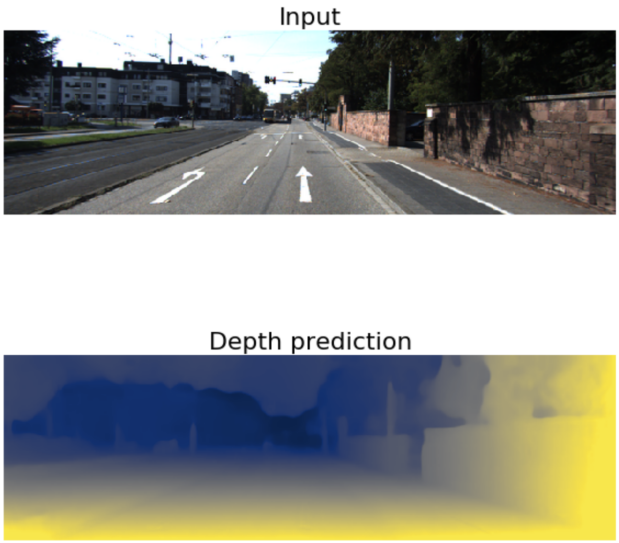



Figure 2. Estimated disparity map for an image from the KITTI dataset using our augmented model

2. **Project all images pixel to 3D space (LiDAR Coordinates)-** Once the per pixel depth estimate is available, all the images pixels can be projected into 3D space using the camera intrinsic matrix $K$ (available as a part of calibration data in [3]). More formally, the image pixel at location $(i, j)$ with the estimated depth $D^{i,j}$ can be projected into 3D according to equation 1.

$$Q^{i,j} = D^{i,j} K^{-1} [i, j, 1]^T \quad (1)$$

3. **Alignment and Filtering -** Unlike stereo, Pseudo-LiDAR generated by monocular images would not be perfectly aligned with the LiDAR (This is evident in Figure 5). Further, the predicted depth map would have to be scaled to match the scale of the true point cloud. We find the scaling and scale the predicted depth map as follows -

   (a) Find the average of absolute depth of all the points in the original point cloud.

   (b) Find the average of absolute depth of all the points in the predicted point cloud.

   (c) Scaling factor would be the ratio of average depth of original point cloud to the average depth of the predicted point cloud (pseudo point cloud).

   After scaling the pseudo point cloud, we filter out the points in the pseudo point cloud that are very far from the original point cloud. This is done to prevent the corruption of the original point cloud by the pseudo points. More formally our algorithm is described below -

   (a) Insert all points from the original point cloud in a KD-Tree.

   (b) Iterate through all the predicted pseudo points and filter out points that are away from the original point cloud by a predefined threshold.

   The filtering operation can be made fast by using a multi-threaded implementation of KD-Tree and processing pseudo points in parallel.

4. **3D Object Detection -** Once the augmented point cloud is available, use [12] for detecting 3D objects.

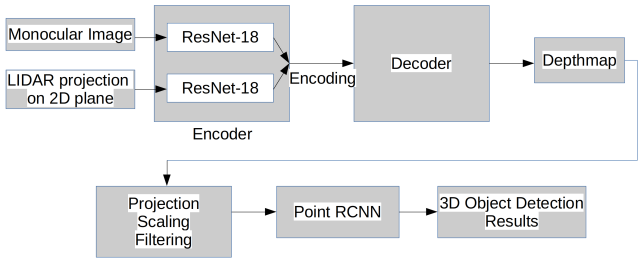



Figure 3. Our end to end architecture taking the monocular images and lidar projection on 2D plane as input and finally generating bounding boxes for 3D object detection

### 3.1. Datasets

We are using the KITTI dataset for all our experiments.

1. For training the depth prediction network we used the the KITTI RAW dataset.

2. For final 3D object detection results, we use KITTI 3D Object detection benchmark dataset

## 4. Experiments and Results

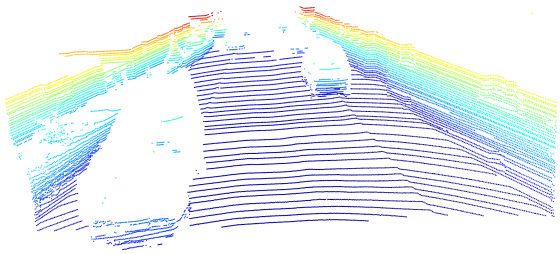
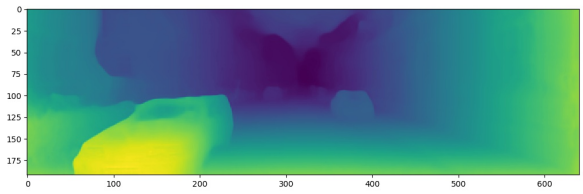


Figure 4. LIDAR point cloud corresponding to the above image



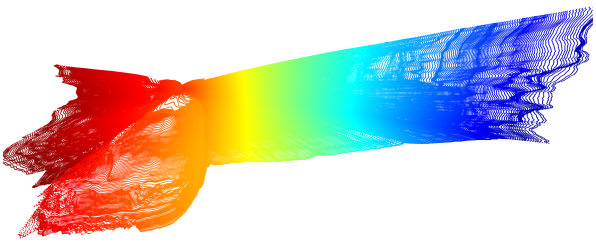



Figure 5. The disparity image above is obtained by inputting the monocular image into our depthmap prediction model.This disparity image is then processed to get the depthmap. Pseudo-LiDAR is finally generated by projecting the RGBD (Estimated Depth) in 3D.

Our model consists of an encoding module and a decoding module. The encoding module is comprised of two resnet-18 models. One of these resnet-18 models is responsilble for creating an encoding of the monocular image and the other resnet-18 model is responsible for encoding the lidar projected on a 2D space. We used the pretrained resnet-18 encoder for the monocular images and froze its layers while training. Only the encoder for lidar projection is trained. The output encodings are then combined by adding and fed into the decoder. Our decoder consists of 6 upsampling layers such that it reconstructs a depthmap with the same dimension as the input. The 3D object detection model that we have used is Point RCNN which is pretrained.

We used the KITTI dataset for all our experiments. For training our depthmap architecture, we made use of the KITTI raw dataset. For getting the results for 3D object detection, we used the KITTI 3D Object detection benchmark dataset.

We have made use of the Adam's optimizer in our training. Our depthmap architecture is trained for 20 epochs.

Figure 6. Depthmap prediction using our depthmap architecture for 2 images from the KITTI dataset.

Figure 6 show the estimated depth for images from KITTI RAW eigen split. We also have some quantitative results for depthmap prediction and we have comapred them with the results quoted in [4] and our results turn out to be better. Figure 7 shows the comparison of the depthmap prediction results. abs_rel is the relative absolute error which is basically equal to:

$$abs\_rel = \frac{abs(gt\_depth - predicted\_depth)}{gt\_depth}$$

As you can see in Figure 7, the abs_rel as well as some other errors reported using our method are significantly smaller than the ones reported in [4].

Figure 5 shows the generated Pseudo-LiDAR data. It is evident that the generated Pseudo-LiDAR data does not perfectly align with the point cloud data and requires alignment and filtering.

To evaluate our final 3D object detection results, we have focused on detecting the car category which has an IOU threshold of 0.7. We have compared the 3D object detection results obtained using our method with the results

Without fusion(monodepth):

| abs_rel | sq_rel | rmse | rmse_log |
|---|---|---|---|
| 0.115 | 0.905 | 4.863 | 0.193 |

With fusion:

| abs_rel | sq_rel | rmse | rmse_log |
|---|---|---|---|
| 0.085 | 0.591 | 3.890 | 0.163 |

Figure 7. Comparison of depth map prediction results using [4] and our method. The table above is using [4] and the table below represents our results.

quoted in [12]. As you can see in Figure 8, the average precision values across all the three categories have a sizable improvement using our method as compared to the results reported in [12]. We attribute this improvement to the quality of our augmented point cloud and believe that the results can be improved further by employing more sophisticated techniques to filter the pseudo point cloud and align it with the original point cloud.

AP for Car(IoU = 0.7 using LiDAR):

| Easy | Medium | Hard |
|---|---|---|
| 88.88 | 78.63 | 77.38 |

AP for Car(IoU = 0.7 using our augmented LiDAR data ):

| Easy | Medium | Hard |
|---|---|---|
| 89.19 | 78.85 | 77.91 |

Figure 8. Comparison of 3D object detection results using [12] and our method. The table above is using [12] and the table below represents our results.

## 5. Future Work

We can incorporate the following improvements in our architecture -

1. Although, we feed in the original point cloud into the depth prediction network, our current network does not explicitly enforce the predicted depth map to be consistent with the original point cloud. We could potentially improve the depth prediction by enforcing explicit constraints on the predicted output.

2. We can eliminate the filtering stage by only projecting the pseudo points corresponding to the objects. This can be accomplished by segmenting the objects form the image.

3. We currently train the depth prediction network and the object detection network separately. We could potentially improve the performance by end-to-end training.

## 6. Code

Code for all our experiments are available at github-link. Use $depth\_prediction$ branch to reproduce our results.

## References


[1] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[2] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, June 2010.

[3] A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[4] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Digging into self-supervised monocular depth estimation. *CoRR*, abs/1806.01260, 2018.

[5] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Lake Waslander. Joint 3d proposal generation and object detection from view aggregation. *CoRR*, abs/1712.02294, 2017.

[6] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. *CoRR*, abs/1812.05784, 2018.

[7] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.

[8] Gregory P. Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K. Wellington. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. *CoRR*, abs/1903.08701, 2019.

[9] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.

[10] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413, 2017.

[11] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]).

[12] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. *CoRR*, abs/1812.04244, 2018.

[13] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q. Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. *CoRR*, abs/1812.07179, 2018.

[14] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *CoRR*, abs/1801.07829, 2018.

[15] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[16] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection, 2017.