Object Detection on Point Cloud data

Sreenithy Chandran
Arizona State University
Tempe, Arizona
schand56@asu.edu

Shenbagaraj Kannapiran
Arizona State University
Tempe, Arizona
shenbagaraj@asu.edu

Abstract

With the abundance of Light Detection And Ranging(LIDAR) sensor data and the expanding self driving car industry carrying out accurate object detection is central to improve the performance and safety of autonomous vehicles. However point cloud data is highly sparse and this sparsity varies from one LIDAR to another and there is a need to study how adaptable object detection algorithms to data from different sensors. Due to the expensive nature of high end LIDAR this work aims to explore if a a network trained using a standard dataset like KITTI could be used for inference with a point cloud data from a 2D LIDAR that is modified to work as a 3D LIDAR. Additionally we also aim to explore if the average precision varies when the input fed to the network changes. We make use of Complex YOLO architecture which is a state of the art real-time 3D object detection network to train the KITTI dataset.

1. Introduction

Point cloud processing is becoming more and more important for autonomous driving due to the strong improvement of automotive Lidar sensors in the recent years. The sensors of suppliers are capable to deliver 3D points of the surrounding environment in real-time. The advantage is a direct measurement of the distance of encompassing objects [**?**]. This allows us to develop object detection algorithms for autonomous driving that estimate the position and the heading of different objects accurately in 3D Compared to images, Lidar point clouds are sparse with a varying density distributed all over the measurement area[7, 2, 12, 3]. Those points are unordered, they interact locally and could mainly be not analyzed isolated. Point cloud processing should always be invariant to basic transformations[11].

In general, object detection and classification based on deep learning is a well known task and widely established for 2D bounding box regression on images [9, 6, 10, 1]. Research focus was mainly a tradeoff between accuracy and efficiency. In regard to automated driving, efficiency is much more important. Therefore, the best object detectors are using region proposal networks (RPN) or a similar grid based RPN-approach. Those networks are extremely efficient, accurate and even capable of running on a dedicated hardware or embedded devices. Object detections on point clouds are still rarely, but more and more important. Those applications need to be capable of predicting 3D bounding boxes. Currently, there exist mainly three different approaches using deep learning :

1. Direct point cloud processing using Multi-Layer-Perceptrons [7, 8]
2. Translation of Point-Clouds into voxels or image stacks by using Convolutional Neural Networks [2, 12, 3]
3. Combined fusion approaches [2, 5]

2. Related Work

Recently, Frustum-based Networks [7] have shown high performance on the KITTI Benchmark suite. The model is ranked1 on the second place either for 3D object detections, as for birds-eye-view detection based on cars, pedestrians and cyclists. This is the only approach, which directly deals with the point cloud using Point-Net [10] without using CNNs on Lidar data and voxel creation. However, it needs a pre-processing and therefore it has to use the camera sensor as well. Based on another CNN dealing with the calibrated camera image, it uses those detections to minimize the global point cloud to frustum-based reduced point cloud. This approach has two drawbacks: i). The models accuracy strongly depends on the camera image and its associated CNN. Hence, it is not possible to apply the approach to Lidar data only; ii). The overall pipeline has to run two deep learning approaches consecutive, which ends up in higher inference time with lower efficiency. The referenced model runs with a too low frame-rate at approximately 7fps on a NVIDIA GTX 1080i GPU [1]. In contrast, Zhou et al. [3] proposed a model that operates only on Lidar data. In regard to that, it is the best ranked model on

KITTI for 3D and birds-eyeview detections using Lidar data only. The basic idea is an end-to-end learning that operates on grid cells without using hand crafted features. Grid cell inside features are learned during training using a Pointnet approach [10]. On top builds up a CNN that predicts the 3D bounding boxes. Despite the high accuracy, the model ends up in a low inference time of 4fps on a TitanX GPU [3]. Another highly ranked approach is reported by Chen et al. [5]. The basic idea is the projection of Lidar point clouds into voxel based RGB-maps using handcrafted features, like points density, maximum height and a representative point intensity [9]. To achieve highly accurate results, they use a multi-view approach based on a Lidar birds-eye-view map, a Lidar based front-view map and a camera based front-view image. This fusion ends up in a high processing time resulting in only 4fps on a NVIDIA GTX 1080i GPU. Another drawback is the need of the secondary sensor input (camera).

3. Contributions

The contributions of the work is two fold :

1. Generate a single birds' eye view and 2D point map of point cloud data for KITTI dataset and feed that as input to the Complex YOLO networkto perform object detection on evaluate which of the two inputs perform better
2. Evaluate the possibility of using a low cost 2D LIDAR as an alternate to 3D LIDAR

4. Complex YOLO

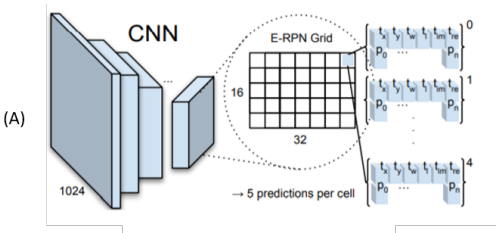
The Complex-YOLO network takes a 2D image(birds eye view or panaroma view) as input. It uses a simplified YOLOv2 [9] CNN architecture Fig 1(B), extended by a complex angle regression and E-RPN, to detect accurate multi-class oriented 3D objects.

Euler-Region-Proposal

The E-RPN parses the 3D position $b_{x,y}$ object dimensions (width $b_w$and length b_l) as well as a probability p0, class scores $p_1...p_n$ and finally its orientation b_ϕ from the incoming feature map. In order to get proper orientation the algorithm has an added complex angle $arg(|z|e^{ib})$ to it:

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_l = p_l e^{t_l}$$
$$b_\phi = arg(|z|e^{ib_\phi}) = arctan_2(t_{im}, t_{re})$$

With the help of this extension the E-RPN estimates accurate object orientations based on an imaginary and real fraction directly embedded into the network. For each grid cell



(B)

layer	filters	size	input	output
conv	24	3x3/1	1024x512x3	1024x512x24
max		2x2/2	1024x512x24	512x256x24
conv	48	3x3/1	512x256x24	512x256x48
max		2x2/2	512x256x48	256x128x48
conv	64	3x3/1	256x128x48	256x128x64
conv	32	1x1/1	256x128x64	256x128x32
conv	64	3x3/1	256x128x32	256x128x64
max		2x2/2	256x128x64	128x64x64
conv	128	3x3/1	128x64x64	128x64x128
conv	64	3x3/1	128x64x128	128x64x64
conv	128	3x3/1	128x64x64	128x64x128
max		2x2/2	128x64x128	64x32x128
conv	256	3x3/1	64x32x128	64x32x256
conv	256	1x1/1	64x32x256	64x32x256
conv	512	3x3/1	64x32x256	64x32x512
max		2x2/2	64x32x512	32x16x512
conv	512	3x3/1	32x16x512	32x16x512
conv	512	1x1/1	32x16x512	32x16x512
conv	1024	3x3/1	32x16x512	32x16x1024
conv	1024	3x3/1	32x16x1024	32x16x1024
conv	1024	3x3/1	32x16x1024	32x16x1024
route	12			
reorg		/2	64x32x256	32x16x1024
route	22 20			
conv	1024	3x3/1	32x16x2048	32x16x1024
conv	75	1x1/1	32x16x1024	32x16x75
E-RPN			**32x16x75**	

Figure 1. Complex YOLO architecture pipeline (A) The 2D image is fed into the CNN The E-RPN grid runs simultaneously on the last feature map and predicts five boxes per grid cell. Each box prediction is composed by the regression parameters t and object scores p with a general probability p0 and n class scores $p_1...p_n$. (B)The model has 18 convolutional and 5 maxpool layers, as well as 3 intermediate layers for feature reorganization respectively.

(32x16) we predict five objects including a probability score and class scores resulting in 75 features each, visualized in Fig. 1(A)

Anchor Box Design

The YOLOv2 object detector predicts five boxes per grid cell. All were initialized with beneficial priors, i.e. anchor boxes, for better convergence during training. Due to the angle regression, the degrees of freedom, i.e. the number of possible priors increased, but we did not enlarge the number of predictions for efficiency reasons. Hence, we defined only three different sizes and two angle directions as priors, based on the distribution of boxes within the KITTI dataset: i) vehicle size (heading up); ii) vehicle size (heading down); iii) cyclist size (heading up); iv) cyclist size (heading down); v) pedestrian size (heading left).

Complex Angle Regression

The orientation angle for each object b_ϕ can be computed from the responsible regression parameters t_{im} and t_{re}, which correspond to the phase of a complex number. The angle is given simply by using arctan2(t_{re}, t_{im}). On one hand, this avoids singularities, on the other hand this results in a closed mathematical space, which consequently has an advantageous impact on generalization of the model. We can link our regression parameters directly into the loss

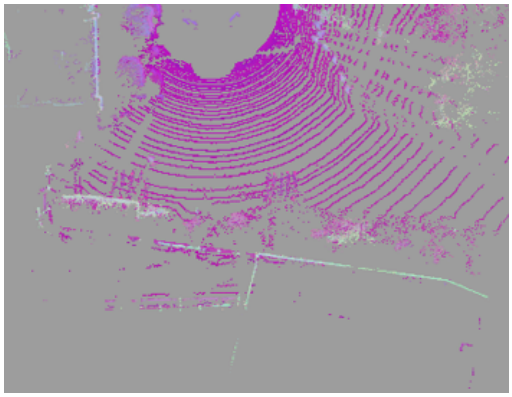


Figure 2. Birds eye view Map

function. The loss function L is based on the the concepts from YOLO and YOLOv2, who defined L_{Yolo}as the sum of squared errors using the introduced multi-part loss.

5. Point Cloud Preprocessing

The 3D point cloud of a single frame, acquired by Velodyne HDL64 laser scanner [4], is converted into a single birds-eye-view map, covering an area of 80m x 40m Fig. 6 directly in front of the origin of the sensor and panaroma view obtained using the following hardware related specifications field of view (Vertical) $+2\,\mathrm{deg}\,to\,-\,24.9\,\mathrm{deg}$, angular resolution: $0.4\,\mathrm{deg}$, field of view(Horizontal) $360\,\mathrm{deg}$,Angular Resolution(Horizontal) $0.08\,\mathrm{deg}\,-0.35\,\mathrm{deg}$.(5Hz - 20Hz) The panoroma image is given in Fig 3

The goal of this work is to evaluate whether a birds eye view of a panaroma view would serve to be a better representation of the sparse point cloud data to get a better object detection result. From the KITTI dataset [4] we obtain four different files 4 different types of files from the KITTI 3D Objection Detection dataset as follows are used in the article.

1. camera_2 image (.png)
2. camera_2 label (.txt)
3. calibration (.txt)
4. velodyne point cloud (.bin)

The image files are regular png file and can be displayed by any PNG aware software. The label files contains the bounding box for objects in 2D and 3D in text. Each row of the file is one object and contains 15 values , including the tag (e.g. Car, Pedestrian, Cyclist). The 2D bounding boxes are in terms of pixels in the camera image . The 3D bounding boxes are in 2 co-ordinates. The size (height, weight, and length) are in the object co-ordinate , and the center on the bounding box is in the camera co-ordinate.

The point cloud file contains the location of a point and its reflectance in the lidar co-ordinate. The calibration file contains the values of 6 matrices

1. P03: The Px matrices project a point in the rectified referenced camera coordinate to the camera_x image
2. R0_rect: R0_rect is the rectifying rotation for reference coordinate (rectification makes images of multiple cameras lie on the same plan)
3. Tr_velo_to_cam : maps a point in point cloud coordinate to reference co-ordinate
4. Tr_imu_to_velo: maps a point in camera coordinate to velodyne co-ordinate

The different coordinate systems used in this evaluation is given fig below. Using this datset we obtained the birds eye view and the panaroma view of the bin file. When obtaining the birds eye view our region of processing is restricted.

6. Experimental Setup

In order to obtain 3D point cloud data on any environment a low cost 2D RPlidar A1 was used. This 2D lidar can perform a 360-degree scan of the environment about a single plane. However, in order to obtain a 3D point cloud data, the lidar setup was attached to an external setup to sweep the environment. This addon setup consists of 3D printed attachments added to the 2D lidar which enables it to rotate about an axis which provides the elevation angle. The 2D lidar gives two measurable reading one being azimuth angle and other being the depth of the reflected surface for the azimuth angle measured at any instant. This is supported by a third parameter the elevation angle which is facilitated by the modified setup and it is driven by servo motor to obtain precise angles. The servo motor is connected to a teensy micro-controller which is in turn connected to raspberry pi 3 through a USB port. The reason for using teensy over raspberry pi to control the servo lies on the fact that raspberry pi does not possess any analog GPIO pins making the control of servo units through the available digital pins which uses a PWM based system to drive the system making it less accurate and prone to jitters in the servo system comprising the overall efficiency and accuracy of the system. In order to overcome this issue teensy microcontroller was used which has good analog pins to drive the servo system. The servo system was driven at 1-degree steps from a range of -30 to 70 degrees to obtain only the region of interest (done similar to velodyne lidar). Finally, the data obtained is sent to the raspberry pi through usb port which in turn is send to the host pc using ad-hoc network. This enables the user

Figure 3. Panaroma Map

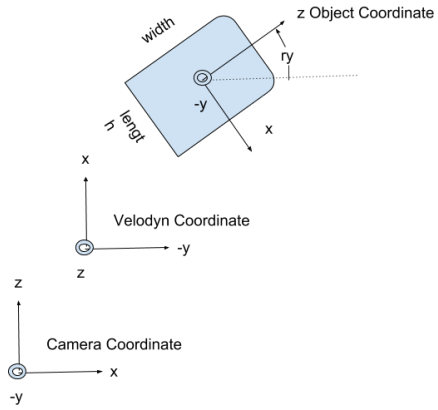


Figure 4. The different coordinate systems used in the evaluation

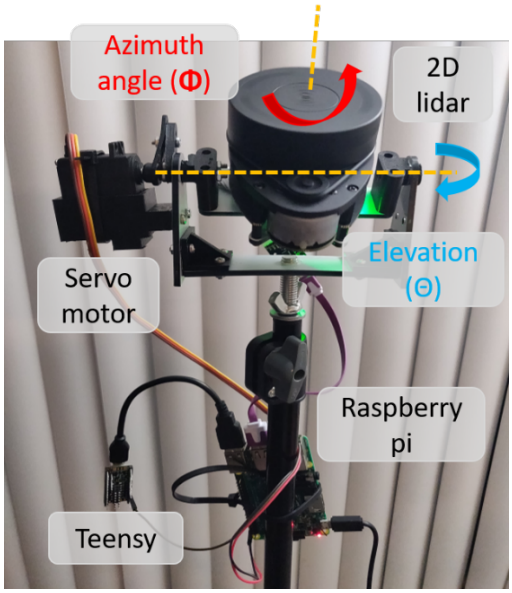


Figure 5. 2D LIDAR modified to collect 3D point cloud data

to wirelessly control the servo system. However, it is important to note that this 2D modified 3D point cloud lidar does not support real-time 3D point cloud data collection and works only on static scenarios.

6.1. Data Collection

The data obtained consists of elevation angle from the servo, azimuth angle and depth value from the lidar which is the converted to XYZ co-ordinate system using the equation below and explained in fig

$$X = \rho sin\theta cos\phi$$

$$Y = \rho sin\theta sin\phi$$

$$Z = \rho cos\theta$$

This XYZ point cloud data is plotted using scatter plot in MATLAB as showin in figure 8 . In order to get more visual sense of the depth RGB map is also added to the point cloud as shown in figure 7. The scatter obtained is sent to the Complex YOLO neural network to perform object detection. Moreover it is worthy note that the output obtained from this modified 2D lidar setup produces more scaled images as they have less range (6m) as compared to velodyne lidar data which has a range of 120m.

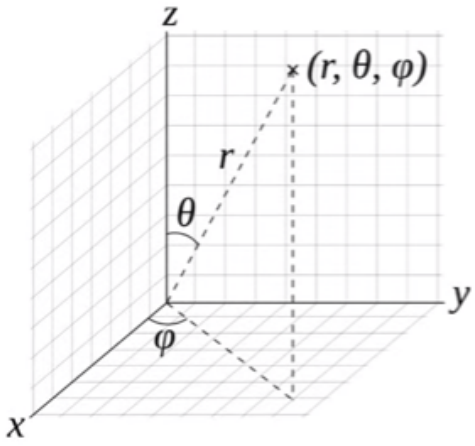


Figure 6. Representation of the point cloud data in terms of the elevation angle, azimuth angle and depth value

7. Results

We evaluated Complex-YOLO on the challenging KITTI object detection benchmark [4], which is divided into three subcategories 2D, 3D and birds-eye-view object detection for Cars, Pedestrians and Cyclists. Each class is evaluated

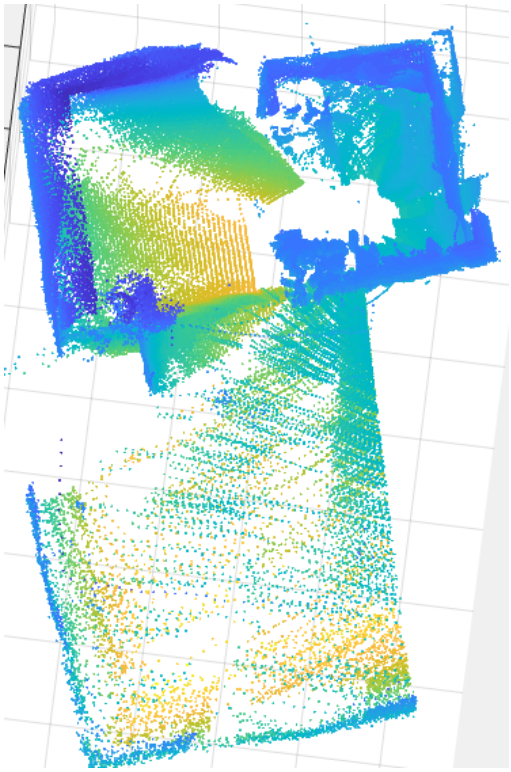


Figure 7. Sample point cloud data of an indoor environment collected from the 2D LIDAR

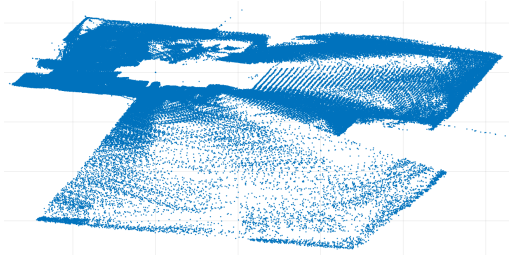


Figure 8. Birds eye view of sample point cloud data of an indoor environment collected from the 2D LIDAR

based on three difficulty levels easy, moderate and hard considering the object size, distance, occlusion and truncation. This public dataset provides 7,481 samples for training including annotated ground truth and 7,518 test samples with point clouds taken from a Velodyne laser scanner, where annotation data is private. Note that we focused on birds-eye-view and panaroma view and perform only 2D box generation.

7.1. Training Details

We trained our model from scratch via stochastic gradient descent with a weight decay of 0.0005 and momentum 0.9, and 120 epochs. First, we applied our pre-processing to generate the birds-eye-view and panaroma view from Velodyne samples. Then, we subdivided the training set with public available ground truth, but used ratios of 85% for training and 15% for validation, because we trained from scratch and aimed for a model that is capable of multi-class predictions. The class distribution with more than 75% Car, less than 4% Cyclist and less than 15% pedestrians. For the first epochs, we started with a small learning rate to ensure convergence. After some epochs, we scaled the learning rate up and continued to gradually decrease it for up to 1,000 epochs

7.2. Evaluation on KITTI

We have adapted our experimental setup and follow the official KITTI evaluation protocol, where the IoU thresholds are 0.7 for class Car and 0.5 for class Pedestrian and Cyclist. Detections that are not visible on the image plane are filtered, because the ground truth is only available for objects that also appear on the image plane of the camera recording. Note, that we ignore a small number of objects that are outside our birds-eye-view map boundaries with more than 40m to the front, to keep the input dimensions as small as possible for efficiency reasons.

Some of the results for birdeye view evaluation is given in Fig 9. The network is able to yield good results on the testing dataset and is able to predict most of the cars and fewer of the pedestrian and cyclists. Additionally it can be seen that the bounding boxes are not properly oriented in some of the images.

Few results for panaroma view is given in Fig 10. Unlike the bird eye view the panaroma view does not yield very good results. The network is able to detect most of the cars however the results for pedestrians and cyclists is lower. In panaroma view we also observe that the bounding boxes are not of the proper size as the object, unlike in the bird eye view case where the bounding box sizes were fixated and only the orientation varied.

7.3. Comparison of the two input methods

We compare the object detection performance of the different methods using Complex Yolo by using mAP. mAP is the metric to measure the accuracy of object detectors like Faster R-CNN, SSD, etc. It is the average of the maximum precisions at different recall values. For evalution of which we used the Intersection over Union given in the KITTI bechmark suite. IoU measures how much overlap between 2 regions, This measures how good is our prediction in the object detector with the ground truth (the real object boundary).IoU is 0.7 for cars, 0.5 for pedestrians and cyclists.

Other important observations to be made include the cases when the IoU values match the threshold value however is an error. This is shown in fig 11. Observe how some-

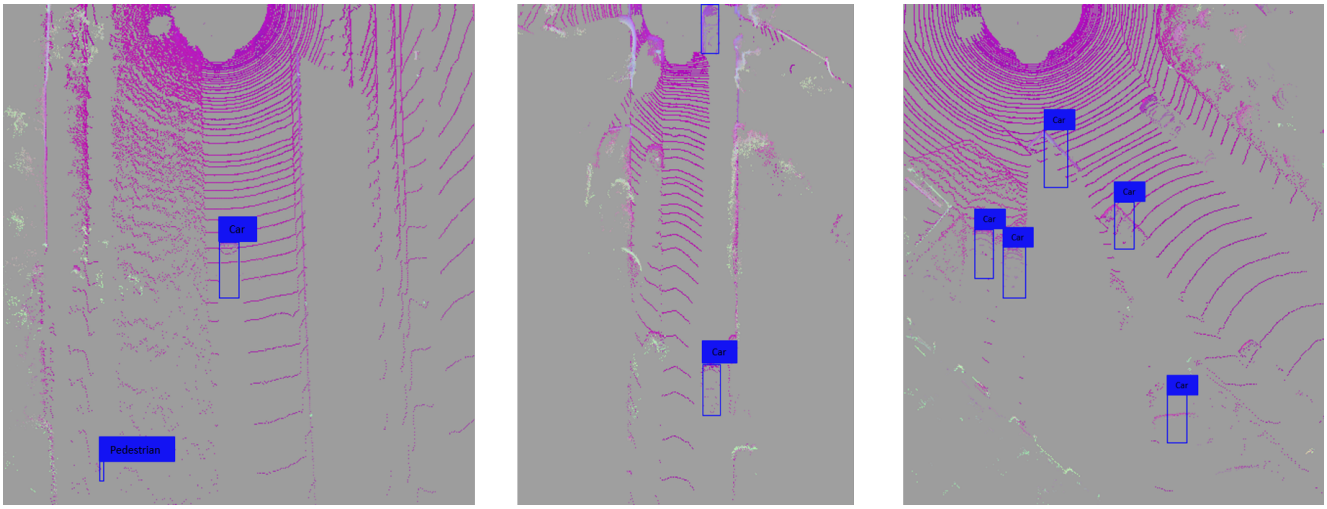


Figure 9. Results on performing object detection on the KITTI dataset when the 3D point cloud data is converted to a 2D birds eye view and fed as input to the Complex YOLO architecture. Some bounding boxes are not oriented properly over the cars

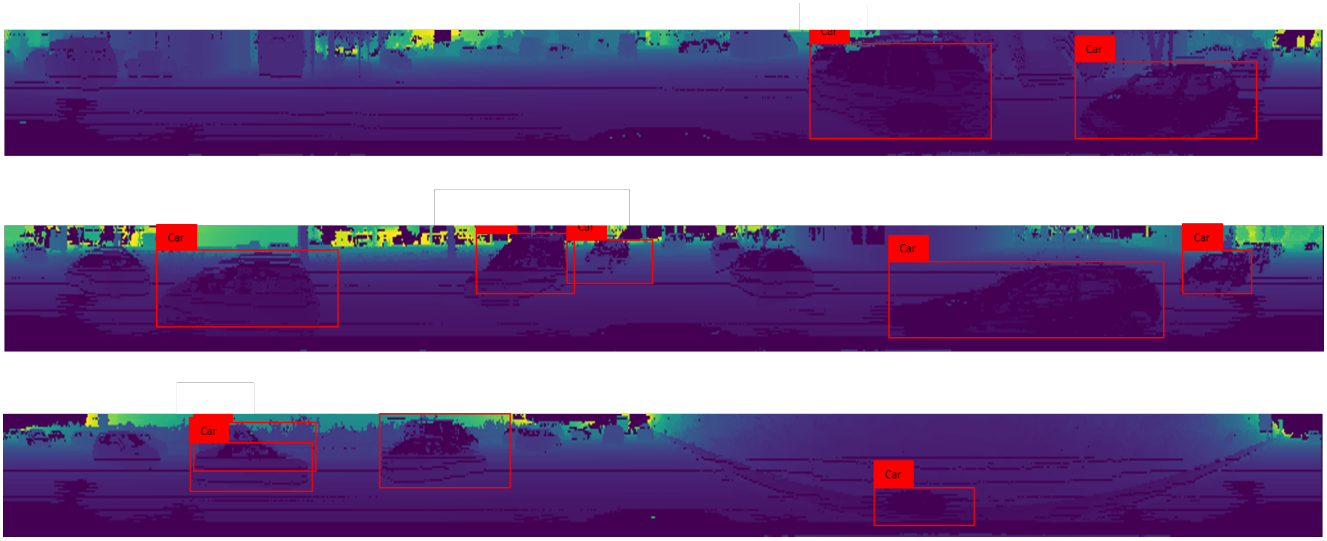


Figure 10. Results on performing object detection on the KITTI dataset when the 3D point cloud data is converted to a 2D panaroma view and fed as input to the Complex YOLO architecture. Numerouisr cars, pedestrians are left undetected by the network

Method	Car	Pedestrian	Cyclist
Birds eye view	61.21	25.92	34.84
Panaroma 40.5	-	-	

Table 1. Distributed Lighting Results on both real and simulated datasets. We compare against two baseline methods, and show that the distributed lighting based on joint optimization performs better in all cases.

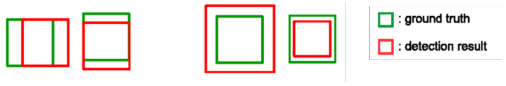


Figure 11. Various bounding box errors

times the ground truth and predicted bounding box centres are offset. However the IoU could match the threshold condition. This is an important work that has to be addressed in future works.

7.4. Evaluation on 2D LIDAR

Results on evaluation on 2D LIDAR data is shown in fig 12. The predicted box is shown on the left iamge and ground truth is shown in the right image the Complex

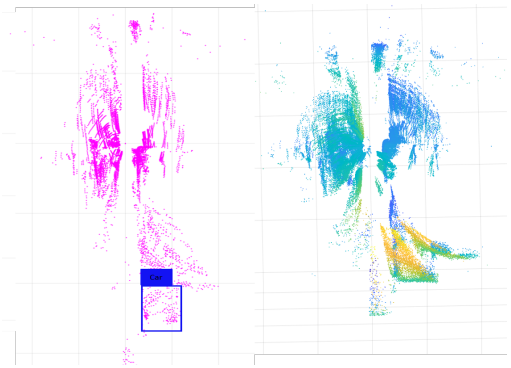


Figure 12. Results on evaluation on 2D LIDAR data, the Complex YOLO network performs very poorly when predicting objects, due to the difference in sparsity between the data obtained from the Velodyne data and RP LIDAR.

YOLO network performs very poorly when predicting objects, due to the difference in sparsity between the data obtained from the Velodyne data and RP LIDAR. Also the Sample points collecetd per second cary greatly between the two methods. It is close to 2.2 million points for an Velodyne HDL-64E and RP LIDAR collects around 500 points. Moreover the range is 6m for RP LIDAR and 120m Velodyne HDL64E.

8. Conclusion

In this paper we present the comparison on the object detection performance of Yolov2 for point cloud data by generating a single birds' eye view and 2D point map of point cloud data for KITTI dataset. This is feed as input to the Complex YOLO network to perform object detection on evaluate which of the two inputs perform better also we present the. This method does not need additional sensors, e.g. camera, like most of the leading approaches since this network makes use of the new E-RPN, an Euler regression approach for estimating orientations with the aid of the complex numbers. The closed mathematical space without singularities allows robust angle prediction. Our approach is able to detect objects of multiple classes (e.g. cars, vans, pedestrians, cyclists, trucks, tram, sitting pedestrians, misc) simultaneously in one forward path. This novelty enables deployment for real usage in self driving cars and clearly differentiates to other models.On evaluation on 2D LIDAR data the Complex YOLO network performs very poorly when predicting objects, due to the difference in sparsity between the data obtained from the Velodyne data and RP LIDAR. In future work, it is planned to add height information to the regression, enabling a real independent 3D object detection in space, and to use tempo-spatial dependencies within point cloud pre-processing for a better class distinction and improved accuracy

9. Individual Contribution

Both the authors contributed equally to the work. Shenbagaraj led the work on the modified 2D LIDAR hardware implementation and data collection. Sreenithy led the work on object detection algorithm and data collection.

References

[1] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision*, pages 354–370. Springer, 2016.

[2] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In *IEEE CVPR*, volume 1, page 3, 2017.

[3] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1355–1361. IEEE, 2017.

[4] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[5] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander. Joint 3d proposal generation and object detection from view aggregation. *arXiv preprint arXiv:1712.02294*, 2017.

[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] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. *arXiv preprint arXiv:1711.08488*, 2017.

[8] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

[9] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint*, 2017.

[10] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[11] M. Simon, S. Milz, K. Amende, and H.-M. Gross. Complex-yolo: Real-time 3d object detection on point clouds. *arXiv preprint arXiv:1803.06199*, 2018.

[12] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *arXiv preprint arXiv:1711.06396*, 2017.