Real-time 3D Object Detection from Point Clouds using an RGB-D Camera

Ya Wang[a], Shu Xu and Andreas Zell
Department of Cognitive Systems, University of Tuebingen, Sand 1, 72076 Tuebingen, Germany

Keywords: Object Detection, 2D, 3D, Real-time, RGB-D Camera, Point Clouds.

Abstract: This paper aims at real-time high-accuracy 3D object detection from point clouds for both indoor and outdoor scenes using only a single RGB-D camera. We propose a new network system that combines both 2D and 3D object detection algorithms to achieve better real-time object detection results and has faster speed by simplifying our networks on real robots. YOLOv3 is one of the state-of-the-art object detection methods based on 2D images. Frustum PointNets is a real-time method using frustum constraints to predict a 3D bounding box of an object. Combining these two approaches can be efficient for real-time 2D-3D object detection, both indoor and outdoor. We not only have the improved training and evaluation accuracy and lower mean loss on the KITTI object detection benchmark, but also achieve better average precision (AP) on 3D detection of all classes in three different levels of difficulty. In addition, we implement our system of on-board real-time 2D and 3D object detection using only an RGB-D camera on three different hardware devices.

1 INTRODUCTION

Nowadays, there are many outstanding 2D and 3D object detection approaches based on deep convolutional neural networks (CNNs), and they can be real-time with graphics processing unit (GPU) acceleration, e.g. YOLO (Redmon and Farhadi, 2018; Redmon et al., 2016; Redmon et al., 2016), Faster R-CNN (Ren et al., 2015), Mask R-CNN (He et al., 2017) and PointNets (Qi et al., 2018; Qi et al., 2017a; Qi et al., 2017b). Nevertheless, accurate and fast detection of 3D objects is a challenge and plays an important role in robotics, especially for autonomous driving. Autonomous robots are widely applied in different research fields and can be equipped with different sensors as well as powerful GPUs. RGB-D cameras are broadly used and they are much cheaper and more lightweight comparing with the Radar or Lidar sensors, meanwhile offering rich scene information like 3D colorful point clouds. Specifically, an RGB image offers us useful 2D information that can be used in state-of-the-art 2D object detection approaches while a depth map is useful for 3D information e.g. geometric position of point clouds, which give more information on object geometry.

Some researchers work on methods that combine both 2D and 3D object detection, e.g. Frustum PointNets (Qi et al., 2018), MV3D (Chen et al., 2017), PointFusion (Xu et al., 2018), and DenseFusion (Wang et al., 2019). Similar to Frustum PointNets, our method can easily and efficiently detect cars and pedestrians for outdoor scenes, which is useful for autonomous driving and street scene understanding. Detecting cars, pedestrians, and cyclists precisely and efficiently for outdoor scenes is useful for autonomous driving and robot obstacle avoidance, while person detection for both indoor and outdoor scenes is also helpful and challenging for emergency rescue and human-robot interaction.

We develop and analyze a modified Frustum PointNet network, which mainly combines 2D and 3D object detection to achieve better pointwise object detection accuracy, and has better performance on real-time robot applications. We also compare our proposed network system with the Frustum PointNets algorithms, using existing benchmarks like the KITTI object detection benchmark (Geiger et al., 2012). Considering that our system has good real-time performance and can be used for robotics, we also implemented it on a Robotnik Summit XL mobile robot, as well as a Jetson TX2 developer kit. All testing experiments were performed and compared on three different hardware devices. Furthermore, the performance of object detection in practical indoor and out-

[a] https://orcid.org/0000-0002-8726-8151

Wang, Y., Xu, S. and Zell, A.
Real-time 3D Object Detection from Point Clouds using an RGB-D Camera.
DOI: 10.5220/0008918904070414
In *Proceedings of the 9th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2020)*, pages 407-414
ISBN: 978-989-758-397-1; ISSN: 2184-4313

door scenes has also been improved.

Our main contribution in this paper is that we propose a new strategy of combining the 2D and 3D object detection approaches to achieve higher accuracy. Another contribution is that we realize 2D and 3D combined object detection with fast inference time in our mobile robots instead of only training and testing on KITTI benchmarks. Last but not least, our strategy works well both for indoor and outdoor scenes.

2 RELATED WORK

2.1 2D Object Detection

There are many popular deep CNNs object detection approaches based on 2D RGB images. 2D object detection can be divided mainly on two parts: region-based CNNs like Mask R-CNN (He et al., 2017) and Faster R-CNN (Ren et al., 2015); one-shot CNNs like YOLOv3 (Redmon and Farhadi, 2018) and SSD (Liu et al., 2016).

Faster R-CNN (Ren et al., 2015) is an end-to-end object detection method with region proposal networks (RPNs) that works in real-time. It is based on the previous region-based work of Fast R-CNN (Girshick, 2015), and the authors use RPNs to replace selective search from fast R-CNN, and later merge RPNs and fast R-CNN into a single network by sharing the convolutional features. Mask R-CNN (He et al., 2017) is based on Faster R-CNN, and extends into image segmentation by adding a mask network parallelly, therefore it can do both object detection and segmentation. However, even with a powerful GPU acceleration like an Nvidia GTX 1080Ti, it can only reach 5 frames per second (fps).

On the contrary, YOLOv3 (Redmon and Farhadi, 2018) is much faster, and it is based on the previous work of YOLOv1 (Redmon et al., 2016) and YOLOv2 (Redmon and Farhadi, 2017). YOLOv3 is a real-time object detection method, which contains Darknet-53 to extract features and feature pyramid networks (FPN) to detect small objects. Therefore, it achieves good performance for detection of objects at different scales. It can reach around 25 fps with the same type of single GPU acceleration. Its 2D bounding boxes of object detection can not only improve tracking accuracy by removing some outliers of feature matching (Wang and Zell, 2018), but also give useful semantic information that might be used for 3D object detection (Qi et al., 2018).

2.2 3D Object Detection

RGB-D cameras can give us rich information about the objects in scenes. They offer not only 2D RGB images but also depth maps and can get 3D point clouds directly using stereo infrared sensors. In (Wang and Zell, 2019a) and (Wang and Zell, 2019b), the authors built an RGB-D map of colorful point clouds using a forward-looking Microsoft Kinect v2 camera installed on a Metralabs Scitos G5 mobile robots, and they showed that colorful point clouds are useful compared with only geometric point clouds.

Most real-time state-of-the-art 3D object detection approaches are using expensive Lidar sensors such as Vote3Deep (Engelcke et al., 2017), PointR-CNN (Shi et al., 2019) and Fast PointRCNN (Chen et al., 2019). Some state-of-the-art object detection approaches based on images lack information about 3D geometry (Su et al., 2015; Kanezaki et al., 2018; Johns et al., 2016). They need multi-view images or projection methods e.g. triangulation, which will cause additional drift error in the measurement.

2.3 Combination of 2D and 3D Object Detection

There exist some state-of-the-art methods that combine both 2D and 3D object detection. MV3D (Chen et al., 2017) is a multi-view based 3D object detection network mainly for autonomous driving. The application is close to ours but we only use a single RGB-D camera without Lidar sensors. The authors combined bird-view point clouds, front-view point clouds and RGB image information, and no speed was mentioned, so we guess there might be some space for real-time autonomous driving tasks.

Frustum PointNets (Qi et al., 2018) is using a 2D detector to get a 2D box and then project into a corresponding frustum constraint to do 3D box regression. PointNet and PointNet++ (Qi et al., 2017a; Qi et al., 2017b) are used for instance segmentation inside each frustum and also used for amodal 3D box estimation. PointFusion (Xu et al., 2018) proposed a deep sensor fusion method for 3D bounding box estimation. It uses RGB images as input of ResNet to get 2D features and uses the same 3D point cloud input from a Lidar sensor before putting it into PointNet, and then gets the 3D feature. Both dense fusion and global fusion are done by comparing with the final model and the baseline model. But these above methods are all needed to use 3D point clouds from Lidar sensors, namely having a good performance within intensity but not perform well without intensity. Besides, the 2D detector part has no open-source code, and they use the ground-truth 2D detection results for training in the KITTI benchmark.

DenseFusion (Wang et al., 2019) is using only an RGB-D camera, and it is similar to PointFusion's dense fusion part, estimating 6DoF object pose by iteration. It is a pixel-wise dense fusion method that concatenates color embeddings and PointNet geometric embeddings for matching points. Different from PointNets, they use average pooling instead of max pooling to get the global features. Later they use the same way to combine local point features with the global feature. The authors also demonstrated their approach in a robot that can grasp and manipulate objects, and they announced that their method is approximately 200x faster than PoseCNN+ICP. However, iterative closest points (ICP) is very slow and few researchers would use it for real-time applications. The authors also announced that they can reach 16 fps with 5 objects in each frame for a Toyota Human Support Robot, but no hardware GPUs, CPUs, and memory information was mentioned. Therefore, it is not possible to fairly compare the speed of our method with theirs. Besides, the object classes are different and the application is also different.

Briefly speaking, the approaches of combining 2D and 3D object detection can normally bring us more precise detection results, but may slow down the system speed. Therefore, fast and accurate real-time 2D and 3D object detection is a big challenge. We want to find a novel and smart strategy for the combination of 2D and 3D object detection, and finally realize both accurate and fast detection for the indoor and outdoor scenes.

3 SYSTEM ANALYSIS

3.1 Our Network Structure

Figure 1 shows our proposed system workflow. The yellow parts highlight our main differences from Frustum PointNets (Qi et al., 2018). In detail, we only use a single RGB-D camera, namely an Intel Realsense D435, in our system to realize both 2D and 3D object detection. 2D object detection in Frustum PointNets mainly works in the region to frustum process, and later object class vector k is used. However, in our designed system, 2D information is fully used as extra channels that CNNs can learn additional features. Meanwhile, amodal 3D box estimation networks also use 2D information again.

3.2 Data Pre-processing

For benchmarks, we choose the KITTI object detection benchmark for both the training and evaluation processes. The pre-processing of having pickle files was the same as the Frustum PointNets authors did. In KITTI (Geiger et al., 2012) the RGB images were taken by a PointGrey Flea2 RGB camera and the 3D information was from a Velodyne HDL-64E 3D Lidar sensor. The calibration files were also needed for coordinates transformation and points projection.

For the real implementation in our robots, we mainly use an Intel Realsense D435 RGB-D camera. This RGB-D camera is lightweight, cheap, easy-to-use, and has official documentation which is convenient and efficient for us to do data collections for the real-world environment, both indoor and outdoor. Since we choose the Intel Realsense D435 camera, pyrealsense2 is used. There is no need to use all point clouds from the RGB-D camera because of large number points and some noise points, so we do frame-by-frame random down-sampling into 2048 points for each frustum. The principle is that we can not reduce points too much and keep the main features from original point clouds.

There are mainly three steps for data pre-processing: 2D bounding box extraction, frustum extraction, and data augmentation. The 2D box can be either directly taken from the KITTI benchmark or received by YOLOv3 2D detector from a D435 camera. Then by using the region2frustum function we can get an extracted frustum. In data augmentation, the 2D box center is randomly shifted and the width and height are randomly re-scaled from 0.9 to 1.1 for our real implementation. The type of objects is represented as one-hot vectors. Finally, the whole data is saved as pickle files and the pickle files contain the information of ID, 2D box parameters (positions and sizes), 3D box parameters (positions and sizes), point clouds positions and intensities, labels, object types, 3D heading bins (yaw angle), and frustum angles. The pickle files are mainly used for training.

3.3 2D and 3D Object Detection

One of our main contributions in this paper is that we propose a new strategy of combining 2D and 3D information into the neural networks. Frustum PointNets first use 2D object methods to get 2D bounding box, then use this information to get a 3D frustum and feed into PointNets, do instance segmentation, finally predict 3D bounding box of each object.

After several trials, we finally designed our system as shown in Fig. 1.

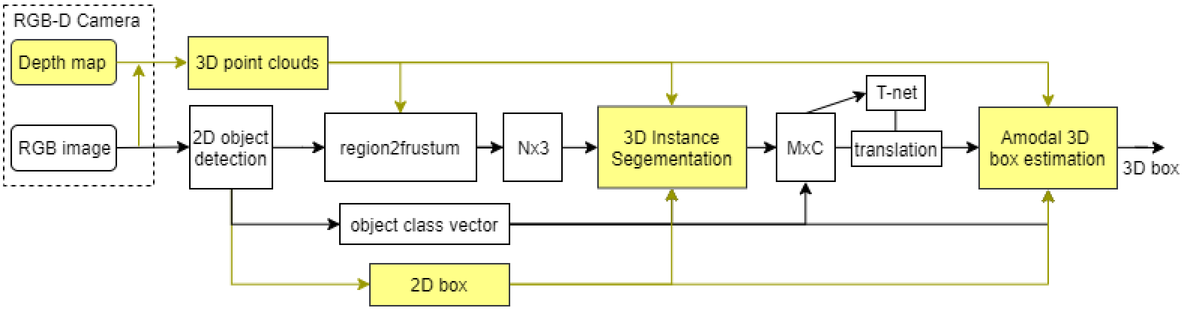


Figure 1: **OurFPN System Workflow**: We only use a single RGB-D camera to get 3D point clouds, and yellow parts highlight our main differences.

For the input, x,y,z of point clouds position as well as width and height of box information from 2D object detection offers useful prior information of objects. The problem is that we need to get 2D box information and find a smart strategy to fuse both 2D and 3D detections. The original Frustum PointNets authors use ground truth 2D box of KITTI to train their model, while we use YOLOv3 fast object detection method to get the 2D box. Even though YOLOv3 is fast, it is not as precise as Frustum PointNets authors designed 2D encoder-decoder detector and they do not open source this part code, but through their paper description they used Fast R-CNN (Girshick, 2015) and reduced VGG (Simonyan and Zisserman, 2014) base network from SSD (Liu et al., 2016), also added the feature pyramid layers (Lin et al., 2017).

Figure 2 shows our sub-system of 3D instance segmentation and the yellow parts highlight our main differences from Frustum PointNets (Qi et al., 2018). Here MLPs denote multi-layer perceptrons, and the number inside the circle means the number of convlutional layers. Each layer has batch normalization with ReLU.

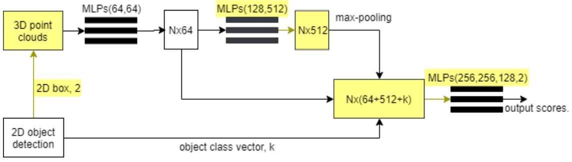


Figure 2: **OurFPN 3D Instance Segmentation**: We add not only the region2frustum output and one hot vector k as object class vector, but also 2D bounding box information. The yellow parts highlight our main differences.

Briefly, for 3D instance segmentation, we can see that 1024 feature vector is not a necessity. For 3D instance segmentation, the precision is high (above 90%) comparing with 3D box prediction, and we notice that even a decrease of the final global feature vector number from 1024 into 512 elements is still sufficient. Therefore we changed the global feature vector, which saves both our training and inference time. Besides, some convolutional layers can be safely removed with our many tests and the robustness of the whole system will not be affected, which also speeds up our whole system.

Figure 3 shows our sub-system of 3D instance segmentation. The yellow parts highlight our main differences from Frustum PointNets (Qi et al., 2018).

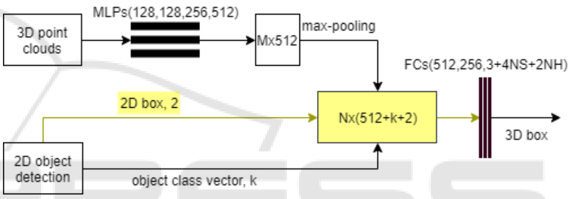


Figure 3: **OurFPN 3D Box Estimation**: We add not only the transformed output from 3D instance segmentation, but also combined 2D bounding box information again in this sub-system. The yellow parts highlight our main differences.

Here NS denotes the number of size clusters (default value 8), and NH denotes the number of heading bins (default value 12). FCs denote fully-connected layers, and the number in the circle means the number of layers.

Our system is more robust and accurate than the original Frustum PointNets system, but if we only add more information inside our system it will be slow. We have a real-time requirement, as a fast real-time system is useful for robot applications. Therefore, we need to do some speed improvements. Totally speaking, our current system does well in a trade-off between accuracy and speed, shown in section 4.

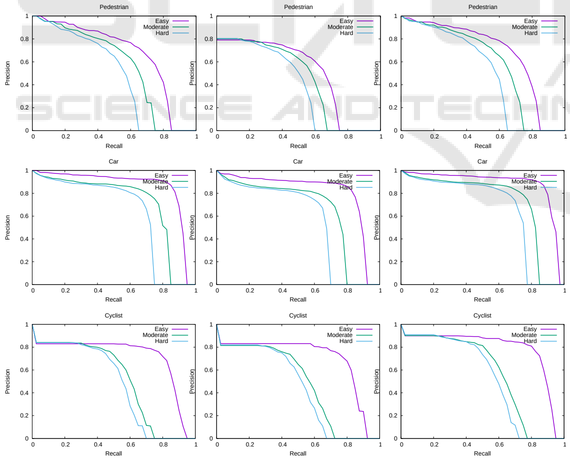
4 EXPERIMENTS

Our experiments are divided into two parts: KITTI 3D object detection benchmark and real-world robot implementation. We mainly compare our methods with two versions of Frustum PointNets (F-PNv1 and F-PNv2) (Qi et al., 2018). F-PNv1 is based on PointNet

(Qi et al., 2017a) and F-PNv2 is based on PointNet++ (Qi et al., 2017b).

4.1 KITTI 3D Object Detection Benchmark

The KITTI 3D object detection benchmark (Geiger et al., 2012) consists of 7481 training images and 7518 test images, as well as the corresponding 3D point clouds, comprising a total of 80256 labeled objects. There are three levels of difficulty (easy, moderate and hard): Easy means that the minimum bounding box height is 40 pixels, the maximum occlusion level is fully visible, and the maximum truncation is 15%; Moderate means that the minimum bounding box height is 25 pixels, the maximum occlusion level is partly occluded, and the maximum truncation is 30%; Hard means that the minimum bounding box height is 25 pixels, the maximum occlusion level is difficult to see, and the maximum truncation is up to 50%. Besides, for cars, a 3D bounding box overlap of 70% is required, while for pedestrians and cyclists only 50% is needed. The reason is that, in 3D space, cars normally have constant geometric shapes while pedestrians and cyclists are more flexible in geometry.



(a) F-PNv1 (b) F-PNv2 (c) OurFPN

Figure 4: **Quantitative results:** The Precision-Recall Curves for Pedestrian, Car, Cyclist 3D Detection. Our network (c) is better than Frustum PointNets (a) and (b) in all three different objects as well as difficulty.

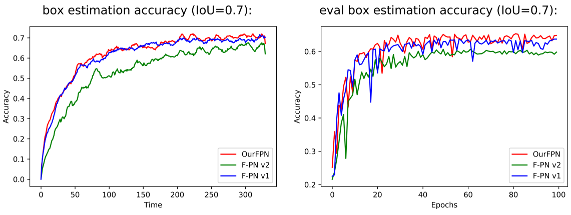
The precision-recall (PR) curve is useful for measuring if the method is good enough for all the positive and negative samples, and it can be calculated as: $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$. Here TP means true positive, FP means false positive, and FN means false negative. Figure 4 shows the PR curves for pedestrians, cars, and cyclists 3D detection in all levels of difficulty. We can see that our RGB-D modified Frustum PointNet named OurFPN (c) is always the best when comparing with the other two Frustum PointNets in (a) and (b), especially much better than Frustum PointNet v2.

Table 1: AP Comparison on KITTI 3D Detection (%).

Network	Class	Difficulty		
		Easy	*Moderate*	*Hard*
F-PNv1	Pedest.	66.49	55.71	49.23
	Car	84.00	69.88	62.74
	Cyclist	70.16	51.61	47.52
F-PNv2	Pedest.	50.60	44.48	45.66
	Car	79.09	62.16	54.24
	Cyclist	69.48	48.87	45.66
OurFPN	Pedest.	**66.84**	**57.69**	**50.95**
	Car	**85.17**	**71.86**	**64.10**
	Cyclist	**76.81**	**56.94**	**53.37**

Comparing our RGB-D modified Frustum PointNet (OurFPN) with two Frustum PointNets (F-PNv1 and F-PNv2), average precision (AP) is calculated in Table 1. We can see that our modified Frustum PointNet outperforms in all classes (Pedestrian, Car, Cyclist) and difficulty (Easy, Moderate, Hard) for 3D detection. Frustum PointNet v1 without intensity is better than v2, but still not as well as ours. Since we only use a single RGB-D camera, there is no need to compare 3D point clouds with intensity, which is normally from expensive Lidar sensors.

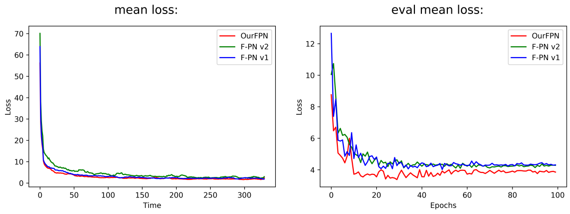
The training and evaluation accuracy comparison is shown in Fig. 5. Our modified Frustum PointNet (OurFPN) achieves the highest accuracy in both training and evaluation processes, while Frustum PointNet v2 is the worst using 3D point clouds without intensity. It is easy to explain that the original Frustum PointNet v2 uses the local regions by grouping and sampling, which heavily depends on the sparsity of points, and the intensity from Lidar sensors helps a lot.



(a) Training accuracy (b) Evaluation accuracy

Figure 5: **Quantitative results**: Training and evaluation accuracy of our modified Frustum PointNet and original Frustum PointNets (v1 and v2) on KITTI.

Besides, we also compare the mean loss during training and evaluation, shown in Fig. 6. OurFPN achieves the lowest mean loss in both training and evaluation processes especially for evaluation, while Frustum PointNets are almost the same.

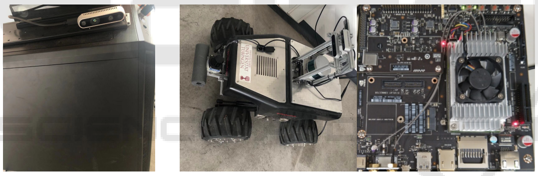


(a) Training accuracy (b) Evaluation accuracy

Figure 6: **Quantitative results**: Training and evaluation mean loss of our modified Frustum PointNet and original Frustum PointNets (v1 and v2) on KITTI.

4.2 Real-world Implementation

In order to perform real-world experiments, we used the hardware and software in Fig. 7: 1. Intel RealSense D435 RGB-D Camera; 2. GPU: Nvidia GTX1080Ti for desktop PC in Fig. 7a, Nvidia GTX1050Ti for Summit XL in Fig. 7b, and Nvidia Jetson TX2 in Fig. 7c; 3. Ubuntu 16.04 LTS with ROS kinetic; 4. Tensorflow 1.13 with Python 3.6.

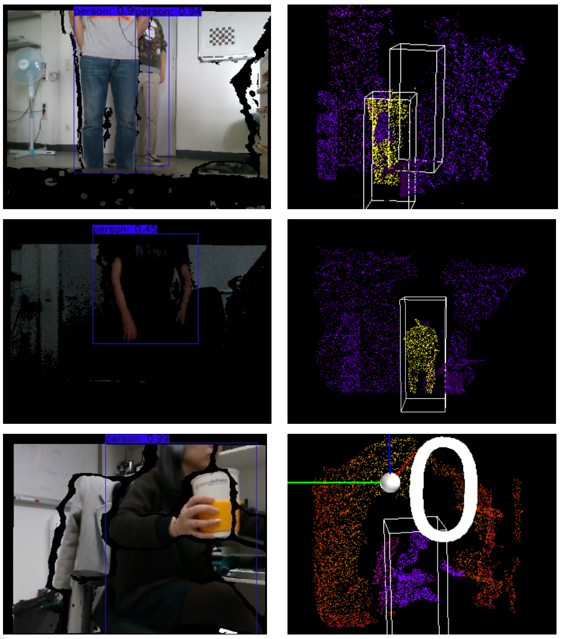


(a) (b) (c)

Figure 7: **Different Hardware Devices**: (a) 1080Ti; (b) 1050Ti; (c) Jetson TX2 developer kit.

4.2.1 Indoor Experiments

The indoor person detection experiment was done using an RGB-D camera Intel Realsense D435. We test both dark and bright scenes, both standing person and sit-down person cases, occluded or not indoor to check the robustness of our system to light. Figure 8 shows an example of 2D box visualization and 3D box visualization of our system. No matter whether the light is enough or not, we can still get good predicted 3D bounding boxes for humans. When a person is either standing or sit-down, we can also get accepted 3D boxes, and the person who is sitting down can also clearly separated from the chair. Besides, even with quite a few point clouds, our method can still detect the occluded person. The speed is 25-30 fps on our mobile robot of Fig. 7b and around 90 fps on our Desktop PC of Fig. 7a.



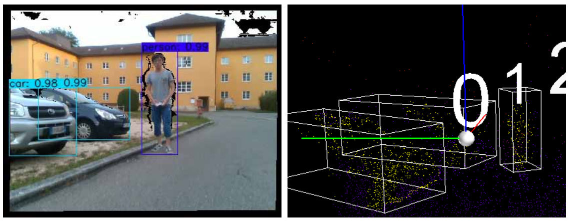
(a) 2D Box Visualization (b) 3D Box Visualization

Figure 8: **Qualitative results**: 2D and 3D predicted boxes visualization of indoor pedestrian (person) detection.

4.2.2 Outdoor Experiments

We did experiments with our Robotnik Summit XL mobile robot for the outdoor parking scene shown in Fig. 7b. Notice that here we only use an installed RGB-D sensor instead of using other sensors e.g. the Velodyne Lidar sensor on top or the Sick Laser scan in front. The speed is around 25-30 fps on our mobile robot.

Figure 9 shows an example of 2D box visualization and 3D box visualization in our system. In Fig. 9 we can see that two cars and one pedestrian can be correctly detected even for crowded occlusion case.



(a) 2D Box Visualization (b) 3D Box Visualization

Figure 9: **Qualitative results**: 2D and 3D predicted boxes visualization of outdoor car and pedestrian detection.

4.3 Speed Comparison on Different Hardware Devices

When measuring the speed of all the network systems, we ran them on three different hardware devices with Nvidia GPUs in Fig. 7. Detailed running speed comparison is shown in Table 2. Here MSG means multi-scale grouping and SSG denotes single scale grouping (Qi et al., 2017b). From Table 2, we can see that our RGB-D modified Frustum PointNet is much faster than Frustum PointNet v2, and almost the same as Frustum PointNet v1 on different GPU devices. However, ours has better accuracy than Frustum PointNet v1. Frustum PointNet v2 MSG and SSG are both slow even in our desktop PC equipped with a single Nvidia GTX 1080Ti GPU, therefore we did not transfer our system on mobile robots and the Jetson TX2 developer kit equipped with less powerful GPUs.

Table 2: Running speed comparison in frames per second (fps) on different Nvidia GPUs.

Methods	***1080Ti***	***1050Ti***	***Jetson TX2***
F-PNv1	70-100	20-30	5-15
F-PNv2 SSG	50-55	--	--
F-PNv2 MSG	15-20	--	--
OurFPN	80-105	25-30	10-15

5 CONCLUSIONS

In this paper, we designed a new network system that combines both useful features of 2D and 3D for challenging object detection tasks. The original Frustum PointNet was only trained and tested on the KITTI benchmark and directly used 2D object detection results. The 3D PointNet sub-system did not re-use 2D information in their network. Our simplified network system (OurFPN) has higher accuracy and faster speed than most current state-of-the-art 3D object detection networks on three different devices, as well as on the KITTI benchmark.

In the near future, we will optimize OurFPN to realize higher precision and faster speed. We also intend to use Lidar sensors as addition since the efficient range of RGB-D cameras is limited, which is not sufficient for outdoor street scene understanding of autonomous driving cars. Actually, we notice that Velodyne can get quite far away points like 30m but quite sparse and not well for close distance points like 3m. On the contrary, RGB-D cameras can work quite well for close objects while can not detect far-away points. We believe that combining these two types of sensor information together will be an efficient way for better object detection results. Besides, we will use the outcome 3D bounding box information of object detection to help improve SLAM system for UAVs, while the existing object-based SLAM system only uses 2D image information.

REFERENCES

Chen, X., Ma, H., Wan, J., Li, B., and Xia, T. (2017). Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1907–1915. IEEE.

Chen, Y., Liu, S., Shen, X., and Jia, J. (2019). Fast point r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 9775–9784. IEEE.

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

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361. IEEE.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448. IEEE.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969. IEEE.

Johns, E., Leutenegger, S., and Davison, A. J. (2016). Pairwise decomposition of image sequences for active multi-view recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3813–3822. IEEE.

Kanezaki, A., Matsushita, Y., and Nishida, Y. (2018). Rotationnet: joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5010–5019. IEEE.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2117–2125. IEEE.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, pages 21–37. Springer.

Qi, C. R., Liu, W., Wu, C., Su, H., and Guibas, L. J. (2018). Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 918–927. IEEE.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). Pointnet: deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660. IEEE.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). Pointnet++: deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NeurPIS)*, pages 5099–5108.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788. IEEE.

Redmon, J. and Farhadi, A. (2017). Yolo9000: better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7263–7271. IEEE.

Redmon, J. and Farhadi, A. (2018). Yolov3: an incremental improvement. *arXiv preprint arXiv:1804.02767*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NeurPIS)*, pages 91–99.

Shi, S., Wang, X., and Li, H. (2019). Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–779. IEEE.

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

Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 945–953. IEEE.

Wang, C., Xu, D., Zhu, Y., Martín-Martín, R., Lu, C., Fei-Fei, L., and Savarese, S. (2019). Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3343–3352. IEEE.

Wang, Y. and Zell, A. (2018). Improving feature-based visual slam by semantics. In *IEEE International Conference on Image Processing, Applications and Systems (IPAS)*, pages 7–12. IEEE.

Wang, Y. and Zell, A. (2019a). Semantic scene segmentation of unordered point clouds on autonomous robots. In *European Conference on Mobile Robot (ECMR)*, pages 1–6. IEEE.

Wang, Y. and Zell, A. (2019b). Semantic segmentation networks of 3d point clouds for rgb-d indoor scenes. In *International Conference on Machine Vision (ICMV)*. SPIE.

Xu, D., Anguelov, D., and Jain, A. (2018). Pointfusion: deep sensor fusion for 3d bounding box estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 244–253. IEEE.