Flash Lidar Object Detection and Tracking

Deep Samal
Evan T Gebhardt
Marilyn Wolf
ksamal3@gatech.edu
egebhardt6@gatech.edu
wolf@ece.gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, United States

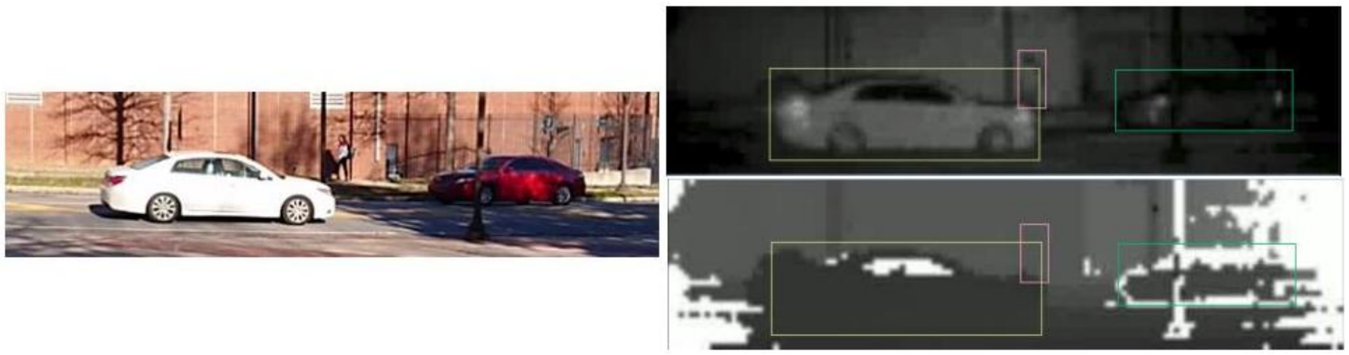

Figure 1: RGB, flash lidar intensity, and flash lidar depth images

ABSTRACT

In this paper we introduce a new dataset for object detection and tracking using Flash lidar. We highlight the challenges faced in extending current lidar based object detection algorithms to flash lidar. Currently lidar data are processed as 3D point clouds, this is because popular lidars such as Velodyne HDL-64 are scanning lidars with field of view(FoV) of 360 degrees and very fine angular resolution. Unfortunately this is not true for flash lidars which have much lower FoV and angular resolution, thus 3D point cloud may not be the most optimum method to analyze flash lidar data. Here we show that processing flash lidar data in the form of intensity and depth map gives us better than expected performance even with small amount of training data. Performance analysis also indicates accuracy of our method will increase with available data. Finally we present a real-time lidar object detector, which is based on YOLO framework, and SORT based object tracker.

CCS CONCEPTS

• **Computer systems organization → Embedded systems**; Robotics.

KEYWORDS

datasets, neural networks, flash lidar, object detection

ACM Reference Format:
Deep Samal, Evan T Gebhardt, and Marilyn Wolf. 2018. Flash Lidar Object Detection and Tracking. In *Proceedings of Anchorage '19: SIGKDD International Workshop on AIoT (Anchorage '19).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

With the advent of autonomous/semi-autonomous vehichle(AV) technologies, Lidar has become the center of attention for its ability to sense 3D structure of the world. Most of the existing research on lidar object detection has been done in context of scanning lidars, which are bulky and have moving parts, like the rotating axle to generate 360 degree scan. These limitations make them unsuitable for large scale adaptation. Therefore, there has been an increasing interest in flash lidars lately, as they are smalller in size and have no moving parts.

While both lidar technologies work on same principle of determining depth from time-of-flight (TOF) information, there are some differences which makes processing flash lidar data difficult, especially if it has to be used in the same context as scanning lidars. Most of the publicly available lidar dataset, e.g. KITTI[4] and APOLLO[5] are created from a scanning lidar which creates a 3D point cloud. This is ideal for AV use-case, where 3D localization of an object is important. But in this paper we are interested in 2D localization of an object using flash lidar data.

2 BACKGROUND

Existing algorithms for object detection using lidar data can be broadly classified into 2 categories:

1. RGB + Lidar data, these techniques use both visual and lidar image simultaneously to detect objects. This requires the multi-camera system to be calibrated and synchronized. Such algorithms can either use more robust RGB detection to crop out area of interest in a point cloud, and then do 3D localization within the sub-section of point-cloud[12] or use both modalities simultaneously[3, 6]. Each point-cloud frame from a scanning lidar can have more than 100K points with an FoV of 360 degrees, the technique of cropping out a RoI from a RGB image reduces complexity and improves detection performance. These cascaded algorithms suffer from low frame rate as the lidar data pipeline has to wait for RGB processing to be complete.

2. Lidar data only, these techniques process 3D point clouds directly to detect objects [7, 10, 15]. In either case, expected output is a 3D bounding box. The most common dataset used in this area of research is KITTI dataset[4], which has about 15k annotated point-clouds with 3D bounding box for objects. There is another line of work on object detection from low power ToF camera such as Kinect [11], but such systems can only be used reliably in indoor scenarios. A Kinect sensor and flash lidar have a large difference in resolution and range, algorithms based on Kinect sensors can not be easily extended to outdoor scenes. LMNet[10] is the closest algorithm to ours. It converts 3D pointclouds to 2D images in spherical co-ordinate system, which is then fed into a 2D CNN with SSD head to detect objects. It was trained on KITTI dataset, and predicts 3D bounding boxes for objects. In addition to 2 channel depth and intensity image, additional input channels were created with azimuth, elevation and height from ground information. However, the performance of this network was worse than other lidar only techniques.

3 DATASET

We have created a flash lidar dataset with over 8k frames. Each frame includes a range channel, and intensity channel. The dataset focuses on cluttered urban environments. In addition, the dataset includes a variety of viewpoints and surveillance scenarios that are not typically found in traditional lidar datasets. A shift away from autonomous vehicle focused scenarios can help to facilitate future lidar research in other application areas.

3.1 Hardware Set-up

The lidar data was captured from an ASC Peregrine Flash Lidar, operating laser wavelength is 1570 nm and has a maximum frame rate of 25 fps. It has an FoV of 30 degrees azimuth and 7.5 degrees elevation, frame rate of capture was 5-10 fps. We also used a FLIR chameleon3 RGB camera synchronized with Flash lidar. As both lidar and RGB were triggered via software, synchronization was handled in software as well.

The output of our flash lidar is a 32x128 image, with 2 channels-depth and intensity, while the RGB output is a 3 channel 1080x1280 image. The maximum range of our lidar is 100 m with range resolution of 5 cm.

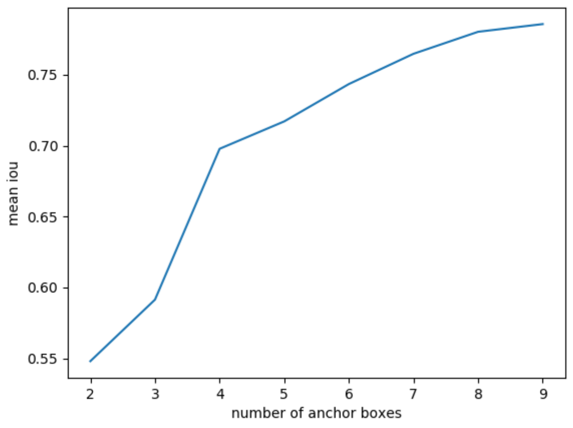

Figure 2: Mean iou for different number of anchor boxes.

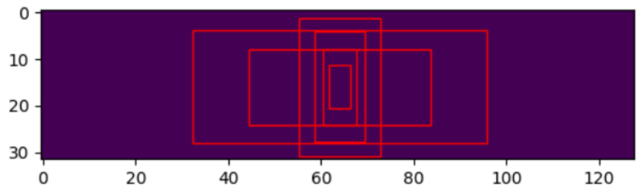

Figure 3: Anchor box prior visualization.

category	number of instances
person	16719
car	1055
bike	223
van	88
bus	140

Table 1: Data category split

3.2 Annotations

Lidar frames were annotated using CVAT annotation tool[1]. There are 5 classes annotated- person, car, van, bus, and bike. The number of instances of each category is shown in table 1. This dataset contains 8,282 annotated frames, out of which 6,876 frames are in training data and 1,406 frames are in validation data.

The lidar dataset also includes information for tracking. We collected 13 sequences containing 374 unique tracks acquired from urban campus environments.

4 CHALLENGES

As mentioned above the FoV of our lidar is less than commonly used scanning lidars such as Velodyne HDL-64, which have an azimuth of 360 degrees and elevation of 26 degrees. Our flash lidar image contains 4,096 pixels compared to more than 100,000 points in scanning lidar(360 degree FoV). Also Flash lidar's horizontal

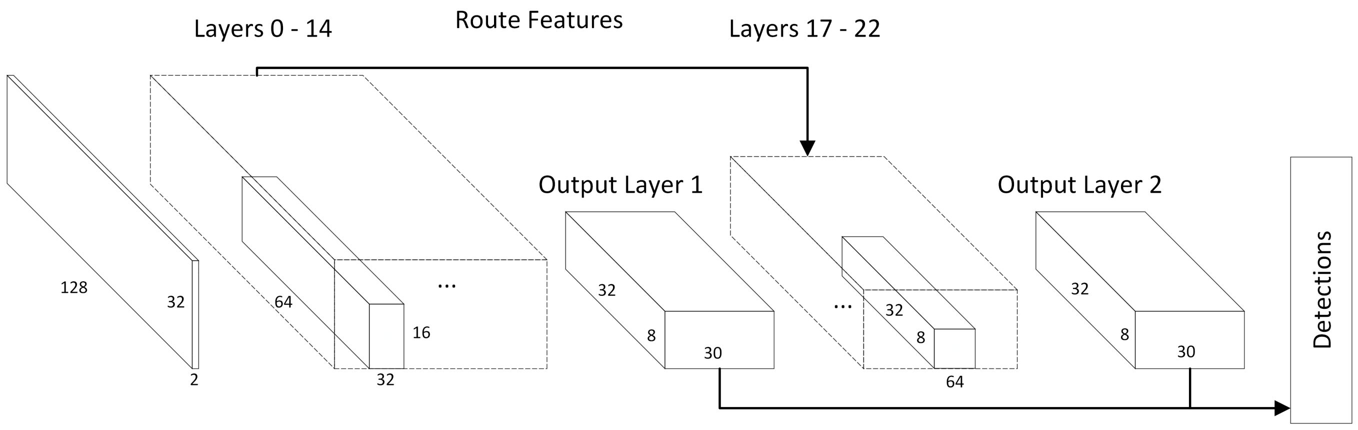

Figure 4: Network Architecture of Object Detector.

angular resolution is 0.234 degrees compared to 0.08 degrees for velodyne scanning lidar. These values lead to quantization effects in the lidar images, particularly at depth boundaries.

We tried to create point clouds from depth images using the intrinsic parameters of our flash lidar. However, calculating lens distortion is difficult because of low image resolution. Additionally measuring lens distortion by using the same lens on a different sensor that produces high resolution image is not an easy option. This is because the lens operates only in IR wavelengths and we would need an IR sensor. Because of the above problems, we decided to use the lidar image produced by lidar camera directly for object detection instead of converting it into 3D point cloud and re-using existing 3D detection techniques.

While LMNet's performance did indicate flaws with directly using lidar depth image for detection, it should be noted that LMNet was trying to regress 3D bounding box co-ordinates.

5 FLASH LIDAR OBJECT DETECTION

Our object detection is based on YOLOv3[14], which is a real-time object detection framework for RGB images. YOLOv3 divides the entire image into 'nxn' grids, object detection proposals are generated from each grid. Detected objects are in the form of bounding boxes, but directly regressing for box co-ordinates is difficult, therefore, box priors are used. Instead of regressing for box position and dimension, the network has to predict offset of box position from center of the grid and offset of box dimensions from pre-defined anchor boxes. These box priors can be generated by analyzing the dataset.

Each image is processed through a 2D CNN with a fully convolution network (FCN) at the end. Finally, output of FCN is fed into YOLO layer(s), the output of a YOLO layer is a matrix of size- 30 x (n x n). 30 features represent x, y, width and height offset, objectness score/confidence and class score for each category (equals to 5 in our case) at a particular grid for each of the anchor box priors, at any given YOLO layer, we use 3 out of 6 anchor box priors. At the end non-maximal suppression is used to limit the number of candidates predicted by this network.

5.1 Architecture

YOLOv3 expects input with a square resolution where the height and width dimensions are multiples of 32. The value of 32 is known as the network stride and can be generalized to other values. The base input resolution for Yolov3 is 608x608, but for our modified network we maintained the input resolution of our lidar image as 32x128. We did this because upsampling our images would not create any meaningful information for the network. Aspect ratio of our image is 1:4, this ratio is maintained throughout all the layers of CNN. We did not use asymmetrical striding across x and y dimensions to create a square FCN output as pixels of our lidar image are almost square, and uneven striding would make sense if the pixels were rectangular.

Network architecture diagram is shown in fig 4 and details of network layers are shown in table 2. Adhering to YOLOv3 design principles, there are 2 output branches, one for detecting large objects and other for small objects. At any given output layer (YOLO layer), we predict 3 anchor boxes. Thus the total number of box predictions is 6.

The number of anchor boxes and their prior dimensions was decided by analyzing all ground truth bounding boxes in our training data. Similar to [13], we did a k-means clustering on all the bounding boxes in our training set and calculated clustering loss and mean intersection over union (iou) for different number of anchor boxes. While more number of anchor boxes will lead to better accuracy (see figure 2 for details), it will also lead to a bigger network (more parameters). Therefore, we decided on 6 anchor boxes as a compromise between accuracy and computational efficiency. Visualization of anchor boxes in figure 3.

Our network is a smaller version of one of the tiny YOLO networks. While original tiny YOLOv3 network has 37 layers with about 8.85 million parameters, our modified network has 25 layers with 102,460 parameters. Thus our network requires considerably less computational resource and power.

5.2 Results

The mean average precision (mAP) of our network is 59.2%. The state of the art performance on the COCO 2017 bounding box

Layer #	Layer Type	# Parameters	Size
0	Conv weight	576	[32, 2, 3, 3]
1	BN weight	32	[32]
2	BN bias	32	[32]
3	Conv weight	9216	[32, 32, 3, 3]
4	BN weight	32	[32]
5	BN bias	32	[32]
6	Conv weight	9216	[32, 32, 3, 3]
7	BN weight	32	[32]
8	BN bias	32	[32]
9	Conv weight	2048	[64, 32, 1, 1]
10	BN weight	64	[64]
11	BN bias	64	[64]
12	Conv weight	18432	[32, 64, 3, 3]
13	BN weight	32	[32]
14	BN bias	32	[32]
15	Conv weight	960	[30, 32, 1, 1]
16	Conv bias	30	[30]
17	Conv weight	4096	[64, 64, 1, 1]
18	BN weight	64	[64]
19	BN bias	64	[64]
20	Conv weight	55296	[64, 96, 3, 3]
21	BN weight	64	[64]
22	BN bias	64	[64]
23	Conv weight	1920	[30, 64, 1, 1]
24	Conv bias	30	[30]

Table 2: Network layer details (learnable parameters only)

Class Category	Precision	Recall	mAP
all	0.671	0.622	0.592
person	0.726	0.681	0.647
car	0.439	0.444	0.439
bike	0.160	0.165	0.160

Table 3: Test result

challenge was 52.5%. While the COCO challenge is significantly harder, we have demonstrated with limited data and principled architecture design we can achieve a quality mAP value. Please refer to table 3 for more details of test performance.

The limited data also explains some of our mAP results. We have an insignificant number of bus, van, and bike instances in our dataset relative to people and cars. We can add more data, or investigate zero-shot learning techniques that can be applied to our lidar data. Inference Speed of our network is 227 fps, thus it is ideal for real-time tracking.

Figure 5 is a plot between total training and test/validation loss with respect to epochs. Initial learning rate is set to 0.01 and is decayed by factor of 10 at epochs 20, 30 and 40, this prevents network from overfitting. As it can be observed, difference between test and training loss decreases until epoch 30, which indicates the network is not overfitting yet, but lack of improvement beyond that indicates there is still room for improvement.

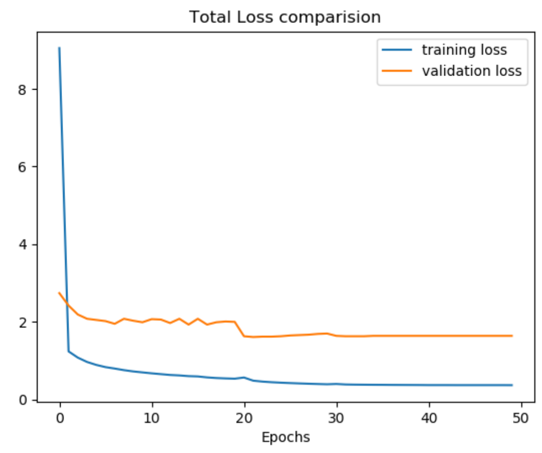

Figure 5: Evolution of loss with epoch.

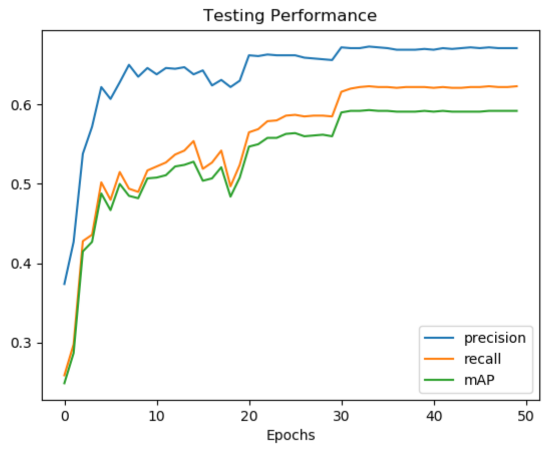

Figure 6: Evolution of network performance on validation data with epoch.

We plan to add more diverse training and testing data in future to address this issue. Similarly, fig 6 shows improvement in test performance with each epoch. As expected performance reaches an asymptote around epoch 40. Because of the above reasons we are limiting training to 50 epochs.

Sample detections from lidar images are shown in fig 7 (images displayed are from depth channel of lidar frames).

5.3 Tracking Results

In addition to the detection results we also performed benchmarking for multiple object tracking [8] [9]. There is a large body of research into multiple object tracking. Challenges, such as, the MOT challenge facilitate this research and provide us with some standards to compare against. They typically use the CLEAR MOT metrics for evaluating tracking performance. These metrics give different information about how your tracking algorithm is performing. They give

Name	IDF1↑	IDP↑	IDR↑	GT	MT	PT	ML	FP↓	FN↓	IDs↓	FM↓	MOTA↑	MOTP↓
Lidar Test 1	19.90%	30.60%	14.70%	8	1	3	4	167	982	26	48	25.20%	0.285
Lidar Test 2	47.10%	74.80%	34.40%	18	3	5	10	13	288	8	17	39.30%	0.195
Lidar Test 3	51.70%	76.80%	39.00%	50	6	21	23	52	748	25	48	41.70%	0.241

Table 4: Tracking evaluations based on CLEAR MOT metrics for lidar sequences.

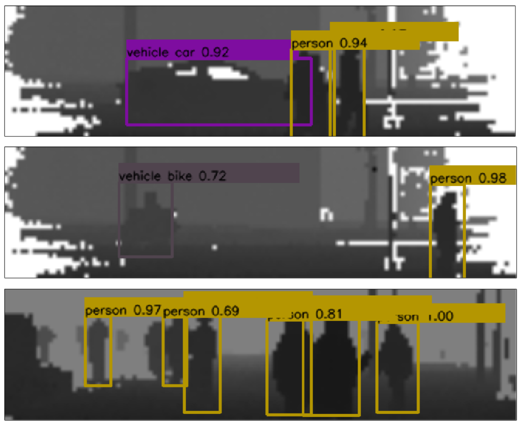

Figure 7: Sample detections from network.

feedback about overall accuracy, track continuity, and bounding box recall and precision. We evaluate tracking performance using these metrics.

The tracking algorithm we used to benchmark our lidar test sequences is the SORT algorithm [2]. SORT stands for simple online realtime tracker. It is designed to be fast and efficient. This is desirable for our applications so we choose to benchmark against this tracking algorithm, instead of one that gives better performance on the MOT metrics. The SORT algorithm effectively adds no overheads to the detection pipeline. However, its performance is strongly coupled to the object detector performance.

SORT uses bounding box intersection over union as its association metric, and a kalman filter to propagate bounding boxes. The filter tracks a center box coordinate, aspect ratio, and area of the box. It also uses linear velocity for the center coordinates and scale. The association metric is simple, but effective given you have a high enough framerate relative to the speed of a given object in your video sequence.

The results of the SORT algorithm on the lidar object detections are shown in Table 4. The results show good performance in MOTA, and MOTP, which represent accuracy and precision of tracks. However, the tracker fragments many of the tracks into multiple tracks. This is shown by the IDF1 , IDP, and IDR metrics performing relatively poorly. The track fragmentations is a direct consequence of the object detector performance. As the dataset expands, we expect these metrics to rise near the level of visual object trackers.

6 CONCLUSION AND FUTURE WORK

We have demonstrated a framework for real time object detection and tracking on flash lidar depth-intensity images. The entire pipeline can run at 227 fps, and requires only 100k learned parameters. All of the flash lidar data used in this paper will also be released for open use. The preparation, annotation, and formatting of this data for use took considerable effort. Our hope is that releasing this data will help to further lidar research in areas other autonomous vehicles, such as surveillance, and person re-identification.

We plan to expand our lidar dataset to include scenes with varying dynamics and weather conditions. The performance of the flash lidar object detector on a moving platform, and subject to different weather conditions like rain, dust, and snow are of great interest to us. Another goal will be to create a dataset that includes multiple modalities. Lidar at different wavelength, thermal-infrared, and visual data are all possibilities. This future data will help to further research in target tracking and association across these modalities.

7 ACKNOWLEDGEMENTS

This research was funded by DARPA Contract Number HR0011-17-2-0045

REFERENCES

[1] 2019. Computer Vision Annotation Tool (CVAT). https://github.com/opencv/cvat
[2] Alex Bewley, ZongYuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. 2016. Simple Online and Realtime Tracking. *CoRR* abs/1602.00763 (2016). arXiv:1602.00763 http://arxiv.org/abs/1602.00763
[3] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. 2017. Multi-View 3D Object Detection Network for Autonomous Driving. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*
[4] A Geiger, P Lenz, C Stiller, and R Urtasun. 2013. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research* 32, 11 (2013), 1231–1237. https://doi.org/10.1177/0278364913491297
[5] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. 2018. The ApolloScape Dataset for Autonomous Driving. *CoRR* abs/1803.06184 (2018). arXiv:1803.06184 http://arxiv.org/abs/1803.06184
[6] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander. 2018. Joint 3D Proposal Generation and Object Detection from View Aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 1–8. https://doi.org/10.1109/IROS.2018.8594049
[7] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. 2018. PointPillars: Fast encoders for object detection from point clouds. *arXiv preprint arXiv:1812.05784* (2018).
[8] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler. 2015. MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking. *arXiv:1504.01942 [cs]* (April 2015). http://arxiv.org/abs/1504.01942 arXiv: 1504.01942.
[9] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. 2016. MOT16: A Benchmark for Multi-Object Tracking. *arXiv:1603.00831 [cs]* (March 2016). http://arxiv.org/abs/1603.00831 arXiv: 1603.00831.
[10] K. Minemura, H. Liau, A. Monrroy, and S. Kato. 2018. LMNet: Real-time Multiclass Object Detection on CPU Using 3D LiDAR. In *2018 3rd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS).* 28–34. https://doi.org/10.1109/ACIRS.2018.8467245
[11] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. 2012. Indoor Segmentation and Support Inference from RGBD Images. In *ECCV.*

[12] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. 2018. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 918–927.

[13] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7263–7271.

[14] Joseph Redmon and Ali Farhadi. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

[15] Yin Zhou and Oncel Tuzel. 2018. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4490–4499.