Complex-YOLO: An Euler-Region-Proposal for Real-time 3D Object Detection on Point Clouds

Martin Simon[†*], Stefan Milz[†], Karl Amende[†*], Horst-Michael Gross[*]

Valeo Schalter und Sensoren GmbH[†], Ilmenau University of Technology[*]
{martin.simon,stefan.milz,karl.amende}@valeo.com
horst-michael.gross@tu-ilmenau.de

Abstract. Lidar based 3D object detection is inevitable for autonomous driving, because it directly links to environmental understanding and therefore builds the base for prediction and motion planning. The capacity of inferencing highly sparse 3D data in real-time is an ill-posed problem for lots of other application areas besides automated vehicles, e.g. augmented reality, personal robotics or industrial automation. We introduce Complex-YOLO, a state of the art real-time 3D object detection network on point clouds only. In this work, we describe a network that expands YOLOv2, a fast 2D standard object detector for RGB images, by a specific complex regression strategy to estimate multi-class 3D boxes in Cartesian space. Thus, we propose a specific Euler-Region-Proposal Network (E-RPN) to estimate the pose of the object by adding an imaginary and a real fraction to the regression network. This ends up in a closed complex space and avoids singularities, which occur by single angle estimations. The E-RPN supports to generalize well during training. Our experiments on the KITTI benchmark suite show that we outperform current leading methods for 3D object detection specifically in terms of efficiency. We achieve state of the art results for cars, pedestrians and cyclists by being more than five times faster than the fastest competitor. Further, our model is capable of estimating all eight KITTI-classes, including Vans, Trucks or sitting pedestrians simultaneously with high accuracy.

方法衍生

Keywords: 3D Object Detection, Point Cloud Processing, Lidar, Autonomous Driving

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 [1]. This allows us to develop object detection algorithms for autonomous driving that estimate the position and the heading of different objects accurately in 3D [2] [3] [4] [5] [6] [7] [8] [9]. Compared to images, Lidar point clouds are sparse with a varying density distributed all over

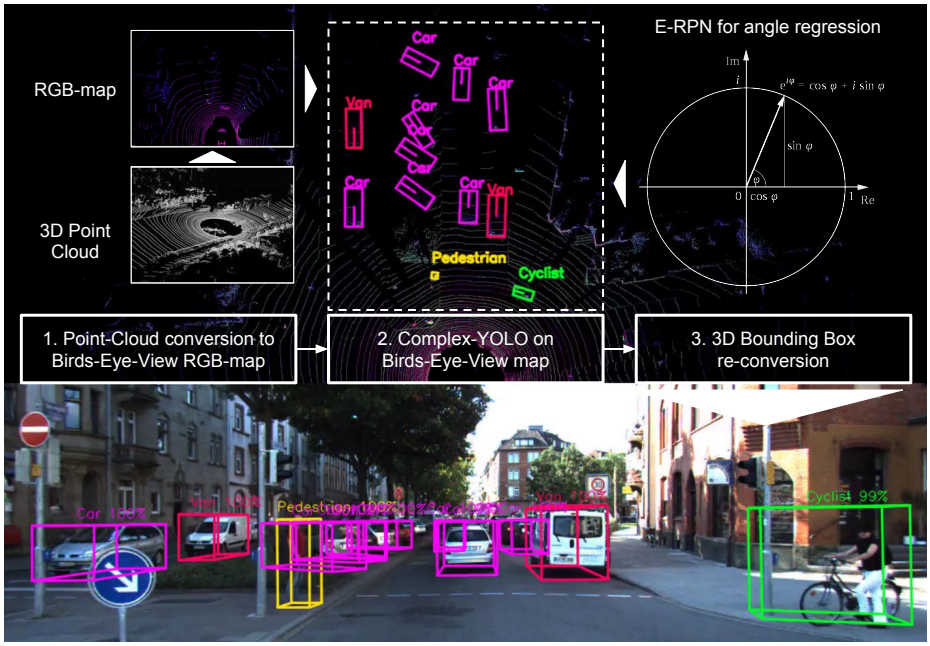

Fig. 1. Complex-YOLO is a very efficient model that directly operates on Lidar only based birds-eye-view RGB-maps to estimate and localize accurate 3D multiclass bounding boxes. The upper part of the figure shows a bird view based on a Velodyne HDL64 point cloud (Geiger et al. [1]) such as the predicted objects. The lower one outlines the re-projection of the 3D boxes into image space. Note: Complex-YOLO needs no camera image as input, it is Lidar based only.

the measurement area. Those points are unordered, they interact locally and could mainly be not analyzed isolated. Point cloud processing should always be invariant to basic transformations [10] [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 [12] [13] [14] [15] [16] [17] [18] [19] [20] [21]. Research focus was mainly a trade-off 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) [3] [22] [15] or a similar grid based RPN-approach [13]. 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 [3]:

1. Direct point cloud processing using Multi-Layer-Perceptrons [5] [10] [11] [23] [24]

2. Translation of Point-Clouds into voxels or image stacks by using Convolutional Neural Networks (CNN) [2] [3] [4] [6] [8] [9] [25] [26]
3. Combined fusion approaches [2] [7]

1.1 Related Work

Recently, Frustum-based Networks [5] have shown high performance on the KITTI Benchmark suite. The model is ranked[1] 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-eye-view 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).

1.2 Contribution

To our surprise, no one is achieving real-time efficiency in terms of autonomous driving so far. Hence, we introduce the first slim and accurate model that is capable of running faster than 50fps on a NVIDIA TitanX GPU. We use the multi-view idea (MV3D) [5] for point cloud pre-processing and feature extraction. However, we neglect the multi-view fusion and generate one single birds-eye-view RGB-map (see Fig. 1) that is based on Lidar only, to ensure efficiency.

[1] The ranking refers to the time of submission: 14th of march in 2018

On top, we present Complex-YOLO, a 3D version of YOLOv2, which is one of the fastest state-of-the-art image object detectors [13]. Complex-YOLO is supported by our specific E-RPN that estimates the orientation of objects coded by an imaginary and real part for each box. The idea is to have a closed mathematical space without singularities for accurate angle generalization. Our model is capable to predict exact 3D boxes with localization and an exact heading of the objects in real-time, even if the object is based on a few points (e.g. pedestrians). Therefore, we designed special anchor-boxes. Further, it is capable to predict all eight KITTI classes by using only Lidar input data. We evaluated our model on the KITTI benchmark suite. In terms of accuracy, we achieved on par results for cars, pedestrians and cyclists, in terms of efficiency we outperform current leaders by minimum factor of 5. The main contributions of this paper are:

1. This work introduces Complex-YOLO by using a new E-RPN for reliable angle regression for 3D box estimation.
2. We present real-time performance with high accuracy evaluated on the KITTI benchmark suite by being more than five times faster than the current leading models.
3. We estimate an exact heading of each 3D box supported by the E-RPN that enables the prediction of the trajectory of surrounding objects.
4. Compared to other Lidar based methods (e.g. [3]) our model efficiently estimates all classes simultaneously in one forward path.

2 Complex-YOLO

This section describes the grid based pre-processing of the point clouds, the specific network architecture, the derived loss function for training and our efficiency design to ensure real-time performance.

2.1 Point Cloud Preprocessing

The 3D point cloud of a single frame, acquired by Velodyne HDL64 laser scanner [1], is converted into a single birds-eye-view RGB-map, covering an area of 80m x 40m (see Fig.4) directly in front of the origin of the sensor. Inspired by Chen et al. (MV3D) [5], the RGB-map is encoded by height, intensity and density. The size of the grid map is defined with $n = 1024$ and $m = 512$. Therefore, we projected and discretized the 3D point clouds into a 2D grid with resolution of about $g = 8cm$. Compared to MV3D, we slightly decreased the cell size to achieve less quantization errors, accompanied with higher input resolution. Due to efficiency and performance reasons, we are using only one instead of multiple height maps. Consequently, all three feature channels (z_r, z_g, z_b with $z_{r,g,b} \in \mathbb{R}^{m \times n}$) are calculated for the whole point cloud $\mathcal{P} \in \mathbb{R}^3$ inside the covering area Ω. We consider the Velodyne within the origin of $\mathcal{P}_\Omega$ and define:

$$\mathcal{P}_\Omega = \{\mathcal{P} = [x, y, z]^T | x \in [0, 40m], y \in [-40m, 40m], z \in [-2m, 1.25m]\} \quad (1)$$

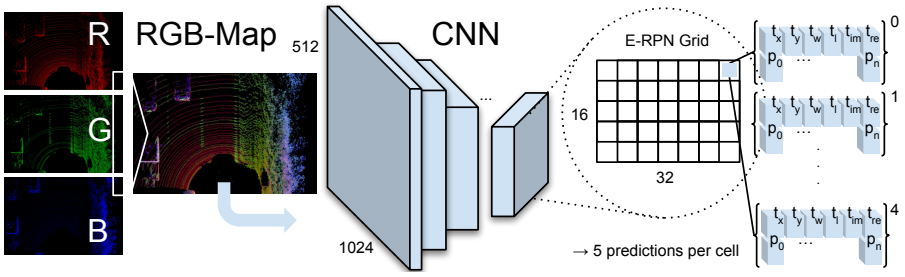

Fig. 2. Complex-YOLO Pipeline. We present a slim pipeline for fast and accurate 3D box estimations on point clouds. The RGB-map is fed into the CNN (see Tab. 1). The E-RPN grid runs simultaneously on the last feature map and predicts five boxes per grid cell. Each box prediction is composed by the regression parameters t (see Fig. 3) and object scores p with a general probability p_0 and n class scores $p_1...p_n$.

We choose $z \in [-2m, 1.25m]$, considering the Lidar z position of 1.73m [1], to cover an area above the ground to about 3m height, expecting trucks as highest objects. With the aid of the calibration [1], we define a mapping function $\mathcal{S}_j = f_{\mathcal{PS}}(\mathcal{P}_{\Omega i}, g)$ with $\mathcal{S} \in \mathbb{R}^{m \times n}$ mapping each point with index i into a specific grid cell $\mathcal{S}_j$ of our RGB-map. A set describes all points mapped into a specific grid cell:

$$\mathcal{P}_{\Omega i \to j} = \{\mathcal{P}_{\Omega i} = [x, y, z]^T | \mathcal{S}_j = f_{\mathcal{PS}}(\mathcal{P}_{\Omega i}, g)\} \tag{2}$$

Hence, we can calculate the channel of each pixel, considering the Velodyne intensity as $I(\mathcal{P}_\Omega)$:

$$\begin{aligned} z_g(\mathcal{S}_j) &= \max(\mathcal{P}_{\Omega i \to j} \cdot [0, 0, 1]^T) \\ z_b(\mathcal{S}_j) &= \max(I(\mathcal{P}_{\Omega i \to j})) \\ z_r(\mathcal{S}_j) &= \min(1.0, \log(N+1)/64) \quad N = |\mathcal{P}_{\Omega i \to j}| \end{aligned} \tag{3}$$

Here, N describes the number of points mapped from $\mathcal{P}_{\Omega i}$ to $\mathcal{S}_j$, and g is the parameter for the grid cell size. Hence, z_g encodes the maximum height, z_b the maximum intensity and z_r the normalized density of all points mapped into $\mathcal{S}_j$ (see Fig. 2).

2.2 Architecture

The Complex-YOLO network takes a birds-eye-view RGB-map (see section 2.1) as input. It uses a simplified YOLOv2 [13] CNN architecture (see Tab. 1), extended by a complex angle regression and E-RPN, to detect accurate multi-class oriented 3D objects while still operating in real-time.

Euler-Region-Proposal. Our E-RPN parses the 3D position $b_{x,y}$, object dimensions (width b_w and length b_l) as well as a probability p_0, class scores $p_1...p_n$ and finally its orientation b_ϕ from the incoming feature map. In order to get

proper orientation, we have modified the commonly used Grid-RPN approach, by adding a complex angle $arg(|z|e^{ib_\phi})$ to it:

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_l &= p_l e^{t_l} \\ b_\phi &= arg(|z|e^{ib_\phi}) = \arctan_2(t_{Im}, t_{Re}) \end{aligned} \tag{4}$$

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 (32x16 see Tab. 1) we predict five objects including a probability score and class scores resulting in 75 features each, visualized in Fig. 2.

Anchor Box Design. The YOLOv2 object detector [13] 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).

Table 1. Complex-YOLO Design. Our nal model has 18 convolutional and 5 maxpool layers, as well as 3 intermediate layers for feature reorganization respectively.

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**	

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, similar to [27]. The angle is given simply by using $\arctan_2(t_{im}, t_{re})$. 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 function (7).

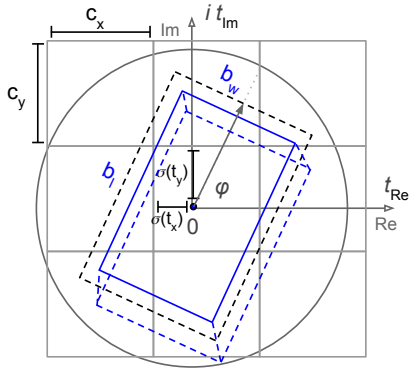

Fig. 3. 3D Bounding box regression. We predict oriented 3D bounding boxes based on the regression parameters shown in YOLOv2 [13], as well as a complex angle for box orientation. The transition from 2D to 3D is done by a predefined height based on each class.

2.3 Loss Function

Our network optimization loss function $\mathcal{L}$ is based on the the concepts from YOLO [12] and YOLOv2 [13], who defined $\mathcal{L}_{\text{Yolo}}$ as the sum of squared errors using the introduced multi-part loss. We extend this approach by an Euler regression part $\mathcal{L}_{\text{Euler}}$ to get use of the complex numbers, which have a closed mathematical space for angle comparisons. This neglect singularities, which are common for single angle estimations:

$$\mathcal{L} = \mathcal{L}_{\text{Yolo}} + \mathcal{L}_{\text{Euler}} \tag{5}$$

The Euler regression part of the loss function is defined with the aid of the Euler-Region-Proposal (see Fig. 3). Assuming that the difference between the complex numbers of prediction and ground truth, i.e. $|z|e^{\mathrm{i}b_\phi}$ and $|\hat{z}|e^{\mathrm{i}\hat{b}_\phi}$ is always located on the unit circle with $|z| = 1$ and $|\hat{z}| = 1$, we minimize the absolute value of the squared error to get a real loss:

$$\mathcal{L}_{\text{Euler}} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left| (e^{\mathrm{i}b_\phi} - e^{\mathrm{i}\hat{b}_\phi})^2 \right| \tag{6}$$

$$= \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[(t_{im} - \hat{t}_{im})^2 + (t_{re} - \hat{t}_{re})^2 \right] \tag{7}$$

Where λ_{coord} is a scaling factor to ensure stable convergence in early phases and $\mathbb{1}_{ij}^{obj}$ denotes that the jth bounding box predictor in cell i has highest intersection over union (IoU) compared to ground truth for that prediction. Furthermore the comparison between the predicted box P_j and ground truth G with IoU $\frac{P_j \cap G}{P_j \cup G}$, where $P_j \cap G = \{x : x \in P_j \wedge x \in G\}$, $P_j \cup G\{x : x \in P_j \vee x \in G\}$ is adjusted to

handle rotated boxes as well. This is realized by the theory of intersection of two 2D polygon geometries and union respectively, generated from the corresponding box parameters b_x, b_y, b_w, b_l and b_ϕ.

2.4 Efficiency Design

The main advantage of the used network design is the prediction of all bounding boxes in one inference pass. The E-RPN is part of the network and uses the output of the last convolutional layer to predict all bounding boxes. Hence, we only have one network, which can be trained in an end-to-end manner without specific training approaches. Due to this, our model has a lower runtime than other models that generate region proposals in a sliding window manner [22] with prediction of offsets and the class for every proposal (e.g. Faster R-CNN [15]). In Fig. 5 we compare our architecture with some of the leading models on the KITTI benchmark. Our approach achieves a way higher frame rate while still keeping a comparable mAP (mean Average Precision). The frame rates were directly taken from the respective papers and all were tested on a Titan X or Titan Xp. We tested our model on a Titan X and an NVIDIA TX2 board to emphasize the real-time capability (see Fig. 5).

3 Training & Experiments

We evaluated Complex-YOLO on the challenging KITTI object detection benchmark [1], which is divided into three subcategories 2D, 3D and birds-eye-view object detection for *Cars*, *Pedestrians* and *Cyclists*. Each class is evaluated 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 do not ran the 2D object detection benchmark, since our input is Lidar based only.

3.1 Training Details

We trained our model from scratch via stochastic gradient descent with a weight decay of 0.0005 and momentum 0.9. Our implementation is based on modified version of the Darknet neural network framework [28]. First, we applied our pre-processing (see Section 2.1) to generate the birds-eye-view RGB-maps from Velodyne samples. Following the principles from [2] [3] [29], 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. In contrast, e.g. VoxelNet [3] modified and optimized the model for different classes. We suffered from the available ground truth data, because it was intended for camera detections first. The class distribution with more than 75% *Car*, less than 4% *Cyclist* and less

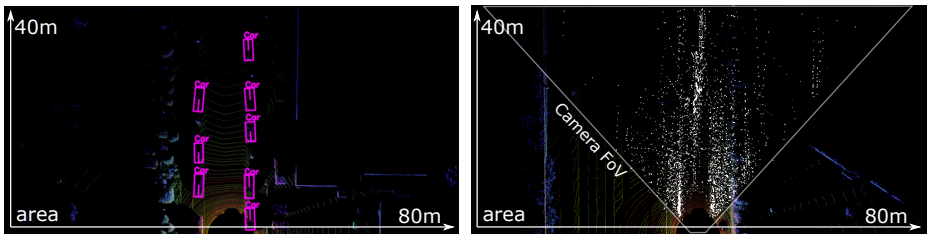

Fig. 4. Spatial ground truth distribution. The figure outlines the size of the birds-eye-view area with a sample detection on the left. The right shows a 2D spatial histogram of annotated boxes in [1]. The distribution outlines the horizontal field of view of the camera used for annotation and the inherited blind spots in our map.

than 15% *Pedestrian* is disadvantageous. Also, more than 90% of all the annotated objects are facing the car direction, facing towards the recording car or having similar orientations. On top, Fig. 4 shows a 2D histogram for spatial object locations from birds-eye-view perspective, where dense points indicate more objects at exactly this position. This inherits two blind spot for birds-eye-view map. Nevertheless we saw surprising good results for the validation set and other recorded unlabeled KITTI sequences covering several use case scenarios, like urban, highway or inner city.

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. Due to the fine grained requirements, when using a birds-eye-view approach, slight changes in predicted features will have a strong impact on resulting box predictions. We used batch normalization for regularization and a linear activation $f(x) = x$ for the last layer of our CNN, apart from that the leaky rectified linear activation:

$$f(x) = \begin{cases} x, & x > 0 \\ 0.1x, & \text{otherwise} \end{cases} \tag{8}$$

3.2 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 [1] (see Fig. 4. We used the average precision (AP) metric to compare the results. 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.

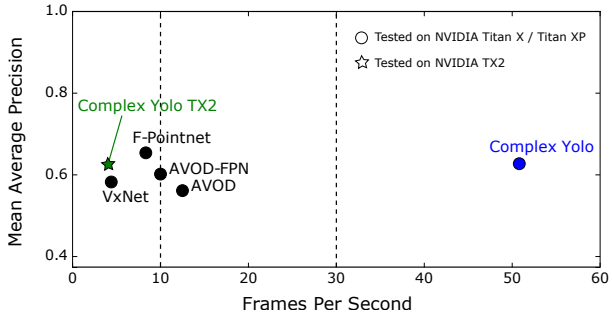

Fig. 5. Performance comparison. This plot shows the mAP in relation to the runtime (fps). All models were tested on a Nvidia Titan X or Titan Xp. Complex-Yolo achieves accurate results by being five times faster than the most effective competitor on the KITTI benchmark [1]. We compared to the five leading models and measured our network on a dedicated embedded platform (TX2) with reasonable efficiency (4fps) as well. Complex-Yolo is the first model for real-time 3D object detection.

Birds-Eye-View. Our evaluation results for the birds-eye-view detection are presented in Tab. 2. This benchmark uses bounding box overlap for comparison. For a better overview and to rank the results, similar current leading methods are listed as well, but performing on the official KITTI test set. Complex-YOLO consistently outperforms all competitors in terms of runtime and efficiency, while still manages to achieve comparable accuracy. With about 0.02s runtime on a Titan X GPU, we are 5 times faster than AVOD [7], considering their usage of a more powerful GPU (Titan Xp). Compared to VoxelNet [3], which is also Lidar based only, we are more than 10 times faster and MV3D [2], the slowest competitor, takes 18 times as long.

Table 2. Performance comparison for birds-eye-view detection. APs (in %) for our experimental setup compared to current leading methods. Note that our method is validated on our splitted validation dataset, whereas all others are validated on the official KITTI test set.

Method	Modality	FPS	Car			Pedestrian			Cyclist		
			Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
MV3D [2]	Lidar+Mono	2.8	86.02	76.90	68.49	-	-	-	-	-	-
F-PointNet [5]	Lidar+Mono	5.9	88.70	84.00	75.33	**58.09**	**50.22**	**47.20**	**75.38**	61.96	54.68
AVOD [7]	Lidar+Mono	12.5	86.80	**85.44**	77.73	42.51	35.24	33.97	63.66	47.74	46.55
AVOD-FPN [7]	Lidar+Mono	10.0	88.53	83.79	**77.90**	50.66	44.75	40.83	62.39	52.02	47.87
VoxelNet [3]	Lidar	4.3	**89.35**	79.26	77.39	46.13	40.74	38.11	66.70	54.76	50.55
Complex-YOLO	Lidar	**50.4**	85.89	77.40	77.33	46.08	45.90	44.20	72.37	**63.36**	**60.27**

3D Object Detection. Tab. 3 shows our achieved results for the 3D bounding box overlap. Since we do not estimate the height information directly with regression, we ran this benchmark with a fixed spatial height location extracted from ground truth similar to MV3D [2]. Additionally as mentioned, we simply injected a predefined height for every object based on its class, calculated from the mean over all ground truth objects per class. This reduces the precision for all classes, but it confirms the good results measured on the birds-eye-view benchmark.

Table 3. Performance comparison for 3D object detection. APs (in %) for our experimental setup compared to current leading methods. Note that our method is validated on our splitted validation dataset, whereas all others are validated on the official KITTI test set.

Method	Modality	FPS	Car			Pedestrian			Cyclist		
			Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
MV3D [2]	Lidar+Mono	2.8	71.09	62.35	55.12	-	-	-	-	-	-
F-PointNet [5]	Lidar+Mono	5.9	81.20	70.39	62.19	**51.21**	**44.89**	**40.23**	**71.96**	56.77	50.39
AVOD [7]	Lidar+Mono	12.5	73.59	65.78	58.38	38.28	31.51	26.98	60.11	44.90	38.80
AVOD-FPN [7]	Lidar+Mono	10.0	**81.94**	**71.88**	**66.38**	46.35	39.00	36.58	59.97	46.12	42.36
VoxelNet [3]	Lidar	4.3	77.47	65.11	57.73	39.48	33.69	31.51	61.22	48.36	44.37
Complex-YOLO	Lidar	**50.4**	67.72	64.00	63.01	41.79	39.70	35.92	68.17	**58.32**	**54.30**

4 Conclusion

In this paper we present the first real-time efficient deep learning model for 3D object detection on Lidar based point clouds. We highlight our state of the art results in terms of accuracy (see Fig. 5) on the KITTI benchmark suite with an outstanding efficiency of more than 50 fps (NVIDIA Titan X). We do not need additional sensors, e.g. camera, like most of the leading approaches. This breakthrough is achieved by the introduction 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. We show the real-time capability even on dedicated embedded platform NVIDIA TX2 (4 fps). 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.

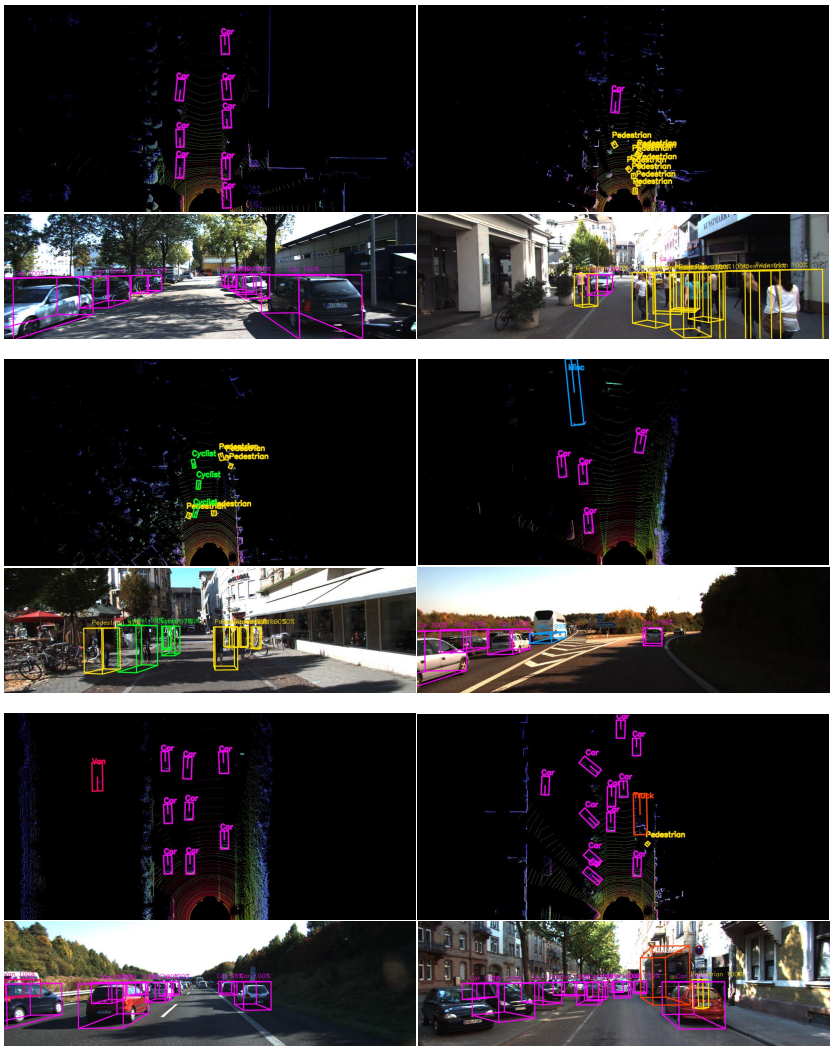

Fig. 6. Visualization of Complex-YOLO results. Note that predictions are exclusively based on birds-eye-view images generated from point clouds. The re-projection into camera space is for illustrative purposes only.

Acknowledgement

First, we would like to thank our main employer Valeo, especially Jörg Schrepfer and Johannes Petzold, for giving us the possibility to do fundamental research. Additionally, we would like to thank our colleague Maximillian Jaritz for his important contribution on voxel generation. Last but not least, we would like to thank our academic partner the TU-Ilmenau for having a fruitful partnership.

References

1. Geiger, A.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). CVPR '12, Washington, DC, USA, IEEE Computer Society (2012) 3354–3361
2. Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3d object detection network for autonomous driving. CoRR **abs/1611.07759** (2016)
3. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. CoRR **abs/1711.06396** (2017)
4. Engelcke, M., Rao, D., Wang, D.Z., Tong, C.H., Posner, I.: Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. CoRR **abs/1609.06666** (2016)
5. Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum pointnets for 3d object detection from RGB-D data. CoRR **abs/1711.08488** (2017)
6. Wang, D.Z., Posner, I.: Voting for voting in online point cloud object detection. In: Proceedings of Robotics: Science and Systems, Rome, Italy (July 2015)
7. Ku, J., Mozifian, M., Lee, J., Harakeh, A., Waslander, S.: Joint 3d proposal generation and object detection from view aggregation. arXiv preprint arXiv:1712.02294 (2017)
8. Li, B., Zhang, T., Xia, T.: Vehicle detection from 3d lidar using fully convolutional network. CoRR **abs/1608.07916** (2016)
9. Li, B.: 3d fully convolutional network for vehicle detection in point cloud. CoRR **abs/1611.08069** (2016)
10. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. CoRR **abs/1612.00593** (2016)
11. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. CoRR **abs/1706.02413** (2017)
12. Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You only look once: Unified, real-time object detection. CoRR **abs/1506.02640** (2015)
13. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. CoRR **abs/1612.08242** (2016)
14. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C., Berg, A.C.: SSD: single shot multibox detector. CoRR **abs/1512.02325** (2015)
15. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. CoRR **abs/1506.01497** (2015)
16. Cai, Z., Fan, Q., Feris, R.S., Vasconcelos, N.: A unified multi-scale deep convolutional neural network for fast object detection. CoRR **abs/1607.07155** (2016)
17. Ren, J.S.J., Chen, X., Liu, J., Sun, W., Pang, J., Yan, Q., Tai, Y., Xu, L.: Accurate single stage detector using recurrent rolling convolution. CoRR **abs/1704.05776** (2017)
18. Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., Urtasun, R.: Monocular 3d object detection for autonomous driving. In: IEEE CVPR. (2016)
19. Girshick, R.B., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. CoRR **abs/1311.2524** (2013)
20. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR **abs/1512.03385** (2015)
21. Chen, X., Kundu, K., Zhu, Y., Ma, H., Fidler, S., Urtasun, R.: 3d object proposals using stereo imagery for accurate object class detection. CoRR **abs/1608.07711** (2016)

22. Girshick, R.B.: Fast R-CNN. CoRR **abs/1504.08083** (2015)
23. Li, Y., Bu, R., Sun, M., Chen, B.: Pointcnn (2018)
24. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds (2018)
25. Xiang, Y., Choi, W., Lin, Y., Savarese, S.: Data-driven 3d voxel patterns for object category recognition. In: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition. (2015)
26. Wu, Z., Song, S., Khosla, A., Tang, X., Xiao, J.: 3d shapenets for 2.5d object recognition and next-best-view prediction. CoRR **abs/1406.5670** (2014)
27. Beyer, L., Hermans, A., Leibe, B.: Biternion nets: Continuous head pose regression from discrete training labels. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **9358** (2015) 157–168
28. Redmon, J.: Darknet: Open source neural networks in c. `http://pjreddie.com/darknet/` (2013–2016)
29. Chen, X., Kundu, K., Zhu, Y., Berneshawi, A., Ma, H., Fidler, S., Urtasun, R.: 3d object proposals for accurate object class detection. In: NIPS. (2015)