Complexer-YOLO: Real-Time 3D Object Detection and Tracking on Semantic Point Clouds

Martin Simon, Karl Amende, Andrea Kraus, Jens Honer,
Timo Sämann, Hauke Kaulbersch and Stefan Milz
Valeo Schalter und Sensoren GmbH
{firstname.lastname}@valeo.com

Horst Michael Gross
Ilmenau University of Technology
horst-michael.gross@tu-ilmenau.de

Abstract

Accurate detection of 3D objects is a fundamental problem in computer vision and has an enormous impact on autonomous cars, augmented/virtual reality and many applications in robotics. In this work we present a novel fusion of neural network based state-of-the-art 3D detector and visual semantic segmentation in the context of autonomous driving. Additionally, we introduce Scale-Rotation-Translation score (SRTs), a fast and highly parameterizable evaluation metric for comparison of object detections, which speeds up our inference time up to 20% and halves training time. On top, we apply state-of-the-art online multi target feature tracking on the object measurements to further increase accuracy and robustness utilizing temporal information. Our experiments on KITTI show that we achieve same results as state-of-the-art in all related categories, while maintaining the performance and accuracy trade-off and still run in real-time. Furthermore, our model is the first one that fuses visual semantic with 3D object detection.

1. Introduction

Over the last few years self-driving cars got more and more into the focus of the automotive industry as well as new mobility players. Today, commercial vehicles already offer manifold automation like assisted or automated parking, adaptive cruise control and even highway pilots. To reach the full level of automation, they require a very precise system of environmental perception, working for every conceivable scenario. Additionally, real world scenarios strictly require real-time performance.

Recent vehicles are equipped with multiple different kind of sensors like ultrasonics, radar, cameras and Lidar (light detection and ranging) as well. With the help of redundancy and sensor fusion, relevant reliability and safety can be achieved. These circumstances significantly boosted the rapid development of sensor technology and the growth of artificial intelligence algorithms for fundamental tasks like object detection and semantic segmentation.

Many modern approaches for these tasks use camera, Lidar or combine both. Compared to camera images, there are some difficulties dealing with Lidar point cloud data. Such point clouds are unordered, sparse and have a highly varying density due to the non-uniform sampling of the 3D space, occlusion and reflection. On the other hand, they offer way higher spatial accuracy and reliable depth information. Therefore, Lidar is more common in the context of autonomous driving. In this paper, we propose Complexer-YOLO, a real-time 3D object detection and tracking on semantic point clouds (see Fig. 1, 2). The main contributions are:

- **Visual Class Features**: Incorporation of visual point-wise Class-Features generated by fast camera-based Semantic Segmentation [39].
- **Voxelized Input**: Extension of Complex-YOLO [42] processing voxelized input features with a variable depth of dimension instead of fixed RGB-maps.
- **Real 3D prediction**: Extension of the regression network to predict 3D box heights and z-offsets to treat targets in three dimensions.
- **Scale-Rotation-Translation score (SRTs)**: We introduce *SRTs*, a new validation metric for 3D boxes, notably faster than intersection over union (IoU), considering the 3DoF pose of the detected object including the yaw angle such as width, height and length.
- **Multitarget-Tracking**: Application of an Online feature tracker decoupled from the detection network, enabling time depending tracking and target instantiation based on realistic, physical assumptions.
- **Realtime capability**: We present a complete novel tracking pipeline with an outstanding overall real-time

in: CVPR-Workshop on Autonomous Driving, Long Beach, USA, IEEE 2019

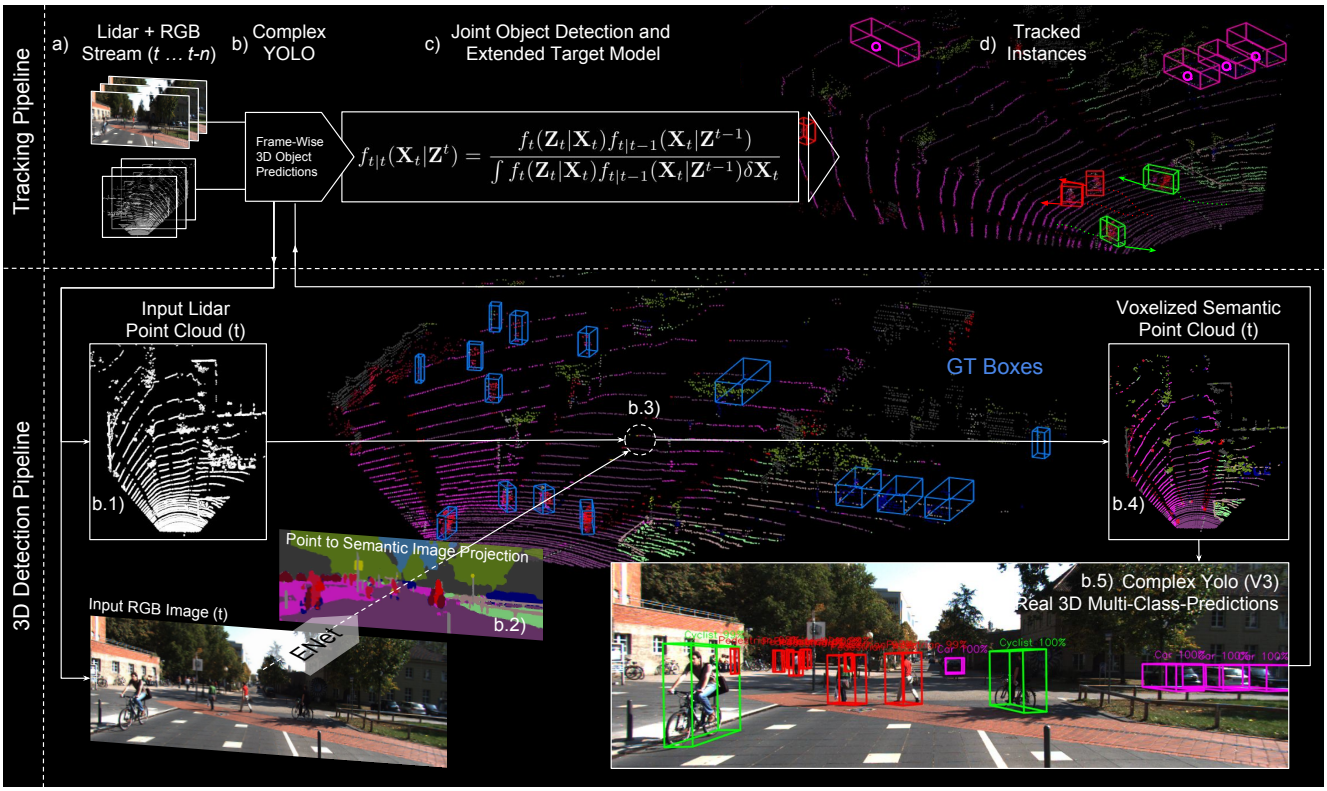

Figure 1: The Complexer-YOLO processing pipeline: We present a novel and complete 3D Detection (b.1-5) and Tracking pipeline (a,b,c,d,e) on Point Clouds in Real-Time. The Tracking-Pipeline is composed by: (a) Lidar + RGB frame grabbing from stream, (b) Frame-wise Complex-YOLO 3D Multiclass predictions, (c) Joint Object and extended Target Model for feature Tracking and (d) 3D object instance tracking within the environmental model. In detail (b) is composed by: (1) The Voxelization of the Lidar frame, (2) the Semantic Segmentation of the RGB image with the aid of ENet, (3) the Point-wise classification by Lidar to Semantic-Image backprojection, (4) the generation of the Semantic Voxel Grid and finally (5): The real 3D Complex YOLO for 3D Multi-class predictions. (see Fig. 2 for more details)

capability, despite state-of-the-art results on semantic segmentation, 3D object detection such as Multitarget-Tracking. The pipeline can be directly brought into every self-driving cars percepting urban scenes.

2. Related Work

In this section, we provide an overview of convolutional neural network (CNN) based object detection, semantic segmentation and multi target tracking.

2.1. 2D Object Detection

Over the last few years many methods for robust and accurate object detection using CNN have been developed. Starting in 2D space on single images, two-stage detectors [35, 12] and one-stage detectors [32, 24, 33, 23, 34, 15] achieved state-of-the-art results, targeting the output of located 2D bounding boxes. Typically, two-stage detectors exploit object proposals and utilize region of interests (RoI) with the help of region proposal networks (RPN) in a first step. Afterwards, they generate the final object predictions using calculated features over the proposed RoIs. As a trade-off for runtime, one-stage detectors skip the proposal generation step and directly output the final object detections. They are usually capable of real-time performance, but mainly outperformed by two-stage detectors in terms of accuracy. YOLOv3 [34], one of the one-stage detectors, combines findings from [32, 33, 11, 22]. It divides the image into a sparse grid, performs multi-scale feature extraction and directly outputs object predictions per grid cell, using dimension clusters as anchor boxes [33].

2.2. 3D Object Detection

Although CNNs were originally designed for image processing, they became a key component for 3D object detection as well. First ideas were to use stereo images as input [3]. Followed by [19] and [6], where 3D convolutions were applied to a voxelized representation of point cloud data and features extracted using 3D CNNs [43], respectively. In

[49], voxel feature encoding was introduced and again processed by CNN to predict objects. Furthermore, VeloFCN [20] created depth maps with the help of front-view projections of 3D point clouds and applied CNN. In contrast, MV3D [4] merged image input with a multi-view representation of point cloud data projected into 2D space. Alternatively, [16] aggregated features from image and birdseye-view representation of point clouds. Another method to fuse camera and Lidar inputs was explored in [28]. In a first step, sub point clouds were extracted in viewing frustums detected by a 2D CNN. Afterwards, a PointNet [29] predicts 3D objects within the frustum point clouds. Recently, PointNet was also used in [45] in combination with 2D CNN and a fusion network. Further similar approaches using birdseye-view representations of point clouds were [25, 47, 42, 1].

2.3. Semantic Segmentation

The goal of semantic segmentation is to classify each pixel of an image into a predefined class. This task is typically achieved by CNNs. Several widely used network architectures have been introduced, e.g. [10, 27, 37, 21]. Similar to the object detection task, there is a trade-off between accuracy and runtime. Therefore, approaches like convolutional factorization, e.g. applied in [14, 26], quantization [30], pruning [13] and dilated convolutions, e.g. applied in [48], came up. ENet [27], one of the most efficient models used a special encoder-decoder structure to highly reduce computational effort. Recently, [39] applied the Channel Pruning method [13] to the ENet to make it more efficient.

2.4. Multi Target Tracking

The task of multi-object tracking (MOT) is usually solved in two phases. First, an algorithm detects objects of interests and second, identical objects in different frames are associated. A widespread approach is using global information about the detections [17, 7]. In contrast to this, online approaches don't have any knowledge of future frames. With this characteristic they have one significant advantage: they are usable in real-world scenarios. Recent work focused especially on tracking of 2D objects from camera input [40, 41]. Online multi-target 3D object tracking based on detections from algorithms with point cloud inputs aren't popular until now. The basics of the Labeled Multi-Bernoulli Filter, which we use for multi target tracking, are explained in the following.

The state x_t^i of the ith target at discrete time t is a random variable. The set of all targets at time step t is a subset of the state space $\mathbb{X}$ and then denoted by

$$\mathbf{X}_t = \left\{x_t^i\right\}_{i=1}^{N_t^x} \subset \mathbb{X}. \tag{1}$$

In turn, the set cardinality $N_t^x = |\mathbf{X}_t|$ at time t is a discrete random variable.

The set of all measurements at time t is again modeled as a random set with set cardinality $N_t^z = |\mathbf{Z}_t|$ and denoted by

$$\mathbf{Z}_t = \left\{z_t^i\right\}_{i=1}^{N_t^z} \subset \mathbb{Z}. \tag{2}$$

Each individual measurement z_t^i is either target-generated or clutter. Yet the true origin is assumed unknown. Further, the set of all measurements up to and including the time step t is denoted by

$$\mathbf{Z}^t = \bigcup_{\tau=1}^{t} \mathbf{Z}_\tau. \tag{3}$$

Both above sets are without order, i.e. the particular choice of indices is arbitrary. Targets and measurements are modeled as Labeled Multi-Bernoulli Random Finite Sets (LMB RFS) as proposed in [2]. A Bernoulli RFS is a set that is either empty with probability $1-r$ or contains a single element. As in [36], the probability density of a Bernoulli RFS may be written as

$$\pi(X) = \begin{cases} 1-r, & \text{if } X = \emptyset, \\ r\,p(x), & \text{if } X = \{x\} \end{cases} \tag{4}$$

with $p(\cdot)$ a spatial distribution on $\mathbb{X}$. A Multi-Bernoulli RFS is then the union of independent Bernoulli RFSs, i.e. $X_{\mathrm{MB}} = \bigcup_i X_{\mathrm{B}}^{(i)}$. In turn a Multi-Bernoulli RFS is well-defined by the parameters $\{r^{(i)}, p^{(i)}\}_i$.

Labeled RFS allow the estimation of both the targets' state and their individual trajectories. For this reason the target state is extended by a label $l \in \mathbb{L}$, i.e. each single target state is given by $\mathbf{x} = (x, l)$ and in turn the multi-target state $\mathbf{X}$ lives on the product space $\mathbb{X} \times \mathbb{L}$ with $\mathbb{L}$ a discrete space. Note that this definition does not enforce the labels l to be distinct. [44] introduced the so called distinct label indicator

$$\Delta(\mathbf{X}) := \delta_{|\mathbf{X}|}(|\mathcal{L}(\mathbf{X})|) \tag{5}$$

that enforces the cardinality of $\mathbf{X}$ to be identical to the cardinality of the projection $\mathcal{L}(\mathbf{X}) = \{\mathcal{L}(\mathbf{x}) : \mathbf{x} \in \mathbf{X}\}$, $\mathcal{L}(\mathbf{x}) = l$. Together with Eq. 4, it follows that the probability density of a LMB RFS is well-defined by the parameter set $\{r^{(l)}, p^{(l)}\}_{l\in\mathbb{L}}$ and the cardinality distribution yields

$$\rho(n) = \prod_{i\in\mathbb{L}}(1-r^{(i)}) \sum_{L\in\mathcal{F}_n(\mathbb{L})} \prod_{l\in L} \frac{r^{(l)}}{1-r^{(l)}} \tag{6}$$

with $\mathcal{F}_n(\mathbb{L})$ the set of all subsets of $\mathbb{L}$ containing n elements.

The core objective of the multi-target tracking is to approximate the multi-target distribution $f_{t|t}(\mathbf{X}_t|\mathbf{Z}^t)$ in each

time step t. This is achieved with the multi-target Bayes filter,

$$f_{t|t}(\mathbf{X}_t|\mathbf{Z}^t) = \frac{f_t(\mathbf{Z}_t|\mathbf{X}_t) f_{t|t-1}(\mathbf{X}_t|\mathbf{Z}^{t-1})}{\int f_t(\mathbf{Z}_t|\mathbf{X}_t) f_{t|t-1}(\mathbf{X}_t|\mathbf{Z}^{t-1}) \delta \mathbf{X}_t} \quad (7)$$

and the Chapman-Kolmogorov prediction

$$f_{t+1|t}(\mathbf{X}_{t+1}|\mathbf{Z}^t) = \int f_{t+1|t}(\mathbf{X}_{t+1}|\mathbf{X}_t) f_{t|t}(\mathbf{X}_t|\mathbf{Z}^t) \delta \mathbf{X}_t \quad (8)$$

with $f_t(\mathbf{Z}_t|\mathbf{X}_t)$ the multi-target measurement set density and $f_{t+1|t}(\mathbf{X}_{t+1}|\mathbf{X}_t)$ the multi-target transition density.

3. Joint Detection and Extended Target Model

3.1. Point Cloud Preprocessing

First, we generate a semantic segmentation map of the front camera images using the efficient model from [39], pre-trained on [5] and fine tuned on KITTI [8]. Second, we quantize the point cloud to a 3D voxel representation, which is able to contain certain features of the points that lie within such a voxel. Our region of interest of the point cloud is set to $[0, 60]m \times [-40, 40]m \times [-2.73, 1.27]m$ in sensor coordinates, according to the KITTI [8] dataset. We chose a resolution of $768 \times 1024 \times 21$ resulting in approximately $0.08m \times 0.08m \times 0.19m$ per cell. Each voxel, where at least one point exists inside its 3D space and is visible to the front camera, is filled with a normalized class value extracted from the semantic map in range $[1, 2]$. Therefore, we project all relevant points into the image using calibrations from [8] and *argmax* over the frequency of all resolved classes. In this way, contextual information with visual features are passed into our voxel map, which is especially helpful for higher ranges with low density of points.

3.2. Voxel based Complex-YOLO

We use the full detection pipeline introduced in [42], but exchange the input map for our voxel representation. Inspired by [34], we exchange max-pooling layers by convolutions with stride 2 and add residual connections. Altogether, we have 49 convolutional layers. Additionally, we add object height h and ground offset z as target regression parameters and incorporate both into the multi-part loss function.

$$\begin{aligned}\mathcal{L} =& \mathcal{L}_{\text{Euler}} \\ &+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[(h_i - \hat{h}_i)^2 + (z_i - \hat{z}_i)^2 \right]\end{aligned} \quad (9)$$

Usually IoU is used to compare detection and ground truth during the training process. However, it has drawbacks when comparing rotated bounding boxes. If two boxes are compared with the same size and position and an angle difference of π the IoU between these two boxes is 1, which means they match perfectly. This is obviously not the case since the angle between the two boxes has the maximum difference it can have. So while training a network it is not penalized and even encouraged by predicting boxes like these. This leads to wrong predictions for the object orientation. Also calculating an exact IoU for rotated bounding boxes in 3D space is a time consuming task.

To overcome these two problems we introduce a new highly parameterizable simple evaluation metric called Scaling-Rotation-Translation score (*SRTs*). The *SRTs* is based on the fact, that given two arbitrary 3D objects of the same shape, one can be transformed into the other using a transformation. Therefore, we can define a score S_{srt} as composite of independent scores for scaling S_s, rotation S_r and translation S_t with

$$S_s = 1 - \min\left(\frac{|1 - s_x| + |1 - s_y| + |1 - s_z|}{w_s}, 1 \right) \quad (10)$$

$$S_r = \max\left(0, 1 - \frac{\theta}{w_r \pi}\right), \quad w_r \in (0, 1] \quad (11)$$

$$r_i = \frac{d_i \cdot w_t}{2}, \quad i \in \{1, 2\} \quad (12)$$

$$S_t = \max\left(0, \frac{r_1 + r_2 - t}{r_1 + r_2}\right) \quad (13)$$

$$p_t = \begin{cases} 0, & \text{if } r_1 + r_2 < t \\ 1, & \text{otherwise} \end{cases} \quad (14)$$

where $s_{x,y,z}$ denotes size ratios in x, y, z directions, θ denotes the difference of the yaw angles, t the Euclidean distance between the two object centers and p_t is a penalty if the objects do not intersect. S_t is calculated in respect to the size of the two objects, because for small objects a small translation can already have a big impact and vice versa for large objects. So the length of the diagonals d_i of both objects are used to calculate two radii r_i.

To adjust the score w_s, w_t and w_r can be used. They control how strict the individual scores are. We used $w_s = 0.3$, $w_t = 1$ and $w_r = 0.5$

All the previous scores are in the interval $[0, 1]$ and can be combined into the final score (S_{srt}) using a simple weighted average and the penalty p_t.

$$\begin{gathered} S_{srt} = p_t \cdot (\alpha\, S_s + \beta\, S_t + \gamma\, S_r) \\ \alpha + \beta + \gamma = 1 \end{gathered} \quad (15)$$

Using α, β, γ the weight of the three sub scores can be defined. We used $\gamma = 0.4$ and $\alpha = \beta = 0.3$ to give more

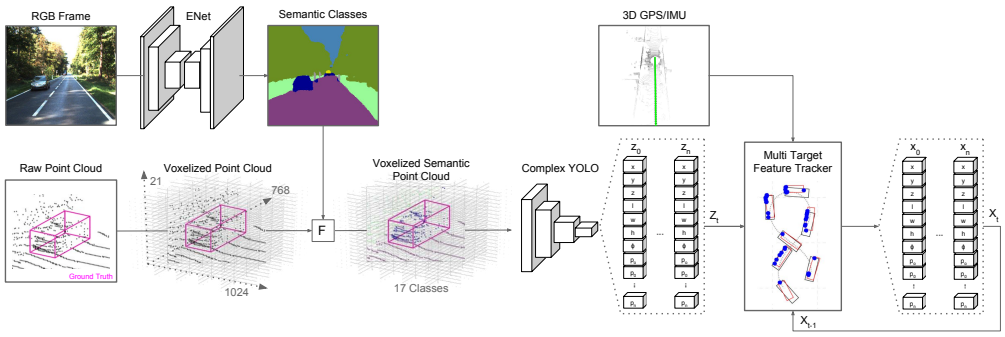

Figure 2: Overview of our architecture.

weight to the angle, because translation and scaling are easier to learn for the network.

SRTs perfectly lines up with the three subtasks (rotation, position, size) a network has to do in order to predict 3D boxes with a yaw angle. It is designed so it can be parametrized to approximate the IoU but considers object orientations. Using all the parameters the score can be adjusted to suit the needs of the problem.

3.3. Extended target model in the LMB RFS

To apply the RFS approach to the output of the YOLO network, which consists of boxes in the three dimensional space, we interpret the output as Gaussian noise corrupted measurements z_t^i, $i \in \{1, \ldots, N_t^z\}$ of the positional parameters (see Eq. 2) and extend those as extended targets [9] x_t^i, $i \in \{1, \ldots, N_t^x\}$ (see Eq. 1) with the measurement noise covariance matrix $R = \text{diag}(0.5^2, 0.5^2, 0.5^2, 0.5^2, 0.5^2, 0.1^2)$. The target is assumed to move according to a coordinated turn motion model [38] with the process noise covariance $Q = \text{diag}(\sigma_a^2, \sigma_\alpha^2)$ consisting of the standard deviation of the acceleration $\sigma_a = 17.89$ and the yaw rates derivative $\sigma_\alpha = 1.49$.

The individual measurements z consist of the box center position $c = [x, y, z]$ in the three dimensional space, the box dimension (length, width, height) $s = [l, w, h]$ and the box orientation along the first dimensions ϕ_t^i (yaw), such that

$$z = [c, s, \phi]\,. \tag{16}$$

The extended target state mean $\bar{x}_t^i$ used for tracking contains the same parameters as the measurements as well as motion parameters of the coordinated turn model consisting of the velocity v and yaw rate $\dot{\phi}$. The state mean of the ith target at time t can be described as

$$\bar{x}_t^i = [c_t^i, s_t^i, \phi_t^i, v_t^i, \dot{\phi}_t^i] \tag{17}$$

with the according state covariance matrix $\bar{P}_t^i$. We can state the measurement equation

$$z = H \cdot \bar{x} \tag{18}$$

for an individual measurement z and the target mean $\bar{x}$, with the measurement matrix

$$H = \begin{pmatrix} \mathrm{I}_7 & 0 \end{pmatrix} \in \mathbb{R}^{7\times 9}, \tag{19}$$

where I_7 is the identity matrix of dimension 7. Based on this measurement equation, a Bayesian filter can be defined where the innovation is calculated using a Kalman filter update according to the stated measurement model. The prediction is performed using an Unscented Kalman filter according to the assumed nonlinear coordinated turn model [38].

In the LMB update step each predicted target is associated with each measurement of the time step and an update according to the defined measurement model is performed. A heatmap is generated from the update likelihood, modeling the association probabilities based on which targets will be kept or discarded. Be $p_a(x^i, z^j)$ the association probability of the measurement z^j and the state x^i. If the non-assignment probability

$$p_{na}(z^j) = 1 - \sum_{x^i \in \mathbf{X}} p_a(x^i, z^j) \tag{20}$$

is higher than a threshold P_{na}, we assume that a new target is born from an unexplained measurement.

The number N_e of targets to be extracted is derived from the mean of the cardinality distribution presented in Eq. 6. All N_e targets with the highest existence probability $r^{(l)}$ are extracted.

4. Experiments

We evaluate Complexer-YOLO on the KITTI benchmarks for 3D object detection and bird's eye view (BEV) detection. Furthermore, we evaluate the capabilities of our multi target tracking with the help of the object tracking

benchmark. Our ablation studies investigate the importance of different input features encoded in our voxel representation and show further findings. Finally, some qualitative results visualize the outcome of our model.

4.1. Training Details

The KITTI dataset [8] consists of 7481 training images and 7518 testing images. First, we follow [4] and use the training/validation split to optimize our settings. Afterwards, we use the full training set for the official evaluation. We augment the training dataset with rotation and increase the size by a factor of 4. Therefore, we randomly pick 3 angles between $[-20, 20]$ deg with a minimum difference of 8 deg to each other. Similar to [47], we use random flipping along the x axis during training.

For training, we use an extended version of the darknet framework [31]. We train the model from scratch for 140k iterations with learning rate scaled at 20k, 80k and 120k iterations respectively.

4.2. Detection Results

We submitted our results to the KITTI vision benchmark suite [8] for Orientation Similarity, BEV, 3D Object Detection and Object Tracking benchmarks on the official test set. To achieve a fair comparison, we only selected some of the leading 3D object detectors that are able to detect at least classes *Car*, *Pedestrian* and *Cyclist*. For tracking, only online methods are listed.

We show evaluation results for Orientation Similarity, BEV and 3D object detection in Table 1. Table 2 shows our results in MOT accurancy and precision (MOTA and MOTP), mostly tracked (MT) and mostly lost (ML).

Unfortunately, the whole evaluation process is based on 2D bounding boxes in camera space due to the handling of *Dontcare* labels and ignored objects, e.g. truncated or occluded (see [8]). Following the 2D Object Detection Benchmark, which is accompanied with BEV and 3D, we achieve 79.31% for class *Car* in *moderate* difficulty. Also, Orientation for these settings is ranked at 79.08%. However, our algorithm detects and tracks bounding boxes in 3D space. Therefore, all detections are projected to the image plane. Although we do not track in image space, we achieve state-of-the-art results while running in real-time using our tracking (visualization Fig. 6). Moreover we are the first one with 3D tracking based on point cloud detections on the KITTI tracking benchmark. But there is an inconsistency compared to BEV and 3D results, where we achieve only 66.07% and 49.44% respectively. Based on less than 50% AP in 3D space, tracking is not able to reach actual results, which we think mainly comes from wrongly counted *Dontcare* objects. In opposition to the KITTI guidelines, we found that their current object detection evaluation scripts fully ignore *Dontcare* labels for BEV and 3D Object detection benchmarks. All such detections count as false positives, which is crucial in our case. Furthermore, most 2D ground truth bounding boxes for class *Pedestrian* are manually refined and do not match a reprojected bounding box from 3D space anymore, which leads to additional wrongly counted false positives, when ignored objects are assigned.

Fig. 3 shows outstanding results on several sequences with different use cases. Our model is able to detect accurate rotated bounding boxes in 3D space for multiple classes even though the strongly unbalanced dataset. With the help of voxelized semantic features, the network is able to detect even small objects like pedestrians or cyclist, as long as they have a minimum spatial distance to other appearing objects.

Method	FPS	Car			Pedestrian			Cyclist		
		Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
Orientation										
AVOD-FPN [16]	10.0	**89.95**	**87.13**	**79.24**	**53.36**	**44.92**	**43.77**	67.61	**57.53**	54.16
SECOND [46]	**20.0**	87.84	81.31	71.95	51.56	43.51	38.78	**80.97**	57.20	55.14
BirdNet [1]	9.1	50.85	35.81	34.90	21.34	17.26	16.67	41.48	30.76	28.66
Complexer-YOLO	15.6	87.97	79.08	78.75	37.80	31.80	31.26	64.51	56.32	**56.23**
BEV										
F-PointNet [28]	5.9	88.70	**84.00**	75.33	58.09	50.22	47.20	**75.38**	**61.96**	**54.68**
AVOD-FPN [16]	10.0	88.53	83.79	**77.90**	**58.75**	**51.05**	**47.54**	68.09	57.48	50.77
VoxelNet [49]	4.4	**89.35**	79.26	77.39	46.13	40.74	38.11	66.70	54.76	50.55
BirdNet [1]	9.1	75.52	50.81	50.00	26.07	21.35	19.96	38.93	27.18	25.51
Complexer-YOLO	**15.6**	74.23	66.07	65.70	22.00	20.88	20.81	36.12	30.16	26.01
3D										
F-PointNet [28]	5.9	81.20	70.39	62.19	**51.21**	**44.89**	40.23	**71.96**	**56.77**	**50.39**
AVOD-FPN [16]	10.0	**81.94**	**71.88**	**66.38**	50.80	42.81	**40.88**	64.00	52.18	46.61
VoxelNet [49]	4.4	77.47	65.11	57.73	39.48	33.69	31.51	61.22	48.36	44.37
BirdNet [1]	9.1	14.75	13.44	12.04	14.31	11.80	10.55	18.35	12.43	11.88
Complexer-YOLO	**15.6**	55.63	49.44	44.13	19.45	15.32	14.80	28.36	23.48	22.85

Table 1: Evaluation of orientation, bird's eye view and 3D detection. Frames per second (FPS) and APs (in %) on KITTI test set.

Method	MOTA [%]	MOTP [%]	MT [%]	ML [%]	FPS
MOTBeyondPixels [41]	**84.24**	**85.73**	**73.23**	**2.77**	3.3
IMMDP [41]	83.04	82.74	60.62	11.38	5.3
3D-CNN/PMBM [40]	80.39	81.26	62.77	6.15	**100**
mbodSSP [18]	72.69	78.75	48.77	8.77	**100**
Ours	75.70	78.46	58.00	5.08	**100**

Table 2: Comparison with non-anonymous pure online submissions on KITTI MOT benchmark [8].

4.3. Ablation Study

We conducted ablation experiments with fixed training setup to investigate the influence of our hyper parameters and several input features. The use of 21 height channels for our voxel map results in similar mAP at IoU threshold 0.7 as using 51 channels (cuboidal voxels). It seems that our network is not able to fully utilize fine grained height information. Furthermore, it is the best trade-off for runtime and accuracy, because runtime was slightly increasing

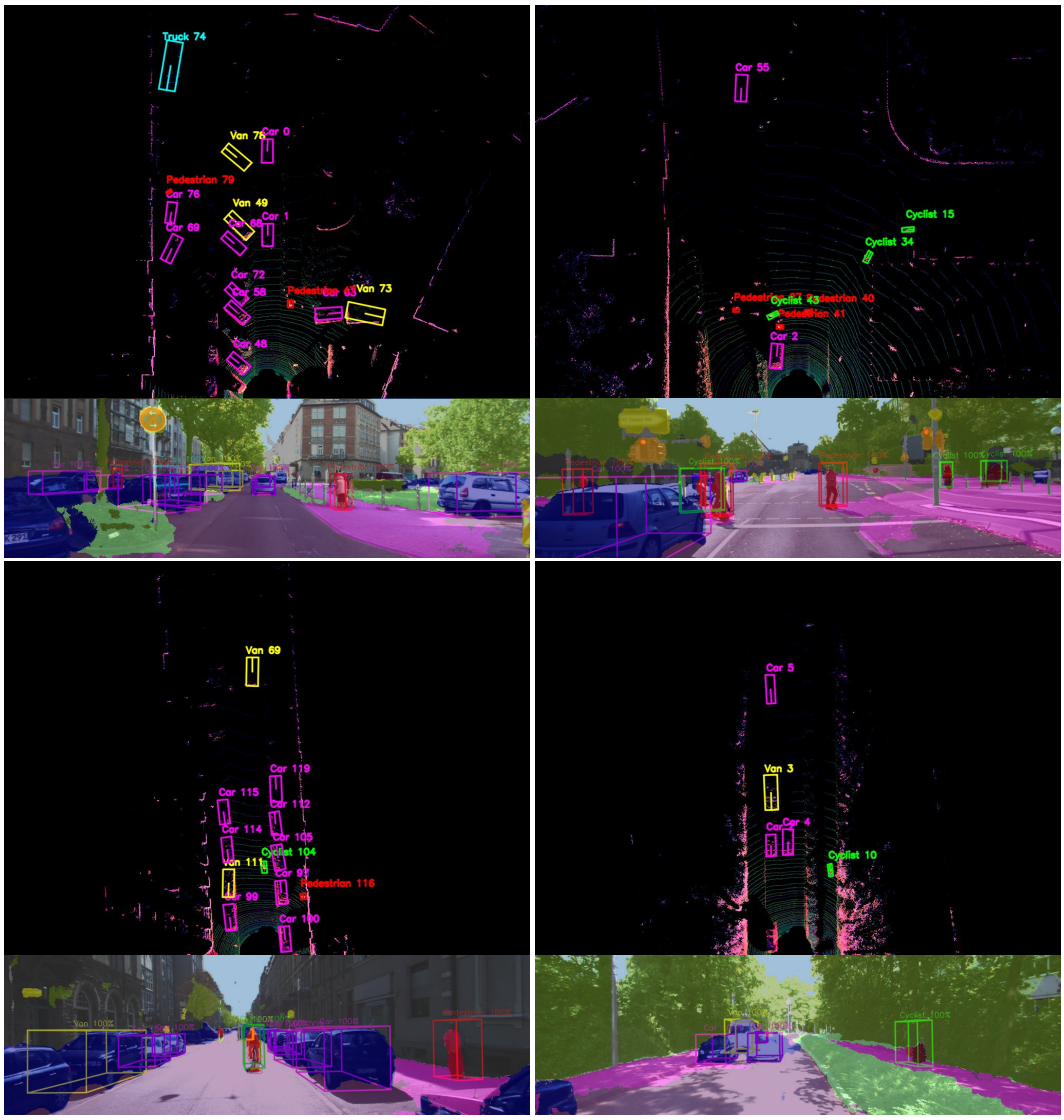

Figure 3: Qualitative results. For visualization, our detections are projected into camera space with overlayed pixel wise semantic segmentation.

for more than 21 height channels with our hardware setup. Table 3 shows results using different voxel maps with intensity values from Lidar sensor normalized in range $[1, 2]$, binary occupancy similar to [25] and our novel semantic map. Additionally, the approach from [42] using extracted features encoded as an RGB image is listed.

In order to reduce wrongly counted false positives due to ignored *Dontcare* labels, we tried to filter our detections in a post processing step. Therefore, we counted the number of 3D points falling into each 3D bounding box. All detections with less then 13 points and less than $52m$ radial distance to the Lidar sensor were removed, because *Dontcare* is often used for objects at higher distances or occluded objects. This improved all object detection results by 1.3% on average, but decreased e.g. BEV for *Car* on *moderate* difficulty by 4.8%. Consequently, our filter removed a few *Dontcare* or ignored detections, but removed correct ones as well. Also, it seemed to have stronger impact on *moderate* settings since valid *easy* detections are all in near range, which explains AP drops from *easy*.

Finally, using *SRTs* for training instead of IoU gives 1.3% improvement on mAP at IoU 0.7 as it directly penalizes orientation. It also halved our training time and resulted in a 10-20% runtime improvement for inference.

Additionally, we tried to limit object rotations into subsections using anchors instead of complex angles for the full $360\,\mathrm{deg}$, but this decreased accuracy. Further investigation is required here, because we see a potential reduction

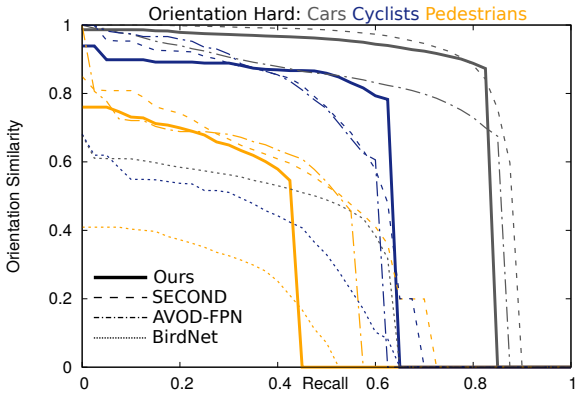

Figure 4: Results for the orientation benchmark compared to SECOND [46], BirdNet [1], AVOD-FPN [16] on official KITTI test set.

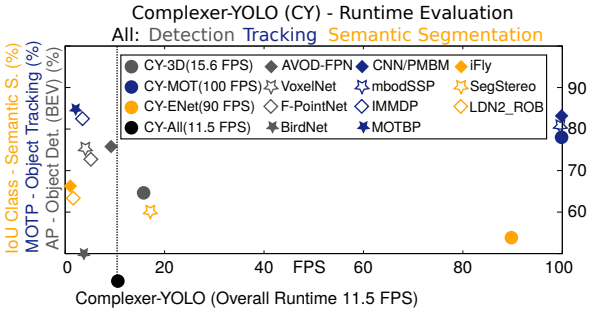

Figure 5: The Complexer-YOLO runtime evaluation (reference hardware: NVIDIA GTX1080i/Titan) shows state-of-the-art results of all single tasks (Semantic Segmentation, 3D object detection (BEV, cars → hard, Tab. 1), Tracking Tab. 2). Results for Semantic Segmentation are taken from the KITTI leader board. We point out, that our overall Detection and Tracking Pipeline is faster than many single task algorithms.

in complexity for the learning task of the network.

Feature	IoU 0.7	SRTs 0.7
RGB	28.64	30.02
Occupancy	31.93	33.24
Intensity	32.39	33.57
Semantic	34.14	35.43

Table 3: Ablation study of different input features. mAP values (in %) for the 3D benchmark on KITTI validation set.

5. Conclusion

In this work we propose Complexer-YOLO, a tracked real-time 3D object detector that operates on point clouds fused with visual semantic segmentation. Our architecture

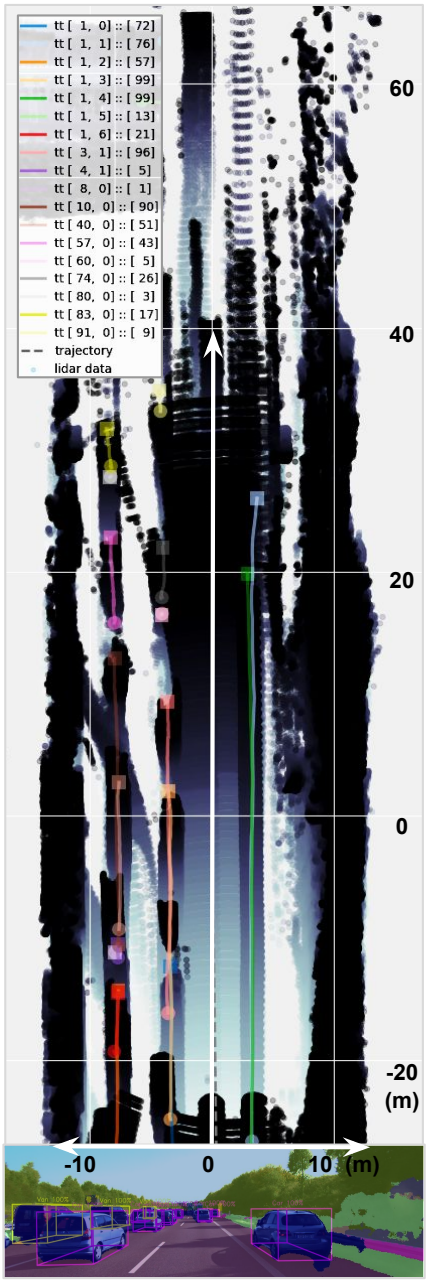

Figure 6: Tracked objects trajectories.

takes advantage of both spatial Lidar data and explored scene understanding from 2D. Detection results obtained from 3D space show competitive performance on KITTI benchmarks [8] compared to state-of-the-art. At the same time, we introduce *SRTs*, a powerful, more flexible and simplified evaluation metric for object comparison that overcomes the limits of IoU.

References

[1] J. Beltran, C. Guindel, F. M. Moreno, D. Cruzado, F. Garcia, and A. de la Escalera. BirdNet: a 3D Object Detection Framework from LiDAR information. 2018. 3, 6, 8

[2] D. S. Bryant, B. T. Vo, B. N. Vo, and B. A. Jones. A Generalized Labeled Multi-Bernoulli Filter with Object Spawning. pages 1–12, 2017. 3

[3] X. Chen, K. Kundu, Y. Zhu, H. Ma, S. Fidler, and R. Urtasun. 3D Object Proposals using Stereo Imagery for Accurate Object Class Detection. aug 2016. 2

[4] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3D object detection network for autonomous driving. *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, pages 6526–6534, nov 2017. 3, 6

[5] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. 2016. 4

[6] 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. 2016. 2

[7] D. Frossard and R. Urtasun. End-to-end Learning of Multi-sensor 3D Tracking by Detection. 2018. 3

[8] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 128, 2012. 4, 6, 8

[9] K. Granstrom, M. Baum, and S. Reuter. Extended object tracking: Introduction, overview and applications. *ISIF Journal of Advances in Information Fusion*, 12(2), Dec. 2017. 5

[10] A. Graves, S. Fernandez, and J. Schmidhuber. Multi-Dimensional Recurrent Neural Networks. 2007. 3

[11] K. He. Deep Residual Learning for Image Recognition. 2015. 2

[12] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask R-CNN. *Proc. IEEE Int. Conf. Comput. Vis.*, 2017. 2

[13] Y. He, X. Zhang, and J. Sun. Channel Pruning for Accelerating Very Deep Neural Networks. 2017. 3

[14] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. MobileNets. 2017. 3

[15] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. 2016. 2

[16] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander. Joint 3D Proposal Generation and Object Detection from View Aggregation. 2017. 3, 6, 8

[17] B. Lee, E. Erdenee, S. Jin, and P. K. Rhee. Multi-class multi-object tracking using changing point detection. aug 2016. 3

[18] P. Lenz, A. Geiger, and R. Urtasun. FollowMe: Efficient Online Min-Cost Flow Tracking with Bounded Memory and Computation. 2014. 6

[19] B. Li. 3D Fully Convolutional Network for Vehicle Detection in Point Cloud. 115, 2016. 2

[20] B. Li, T. Zhang, and T. Xia. Vehicle Detection from 3D Lidar Using Fully Convolutional Network. 2016. 3

[21] G. Lin, A. Milan, C. Shen, and I. Reid. RefineNet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation. In *CVPR 2017*, 2017. 3

[22] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature Pyramid Networks for Object Detection. dec 2016. 2

[23] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal Loss for Dense Object Detection. *Proc. IEEE Int. Conf. Comput. Vis.*, 2017. 2

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. 2016. 2

[25] W. Luo, B. Yang, and R. Urtasun. Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting with a Single Convolutional Net. 2018. 3, 7

[26] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi. ESPNet : Efficient Spatial Pyramid of Dilated Convolutions for Semantic Segmentation. 2018. 3

[27] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet : A deep neural network architecture for real-time semantic segmentation. 2016. 3

[28] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum PointNets for 3D Object Detection from RGB-D Data. 2017. 3, 6

[29] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. 74, 2016. 3

[30] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. XNOR-net: Imagenet classification using binary convolutional neural networks. 2016. 3

[31] J. Redmon. Darknet: Open source neural networks in c. http://pjreddie.com/darknet/, 2013–2016. 6

[32] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. *Proc. IEEE Int. Conf. Comput. Vis.*, 2015. 2

[33] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. *Eur. Pharm. Contract.*, dec 2016. 2

[34] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. 2018. 2, 4

[35] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 2

[36] S. Reuter, B.-T. Vo, B.-n. Vo, and K. Dietmayer. The labeled multi-bernoulli filter. *IEEE Trans. Signal Process.*, 2014. 3

[37] E. Romera, J. M. lvarez, L. M. Bergasa, and R. Arroyo. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, Jan 2018. 3

[38] M. Roth, G. Hendeby, and F. Gustafsson. Ekf/ukf maneuvering target tracking using coordinated turn models with polar/cartesian velocity. In *Information Fusion (FUSION), 2014 17th International Conference on*, pages 1–8. IEEE, 2014. 5

[39] T. Sämann, K. Amende, S. Milz, C. Witt, M. Simon, and J. Petzold. Efficient Semantic Segmentation for Visual Bird ' s-eye View Interpretation. 2018. 1, 3, 4

[40] S. Scheidegger, J. Benjaminsson, E. Rosenberg, A. Krishnan, and K. Granstrom. Mono-Camera 3D Multi-Object Tracking Using Deep Learning Detections and PMBM Filtering. 2018. 3, 6

[41] S. Sharma, J. A. Ansari, J. K. Murthy, and K. M. Krishna. Beyond Pixels: Leveraging Geometry and Shape Cues for Online Multi-Object Tracking. 2018. 3, 6

[42] M. Simon, S. Milz, K. Amende, and H.-M. Gross. Complex-YOLO: Real-time 3D Object Detection on Point Clouds. 2018. 1, 3, 4, 7

[43] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. *Proc. IEEE Int. Conf. Comput. Vis.*, 2015 International Conference on Computer Vision, ICCV 2015:4489–4497, 2015. 2

[44] B.-t. Vo and B.-n. Vo. A Random Finite Set Conjugate Prior and Application to Multi-Target Tracking. 2011. 3

[45] D. Xu, D. Anguelov, and A. Jain. PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation. 2017. 3

[46] Y. Yan, Y. Mao, and B. Li. SECOND: Sparsely Embedded Convolutional Detection. *Sensors*, 18(10):3337, 2018. 6, 8

[47] B. Yang, W. Luo, and R. Urtasun. PIXOR: Real-time 3D Object Detection from Point Clouds. *Cvpr 2018*, pages 7652–7660, 2018. 3, 6

[48] F. Yu and V. Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. 2016. 3

[49] Y. Zhou and O. Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. *CVPR*, 2017. 3, 6