2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

MB-Net: MergeBoxes for Real-Time 3D Vehicles Detection

Nils Gählert[1], Marina Mayer[2], Lukas Schneider[3], Uwe Franke[3] and Joachim Denzler[4]

***Abstract*— High performance vehicle detection and pose estimation in RGB images is essential for driver assistance systems as well as for autonomous vehicles. Classical 2D box-based detection schemes allow roughly estimating the position of other vehicles, but not their orientation relative to the ego-vehicle. Recent approaches use 3D models to derive the pose of other vehicles from single monocular images but do not reach real-time performance. In this paper we present an approach that achieves competitive performance on the challenging KITTI Object Detection and Orientation Estimation benchmark while being the fastest approach with over 40 FPS. The key is a novel representation named *MergeBox* whose parameters can be estimated extremely efficiently. We extend SSD – a current fast state-of-the-art 2D box object detector – with this representation to our *MB-Net*. In contrast to all other current state-of-the-art methods we do not require explicit information on the object orientation for training our model. This reduces label costs significantly, a further advantage for practical applications that require labeling of databases that are much bigger than those used for research.**

I. INTRODUCTION

Autonomous driving has gained attention both from industrial and academic research facilities within the last few years. To provide maximum safety and comfort for all passengers and other road users, a detailed knowledge of the current state of a self-driving car and the surroundings is crucial. Hence, perceiving the environment is an important task to be solved. This problem can be best addressed with a set of different sensors, such as LiDAR, RADAR and one or more RGB cameras, whereas each type of sensor has advantages for particular tasks. While e.g. LiDAR provides very accurate information about the distance of a 3D point, it yields a sparse representation of the scene and only little semantic information is encoded. Besides that, LiDAR sensors are relatively expensive. In contrast, RGB cameras provide a very dense representation and hence can be used to obtain rich semantic information. As they are quite cheap, most consumer vehicles are already equipped with at least one RGB camera. Finally, combining the data of these sensors allows for a complete scene representation of the car's surrounding environment. Accordingly, we aim to obtain as much information as possible from each sensor individually to achieve best possible performance. This paper focuses on the data acquisition of single monocular images.

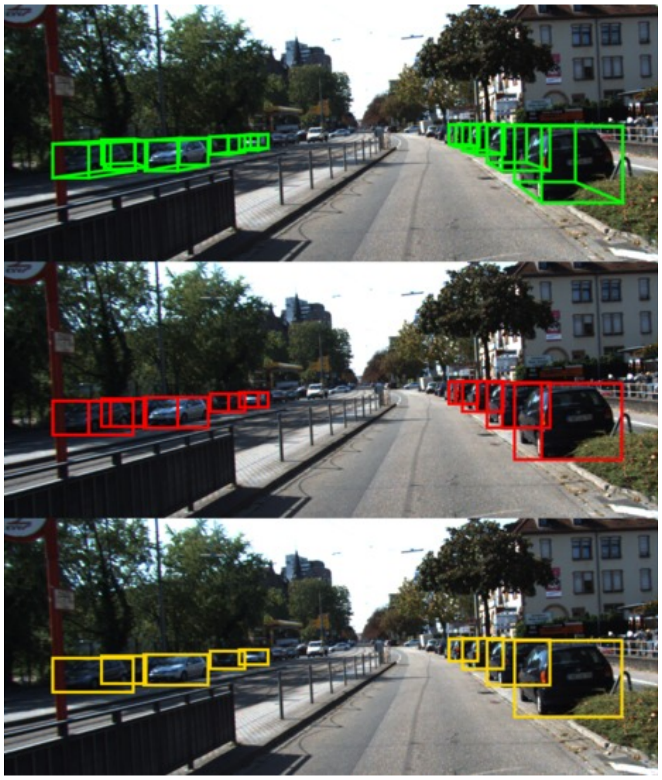
Fig. 1. Standard 2D bounding box **(bottom)**. Important information such as the orientation of a vehicle is not encoded in this representation. Using the MergeBox representation **(center)** 3D bounding boxes **(top)** can be generated. Image taken from the KITTI [1] *test* set.

One main challenge that needs to be tackled for autonomous driving is object detection – currently this is typically addressed by drawing the minimum 2D bounding box around a particular object. This 2D object detection approach is a well studied problem with convolutional neural networks (CNNs). However, by using a simple 2D bounding box, several important geometric properties are lost, e.g. if a self-driving car needs to overtake another car, it is important to know its length. To better understand the intention of other vehicles it is furthermore helpful to estimate their 3D orientation, as this determines the possible trajectories of these vehicles. Again, this information is missing if an object is described by a 2D bounding box.

Hence, in this paper we seek to find a solution to obtain this missing information and mainly highlight the following contributions: (1) This paper introduces a novel scheme for vehicle detection that allows to estimate real 3D outlines

[1]Daimler AG, R&D, Germany and University of Jena, Computer Vision Group, Jena, Germany `nils.gaehlert@daimler.com`
[2]Daimler AG, R&D, Germany and Hochschule für Technik, Stuttgart, Germany `marina.mayer24@gmail.com`
[3]Daimler AG, R&D, Germany `lukas.schneider@daimler.com`
[3]Daimler AG, R&D, Germany `uwe.franke@daimler.com`
[4]University of Jena, Computer Vision Group, Jena, Germany `joachim.denzler@uni-jena.de`

by means of bounding boxes in the 3D scene. To this end, a novel *MergeBox* representation is proposed that can be inferred directly from a single monocular image using arbitrary object detection frameworks such as SSD [2] or Faster R-CNN [3]. (2) Despite its ability to estimate a 3D outline including size and orientation, the proposed method does not require orientation annotations during training. Instead, only three clicks are necessary for the annotation of a vehicle. This is an inalienable basis for the creation of large datasets. (3) Finally, this is the first method on the challenging KITTI Object Detection and Orientation Estimation benchmark [4] that runs in real-time while being competitive with other leading approaches.

II. RELATED WORK

Methods that address simultaneous object detection and orientation estimation can be separated into two different streams of research: (a) extending a 2D box approach by additionally learning both orientation as well as the object's dimensions and (b) direct prediction of the bounding cuboid in 3D space.

A. Learning Orientation and Dimensions

In this first category a classical 2D bounding box detector is extended to estimate orientation as well as dimensions. As these detectors are usually based on CNNs, typically the viewpoint (cf. IV-A) is predicted, as convolutional layers are translationally invariant.

[5] use shared CNN features and additional fully connected layers for dimensions prediction as well as for classification and viewpoint estimation. To this end, a novel MultiBin loss is introduced that discretizes the 360° range into n bins to perform classification as well as regression of the residual angle. Discretization in combination with classification for orientation estimation is also used in [6]–[8]. [9] clusters different visual features of the objects such as viewpoint, etc. and uses an AdaBoost scheme for detection. Von Mises loss in combination with biternion representation is used in [10] to estimate the viewpoints of objects.

Despite their ability to predict an orientation in addition to sole location, we opt for an approach that directly estimates a 3D cuboid encoding all the important information directly: location, dimensions and the true orientation in 3D instead of a location dependent viewpoint.

B. Direct Prediction of 3D Bounding Boxes

Instead of learning the orientation and the dimensions in addition to a 2D bounding box detection, it is possible to directly predict the 3D bounding box. This second category can be further split into two separate groups. The first stream of research employs depth information. [11]–[13] make use of the available LiDAR data to gain information about the depth in addition to a monocular RGB image. 3D Voxel Patterns – abstract representations of 3D objects encoding different properties such as visibility of an object, truncation, etc. – are introduced in [14]. Fitting an L-Shape into a 3D LiDAR point cloud was shown to work efficiently in [15]

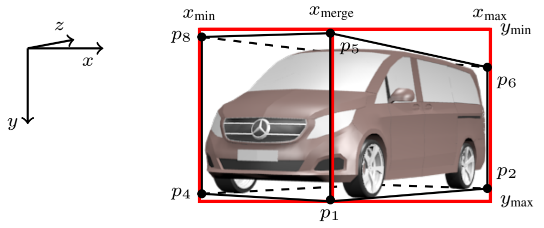

Fig. 2. Nomenclature we use for the 3D bounding box with vertices p_i and the MergeBox representation x_i and y_i in the image plane.

and [16]. We also aim for object detection by means of 3D cuboids, however, we base on monocular image cues only.

Accordingly, the second and most related group of methods is formed by such monocular approaches. [17], [18] build part models with several keypoints that are linked to particular parts in the car to obtain a very detailed model of the vehicle and its shape. In contrast, [19]–[21] focus on efficient 3D proposal generation as part of the object detection pipeline.

In this paper, we propose a system that combines the speed of 2D object detection pipelines with the information gained from 3D bounding boxes. In contrast to all 2D methods we neither learn the viewpoint nor the orientation of the object of interest explicitly. Instead, we are able to calculate the object's orientation in a closed form. This has two main advantages: First, it reduces the number of parameters of our model resulting in a fast inference. Second, it simultaneously relaxes the requirements needed for ground truth data (GT) used to train the model. While GT generation of 3D orientation is expensive – e.g. as external data like LiDAR is required or by manually adjusting a CAD model to fit the projected object – we only require a simple 2D bounding box with one single additional parameter that is fast and easy to annotate in addition to the class of the object, i.e. which combination of front/back/side is visible.

III. MODEL

We aim to combine the advantages of 3D bounding boxes – e.g. the knowledge of dimensions and the real 3D orientation – with the speed of a 2D bounding box detector. To this end, we create a new representation which we call *MergeBox*. As shown in Fig. 2 it mainly consists of a 2D bounding box with an additional parameter at $x = x_{\text{merge}}$ that should indicate the projected edge between the front or the back of a vehicle and its side. We therefore reduce the number of required parameters to describe the 3D bounding box from 9 (3× translation, 3× orientation, 3× dimension) to 5 $(x_{\text{min}}, x_{\text{merge}}, x_{\text{max}}; y_{\text{min}}, y_{\text{max}})$. To obtain this representation of a vehicle, we train a specialized neural network named *MB-Net*. The final 3D bounding box can be deduced from the MergeBox representation. We first calculate the translation of the final 3D bounding box followed by a scaling in 3D and the calculation of the orientation. We do this for different templates t to describe several types of vehicles (compact, sedan, van, etc.) and finally select the template t whose projection best fits into the MergeBox.

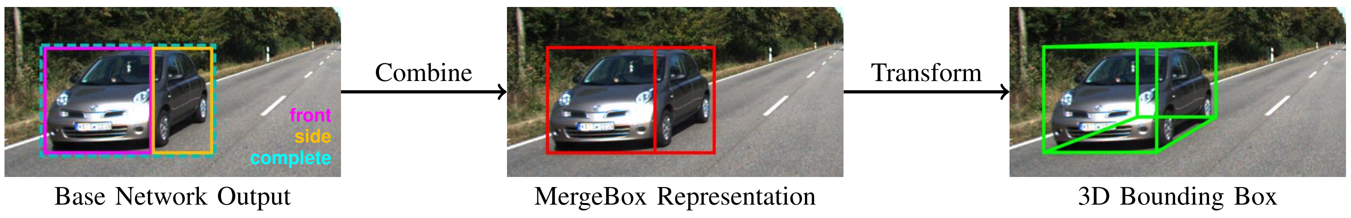

Fig. 3. Our Detection pipeline: First, we detect different sides of the car as well as the complete car **(left)**. The detections for front and side of the vehicle are shown in magenta and yellow, respectively. The complete vehicle detection is drawn in cyan. Combining these boxes results in the MergeBox representation **(center)** and the corresponding 3D bounding box **(right)**. Best viewed in color.

A. *MergeBox Representation*

Performing a simple 2D bounding box detection task is well understood and can be implemented very efficiently and fast. Current state-of-the-art object detectors can be grouped into two different architectures: (i) one stage detectors such as SSD [2], Yolo [22] and Yolo9000 [23] and (ii) two stage detectors such as Faster R-CNN [3]. Due to their design, one stage detectors are extremely fast as both localization and classification is done within one network pass. Two stage detectors, in contrast, typically generate more accurate results but require more computation time.

Our goal is to design a bounding box representation that can independently be used with all different object detection frameworks.

The idea of the MergeBox representation is that an additional parameter x_{merge} is added to a standard 4 degrees of freedom (DoF) bounding box, where x_{merge} represents the edge between the front and a side or the back and a side of a vehicle, respectively. Fig. 2 shows an exemplary MergeBox with its 5 DoF $(x_{\min}, x_{\text{merge}}, x_{\max}; y_{\min}, y_{\max})$.

To this end, we can use any 2D bounding box detector and train it to classify 4 different classes: (i) front of a vehicle, (ii) back of a vehicle, (iii) side of a vehicle and (iv) complete vehicle.

Ideally, we detect three 2D boxes per object and combine them to build a MergeBox: front/back, a side and a complete vehicle as shown in Fig. 3. However, if there are only two detections per object, it is still possible to create a MergeBox for the following cases:

- A front/back and a side is detected, but no complete vehicle.
- A complete vehicle is detected and one part of the vehicle (i.e. front, back or side) fully covers the complete vehicle detection, i.e. the vehicle is moving in the direction $\pm\theta_{\text{ray}} \pm 90°$ with θ_{ray} being the angle of the visual ray through the object's center.
- A complete vehicle is detected and only one part of the vehicle (i.e. front, back or side) that partly covers the detection. In this case we know that the other part of the vehicle was not detected. However, to a certain degree we still can derive the MergeBox representation as we can fill up the missing space with the adjacent class.

As the assignment only requires minimal additional computing after any standard 2D object detection pipeline it can efficiently be used to represent the 3D vehicle with only 5 parameters in addition to the classification result of the object detector.

TABLE I

TEMPLATES USED TO DERIVE THE 3D BOUNDING BOX.

Template t	Length l_t	Width w_t	Height h_t
Compact	3.50 m	1.60 m	1.50 m
Sedan	5.10 m	1.90 m	1.45 m
Estate Car	4.70 m	1.80 m	1.45 m
SUV	4.90 m	2.00 m	1.70 m
Van	4.90 m	1.85 m	2.00 m
Large Van	6.50 m	1.95 m	2.50 m

B. *Deriving the 3D Bounding Box*

After creating a 5 DoF MergeBox representation, a 3D bounding box can easily be derived. To this end, we start with an initial 3D cuboid and adjust it by moving and rotating it such that the projections of its vertices p_i fit into the MergeBox with its parameters (x_i/y_i) as shown in Fig. 2.

However, as we only add one additional parameter (x_{merge}) it is not possible to obtain the full 3D orientation (yaw, pitch, roll) and the full dimensions (length, width, height) as this would require 6 additional parameters. Instead, we analyzed the ground truth of the KITTI dataset [1] and built templates of different car types that we use to derive the vehicle's dimension jointly with its orientation, i.e. its yaw angle θ. These initial bounding box dimensions $(l_t/w_t/h_t)$ for template t are given in Tab. I. We show in IV-F that the actual choice of templates is not critical for the accuracy of the proposed approach.

As we use depth-normalized parameters, we can assume that the vehicle of interest is standing on the ground plane. For each template t we perform 3 steps to derive the parameters for the 3D bounding box and finally select the best one:

1) **Translation.** To localize the final 3D bounding box, we calculate the translation δ_x and δ_z in x- and z-axis – in the used coordinate system the x-axis is pointing to the right and the z-axis is aligned with the roll axis of the ego-vehicle. To this end, we enforce the point closest to the camera p_1 of all templates t to be projected into the image plane at $(x_{\text{merge}}/y_{\max})$ using the projection

matrix A by

$$\begin{pmatrix} x_{\text{merge}} \\ y_{\text{max}} \\ 1 \end{pmatrix} \overset{!}{=} A \left[T_\delta \; p_1^{(t)} \right]. \tag{1}$$

We use homogeneous coordinates and represent the translation $(\delta_x, 0, \delta_z)^T$ by the homogeneous transformation matrix T_δ.

2) **Scaling.** Our initial bounding box with dimensions of template t must be scaled by a factor k such that its projected height matches the detected MergeBox height $(y_{\text{max}} - y_{\text{min}})$ by

$$y_{\text{min}} \overset{!}{=} \left(A \left[T_\delta \; S_y^{(k)} \; p_5^{(t)} \right] \right)_y \tag{2}$$

with $S_y^{(k)}$ being the homogeneous scale transformation matrix for a scaling in y-direction with factor k.
This step is negligible if the MergeBox representation is used in combination with Stereo vision or LiDAR to obtain the absolute depth and hence the absolute scale k_{abs}.

3) **Orientation.** Depending on which projection of the vehicle's parts is larger compared to its 3D dimension we calculate the orientation θ of template t by forcing the projection of the rotated template to either fit x_{max} or x_{min}

$$x_{\text{max}} \overset{!}{=} \left(A \left[T_\delta \; R_\theta \; S^{(k)} \; p_2^{(t)} \right] \right)_x \tag{3}$$

$$x_{\text{min}} \overset{!}{=} \left(A \left[T_\delta \; R_\theta \; S^{(k)} \; p_4^{(t)} \right] \right)_x . \tag{4}$$

R_θ denotes the rotation matrix around the y-axis in the vehicle coordinate frame, $S^{(k)}$ is the isotropic homogeneous scaling matrix. θ therefore corresponds to the real yaw angle of a vehicle.

4) **Template Selection.** We select the template b that minimizes the difference between remaining edge at x_{min} or x_{max} and the point p_2 or p_4

$$b = \arg\min_t \left| x_{\text{min}} - \left(A \left[T_\delta \; R_\theta \; S^{(k)} \; p_4^{(t)} \right] \right)_x \right| \tag{5}$$

$$b = \arg\min_t \left| x_{\text{max}} - \left(A \left[T_\delta \; R_\theta \; S^{(k)} \; p_2^{(t)} \right] \right)_x \right| . \tag{6}$$

Finally, we end up with a best fitting template b as well as its orientation θ and depth-normalized scale k.

In contrast to all other current state-of-the-art method we don't need to learn the orientation or the dimension. Instead, we can calculate θ which reduces the number of parameters in our network.

Usually, orientation ground truth is generated by experts that use a 3D CAD model and manually rotate it until its projection fits the object in the image. As this task is very complex and requires a lot of expertise, the labeling costs for ground truth generation is expensive. By using our MergeBox representation, labeling cost can be significantly reduced. This is due to the fact that in addition to the 2D bounding box only one single parameter must be annotated to represent the 3D box in the image. This parameter can easily be identified, since it is the edge between the front/back and one side of a vehicle. KITTI [1] is the only publicly available dataset based on real world scenes that provide 3D bounding boxes as well as orientation data for different scenarios in autonomous driving. Our novel MergeBox representation enables the creation of large scale datasets for 3D bounding boxes.

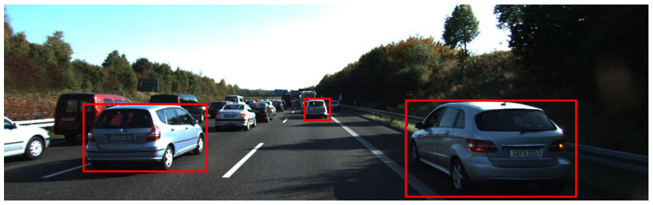
Fig. 4. Example of *orientation* θ vs. *viewpoint* α. All marked cars have the same 3D orientation θ compared to the ego-vehicle, but as their locations differ the vehicles appear under a different visual angle. Hence, their viewpoint α is different.

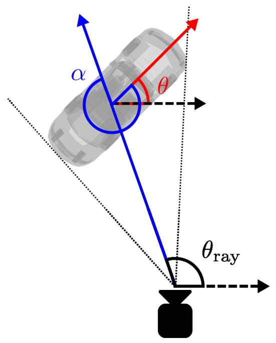

Fig. 5. Visual difference between orientation θ and viewpoint α. For self-driving cars the orientation of other vehicles is required to compute their trajectory and avoid collisions.

IV. EXPERIMENTS

In this section we first want to emphasize the difference between *orientation estimation* and *viewpoint estimation* and give details of the experiments we carried out for the KITTI Object Detection and Orientation Estimation challenge.

A. Orientation vs Viewpoint

While *orientation* denotes the real 3D orientation θ of an object relative to the observer, the angle under which an observer sees an object is defined as the *viewpoint* α. Fig. 4 illustrates the difference between these two terms. All boxed cars are moving in the same direction, their orientation compared to the ego-vehicle is $0°$. In contrast, the viewpoints of all cars differ as these cars are located on different lanes. While one can see the back and the right side of the car on the left lane, the left side and the back of the car right to the ego-vehicle is visible. For the car in the same lane, however, we only see the back. Thus, the viewpoint α for all cars is different.

To represent other vehicles on the road and avoid collisions, the real orientation θ in the coordinate frame of the ego-vehicle is required. In the KITTI Object Detection and

TABLE II

RESULTS OF THE ORIENTATION SCORE (OS) ON THE TEST SET OF THE KITTI OBJECT DETECTION AND ORIENTATION ESTIMATION [4] BENCHMARK. THE TOP 5 RANKED RESULTS AS WELL AS THE FASTEST 2 METHODS ARE COMPARED. WE ACHIEVE COMPETITIVE RESULTS WHILE OUTPERFORMING ALL OTHER METHODS W.R.T. TO SPEED BY AT LEAST A FACTOR OF 2 WHILE BEING THE ONLY APPROACH THAT IS INDEPENDENT OF ORIENTATION GROUND TRUTH.

Method	Task	Easy	Moderate	Hard	Speed	Independent of Orientation GT
Deep MANTA [17]	OS	**99.94 %**	**99.81 %**	**99.71 %**	700 ms	
Deep3DBox [5]	OS	99.91 %	99.66 %	99.45 %	1500 ms	
SubCNN [20]	OS	99.85 %	99.52 %	99.23 %	2000 ms	
DeepStereoOP [24]	OS	99.53 %	97.54 %	97.15 %	3400 ms	
Mono3D [19]	OS	98.59 %	97.69 %	97.32 %	4200 ms	
YOLOv2-3cls [23]	OS	40.46 %	39.81 %	40.74 %	50 ms	
FRCNN+Or [6]	OS	99.25 %	98.99 %	98.81 %	100 ms	
MB-Net (Ours)	OS	98.92 %	97.70 %	97.75 %	$\mathbf{22.52 \pm 1.14}$ **ms**	✓

Orientation Estimation benchmark, however, the viewpoint α is evaluated. Hence, many approaches try to directly learn and estimate α. Obtaining θ from α is done by $\theta = \alpha + \theta_{\text{ray}}$. θ_{ray} corresponds to the visual ray through the center of the object as shown in Fig. 5. Hence, the center of the object must be known – de facto knowledge on the 3D bounding box is required to correctly determine θ_{ray}. This introduces two sources of error Δi when deriving the required real orientation θ from α which is used for the benchmark evaluation

$$\Delta\theta = \Delta\alpha + \Delta\theta_{\text{ray}} \tag{7}$$

$$= \Delta\alpha + \Delta\mathbf{B}_{\text{3D}} \tag{8}$$

with $\mathbf{B}_{\text{3D}}$ being the 3D bounding box.

To reduce the error of the real orientation θ we therefore build a system that primarily focuses on the real orientation instead of the viewpoint.

B. Dataset

We evaluate our approach using the challenging KITTI Object Detection and Orientation Estimation benchmark [4] based on the KITTI dataset [1]. The dataset contains 7481 train and 7518 test images. We follow [14] to create a training and validation set with 3682 and 3799 images, respectively. This split ensures that all images of one video sequence are fully contained either in the training or the validation set. From the training set we extract all vehicles of type *car* and *van* with a difficulty of either *easy* or *moderate*. In addition we manually add some vehicles with an occlusion tag of *unknown* to ensure that they are not ignored during training. We extract all visible sides of the vehicles – i.e. fronts, backs and sides – as well as the complete 2D bounding box.

C. Metrics

We evaluate our approach using the metrics that are used in the KITTI Object Detection and Orientation Estimation benchmark. For object detection we use Average Precision (AP) and Average Orientation Similarity (AOS) for the viewpoint α as defined in [4]

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} \max_{\tilde{r}:\tilde{r} \geq r} s(\tilde{r}) \tag{9}$$

$$AOS = \frac{1}{|\mathcal{D}(r)|} \sum_{i \in \mathcal{D}(r)} \frac{1 + \cos \Delta_\theta^{(i)}}{2} \delta_i \tag{10}$$

with $r = \frac{TP}{TP+FN}$ and $\mathcal{D}$ being the set of all detections at recall level r. A bounding box detection will be classified as a TP if its intersection over union (IoU) with the GT box is ≥ 0.7. More details are given in [4]. As $AP \geq AOS$ we propose to also evaluate on the Orientation Score (OS) as introduced by [5] that is calculated by

$$OS = \frac{AP}{AOS}. \tag{11}$$

D. Training the MB-Net

As our approach is independent of the object detector as well as the base network we choose the SSD [2] architecture with Inception v1 [25] as the base network for MB-Net. We use cross entropy loss for classification and smooth L_1 loss [26] for localization. We train the network with Adam optimizer and an initial learning rate of 10^{-4}, $\gamma = 0.5$ and a stepsize of 500000. To evaluate on the *validation* set we train approx. 600000 iterations. For final evaluation on the *test* set we train on all 7481 available images of both training and validation set for approx. 1.2M iterations. Both training and inference are performed on a NVIDIA TITAN Xp.

E. Results

We select the 5 best performing methods as baselines regarding accuracy as well as the fastest two submissions to compare with our results as we aim to provide a system that is able to be run in real-time. The results of the evaluation on the KITTI *test* set is provided in Tab. II. Overall, we achieve an accuracy of 77.91 % for Detection (AP) and 76.12 % for Orientation (AOS) for the *moderate* difficulty in the KITTI Object Detection and Orientation Estimation benchmark. Simultaneously, we reach competitive performance for the Orientation Similarity (OS) while outperforming all other baselines in speed by at least a factor of 2.

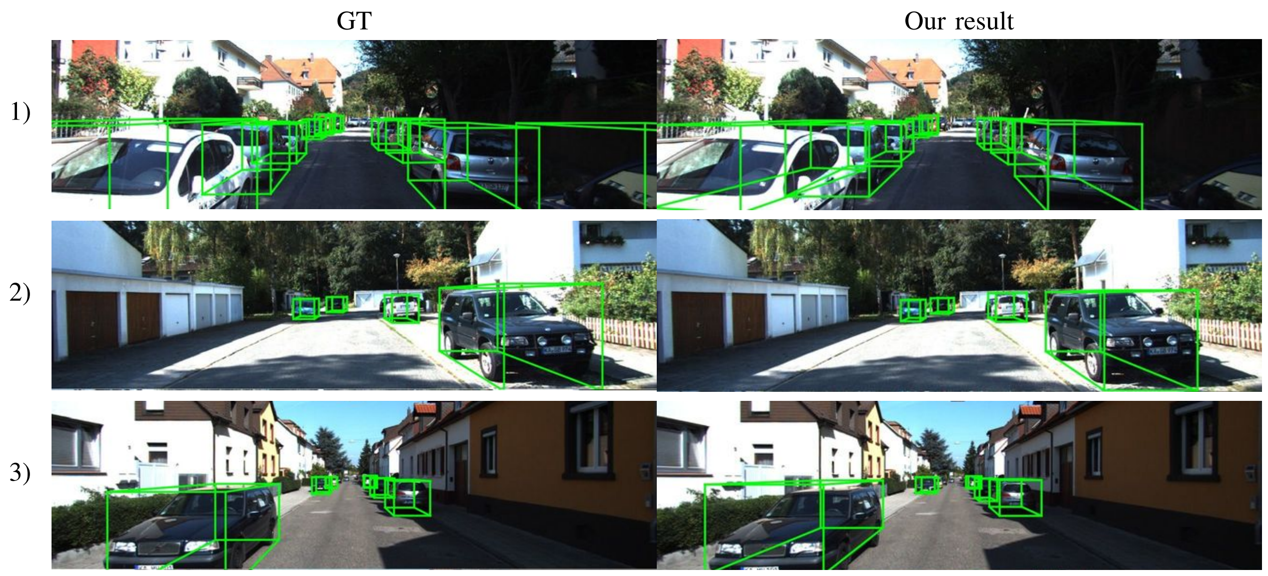

Fig. 6. Our results: 3D bounding box examples of the *validation* set. GT annotation is shown on the left, our results on the right.

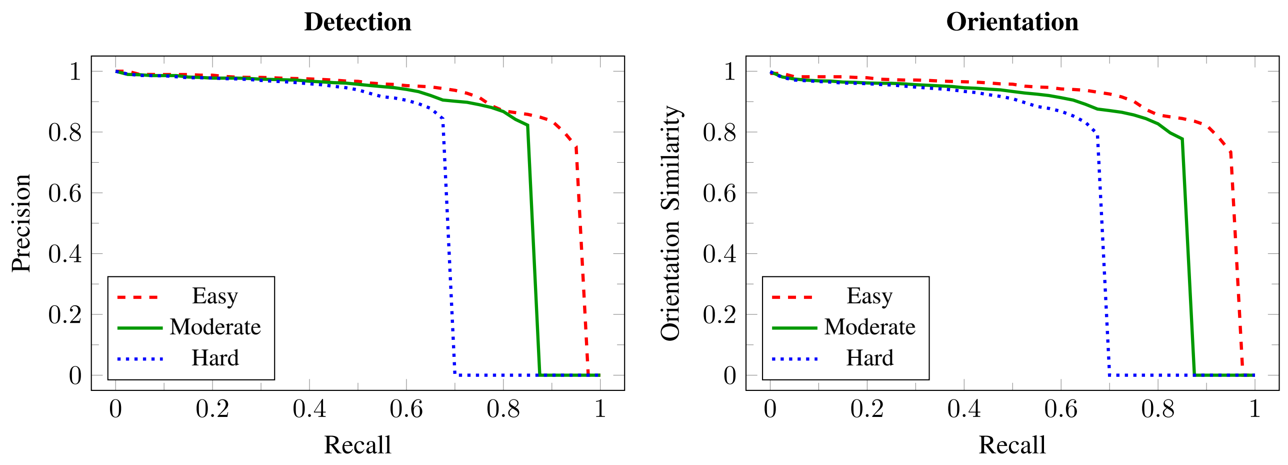

Fig. 7. Precision-recall curves for both object detection and orientation estimation results of the KITTI *test* set. For Detection (AP) we achieve 77.91 % and for Orientation (AOS) 76.12 % for the difficulty *moderate*.

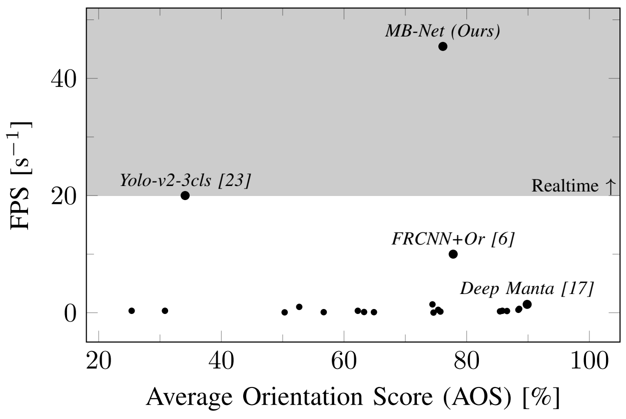

Fig. 8. Current submissions on the KITTI Object Detection and Orientation Estimation benchmark [4]. There are only two approaches that allow for realtime detection, i.e. with a speed of $\geq$ 20 FPS according to reported runtimes of the authors.

We show several visual examples from the *validation* set in Fig. 6, precision-recall curves for all difficulties (*easy*, *moderate*, *hard*) of the KITTI *test* set are given in Fig. 7.

Fig. 8 illustrates all submissions on the KITTI Object Detection and Orientation Estimation benchmark with their accuracy as well as their reported speed.

F. Ablation Studies

We conduct an ablation study to analyze the impact of different template boxes on the accuracy of MB-Net. To this end, we evaluate the *validation* set multiple times with a subset of the used templates provided in Tab. I. We show in Tab. III that the real orientation θ can be derived very accurately even if only some of the templates t_i are used. This result may be surprising; however, most of the vehicles available on the market have similar ratios of width, length and height. As these ratios are an important key during the calculation of the orientation, minor changes in these ratios

TABLE III

ABLATION STUDY USING THE KITTI VALIDATION SET. WE SHOW THAT THE AVERAGE ORIENTATION SCORE (AOS) FOR THE KITTI MODERATE DIFFICULTY REMAINS CONSTANT WHEN ADDING MORE TEMPLATES.

Compact	Sedan	Estate Car	SUV	Van	Large Van	OS	AP	AOS
✓						96.66 %	80.81 %	78.11 %
✓	✓					96.67 %	80.81 %	78.12 %
✓	✓	✓				96.68 %	80.81 %	78.13 %
✓	✓	✓	✓			96.68 %	80.81 %	78.13 %
✓	✓	✓	✓	✓		**96.71 %**	80.81 %	**78.15 %**
✓	✓	✓	✓	✓	✓	96.70 %	80.81 %	78.14 %

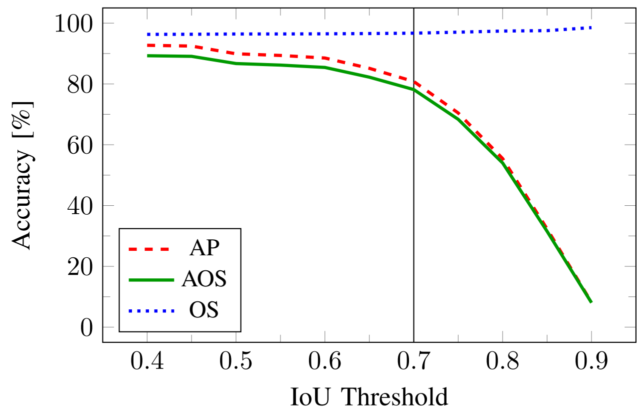

Fig. 9. Ablation study on the KITTI *validation* set to evaluate the localization accuracy of our detections for the *moderate* class. If the IoU overlap between a prediction and GT is relaxed to be a true positive prediction, both AOS and AP increase. In the KITTI Object Detection and Orientation Estimation Benchmark [4] an IoU overlap of 0.7 is required.

will not lead to larger errors of the final orientation. As a result, our model could be simplified with only one template. However, we decide to make use of all available templates to allow for a better description and classification of the appearing vehicles.

Additionally, we evaluate the localization accuracy of our projected boxes. We vary the IoU threshold needed to label a prediction as TP. As shown in Fig. 9 we achieve better results for smaller IoU thresholds on the KITTI *validation* set. At the same time the OS value increases with higher IoU thresholds, which means that the orientation estimation benefits from a perfectly localized detection. Since the overall detection rate increases for smaller IoU thresholds, we see that there is huge potential to push this approach even further as especially for small objects a localization offset of just a few pixels results in a huge difference in the IoU value. Hence, refining the localization for small objects could yield a large gain in accuracy for both detection and orientation.

G. Failure Cases

In the visual inspection of the results, we can identify different causes for drops in accuracy. The network outputs some false positive detections due to reflections as shown in Fig. 10. Training such cases as hard negative examples

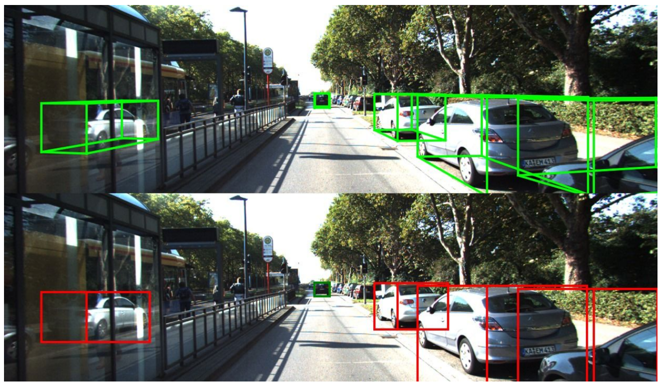

Fig. 10. Failure due to reflection that results in a false positive in the KITTI *test* set. The calculated 3D boxes are shown in green **(top)**, MergeBox representations in red **(bottom)**.

is not possible as there is no depiction or reflection tag in the training data. In addition, there is only a little amount of such reflections and depictions in the KITTI dataset.

In Fig. 11 we can identify a false negative as the network detects a *vehicle* object but no other detection that can be used to generate a MergeBox. By design, vehicles with only one part visible, i.e. vehicles that are orientated to the direction of $\pm\theta_{\text{ray}} \pm 90°$, are more susceptible to this kind of induced errors. In KITTI only $18.5\,\%$ of all annotated GT data correspond to vehicles with only one side visible. Hence, it is likely that the network is exhaustively trained on the existence of another visible part of the vehicle. This leads to a drop in the detection confidence of these kinds of vehicles. To overcome this problem, a balanced training of both vehicles with one visible side as well as ones with two sides visible is required. However, this cannot be accomplished with only using the KITTI dataset. Therefore, more balanced data is required additionally.

Finally, we find several false positives due to missing GT annotations in the *validation* set. An example is given in Fig. 12.

V. CONCLUSIONS

In this paper, we introduce the MergeBox as a novel representation for objects that combines the strengths of both the 2D object detection pipelines – especially their speed – and the representation of a vehicle as a 3D object. This 3D representation includes the vehicle's orientation that is important for trajectory planning of other vehicles and collision avoidance. In contrast to other approaches we thereby don't require GT data annotated with orientation and dimensions thus reducing the labeling costs for GT generation. Instead, the 3D orientation of a vehicle can be derived from the MergeBox representation in a closed form. As the proposed representation is network independent, it can be combined with any current state-of-the-art 2D bounding box detector. Hence, it can also be used with very fast single shot detectors such as SSD [2]. In the very challenging KITTI Object Detection and Orientation Estimation benchmark our MB-Net achieves competitive results and outperforms all other

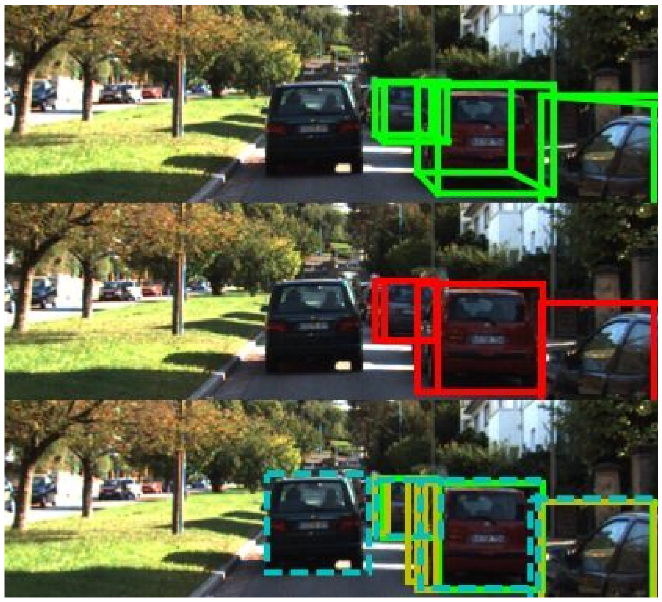

Fig. 11. False negative in the KITTI *test* set. The initial network predictions are shown, but the network only predicts a *vehicle* object **(bottom)**. To build a MergeBox we require at least 2 different detections per object – in this case a detection for *back* is missing. As no MergeBox can be generated **(center)** there is no 3D box **(top)**.

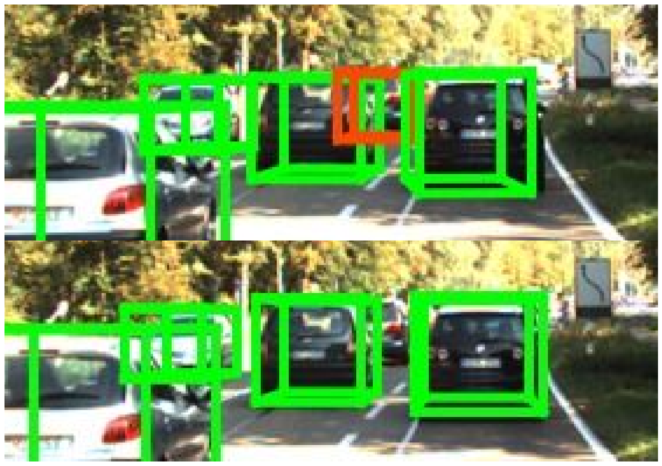

Fig. 12. Detection that is treated as a false positive in the KITTI *validation* set **(top)**. The vehicle marked in red is correctly detected but a GT annotation is missing **(bottom)**.

submissions in terms of speed being the only submission that is able to run in real time with ≥ 20 FPS.

References

[1] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[2] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

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

[4] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3354–3361.

[5] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3d bounding box estimation using deep learning and geometry," in *CVPR*, 2017.

[6] C. Guindel, D. Martin, and J. M. Armingol, "Joint Object Detection and Viewpoint Estimation using CNN features," in *IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, 2017.

[7] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1521–1529.

[8] P. Poirson, P. Ammirato, C.-Y. Fu, W. Liu, J. Kosecka, and A. C. Berg, "Fast single shot detection and pose estimation," in *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE, 2016, pp. 676–684.

[9] E. Ohn-Bar and M. M. Trivedi, "Learning to detect vehicles by clustering appearance patterns," *T-ITS*, 2015.

[10] M. Braun, Q. Rao, Y. Wang, and F. Flohr, "Pose-rcnn: Joint object detection and pose estimation using 3d object proposals," in *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*. IEEE, 2016, pp. 1546–1551.

[11] B. Li, "3d fully convolutional network for vehicle detection in point cloud," in *IROS*, 2017.

[12] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3d lidar using fully convolutional network," in *RSS 2016*.

[13] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *IEEE CVPR*, 2017.

[14] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Data-driven 3d voxel patterns for object category recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[15] X. Zhang, W. Xu, C. Dong, and J. M. Dolan, "Efficient l-shape fitting for vehicle detection using laser scanners," in *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE, 2017, pp. 54–59.

[16] X. Shen, S. Pendleton, and M. H. Ang, "Efficient l-shape fitting of laser scanner data for vehicle pose estimation," in *Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM), 2015 IEEE 7th International Conference on*. IEEE, 2015, pp. 173–178.

[17] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teulire, and T. Chateau, "Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image," in *CVPR*, 2017.

[18] B. Pepik, M. Stark, and B. Schiele, "Multi-view and 3d deformable part models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 11, pp. 2232–2245, 2015.

[19] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3d object detection for autonomous driving," in *CVPR*, 2016.

[20] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Subcategory-aware convolutional neural networks for object proposals and detection," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.

[21] X. Chen, K. Kundu, Y. Zhu, A. Berneshawi, H. Ma, S. Fidler, and R. Urtasun, "3d object proposals for accurate object class detection," in *NIPS*, 2015.

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[23] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[24] C. C. Pham and J. W. Jeon, "Robust object proposals re-ranking for object detection in autonomous driving using convolutional neural networks," *Signal Processing: Image Communiation*, 2017.

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Computer Vision and Pattern Recognition (CVPR)*, 2015. [Online]. Available: http://arxiv.org/abs/1409.4842

[26] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.