SMOKE: Single-Stage Monocular 3D Object Detection via Keypoint Estimation

Zechen Liu[1] Zizhang Wu[1] Roland Tóth[2]
[1]ZongMu Tech [2]TU/e
{zechen.liu, zizhang.wu}@zongmutech.com, R.Toth@tue.nl

Abstract

Estimating 3D orientation and translation of objects is essential for infrastructure-less autonomous navigation and driving. In case of monocular vision, successful methods have been mainly based on two ingredients: (i) a network generating 2D region proposals, (ii) a R-CNN structure predicting 3D object pose by utilizing the acquired regions of interest. We argue that the 2D detection network is redundant and introduces non-negligible noise for 3D detection. Hence, we propose a novel 3D object detection method, named SMOKE, in this paper that predicts a 3D bounding box for each detected object by combining a single keypoint estimate with regressed 3D variables. As a second contribution, we propose a multi-step disentangling approach for constructing the 3D bounding box, which significantly improves both training convergence and detection accuracy. In contrast to previous 3D detection techniques, our method does not require complicated pre/post-processing, extra data, and a refinement stage. Despite of its structural simplicity, our proposed SMOKE network outperforms all existing monocular 3D detection methods on the KITTI dataset, giving the best state-of-the-art result on both 3D object detection and Bird's eye view evaluation. The code is available at https://github.com/lzccccc/SMOKE.

1. Introduction

Vision-based object detection is an essential ingredient of autonomous vehicle perception and infrastructure-less robot navigation in general. This type of detection methods is used to perceive the surrounding environment by detecting and classifying object instances into categories and identifying their locations and orientations. Recent developments in 2D object detection [28, 20, 27, 18, 12, 42] have achieved promising performance on both detection accuracy and speed. In contrast, 3D object detection [3, 16, 43] has proven to be a more challenging task as it aims to estimate pose and location for each object simultaneously.

Currently, the most successful 3D object detection methods heavily depend on LiDAR point cloud [43, 30, 40] or LiDAR-Image fusion information [17, 33, 5] (features learned from the point cloud are key components of the detection network). However, LiDAR sensors are extremely expensive, have a short service life time and are too heavy for autonomous robots. Hence, LiDARs are currently not considered to be economical to support autonomous vehicle operations. Alternatively, cameras are cost-effective, easily mountable and light-weight solutions for 3D object detection with long expected service time. Unlike LiDAR sensors, a single camera in itself can not obtain sufficient spatial information for the whole environment as single RGB images can not supply object location information or dimensional contour in the real world. While binocular vision restores the missing spatial information, in many robotics applications, especially Unmanned Aerial Vehicles (UAVs), it is difficult to realize binocular vision. Hence, it is desirable to perform 3D detection on a monocular image even if it is a more difficult and challenging task.

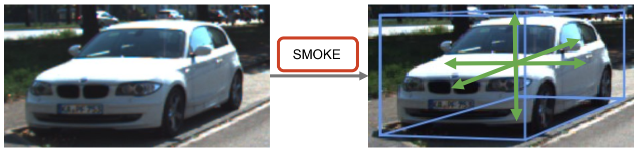

Figure 1. SMOKE directly predicts the 3D projected keypoint and 3D regression parameters on a single image. The whole network is trained end-to-end in a single stage.

Previous state-of-the-art monocular 3D object detection algorithms [25, 1, 21] heavily depend on region-based convolutional neural networks (R-CNN) or region proposal network (RPN) structures [28, 18, 7]. Based on the learned high number of 2D proposals, these approaches attach an additional network branch to either learn 3D information or to generate a pseudo point cloud and feed it into point-cloud-detection network. The resulting multi-stage complex process introduces persistent noise from 2D detection, which significantly increases the difficulty for the network to learn 3D geometry. To enhance performance, geometry reasoning [25], synthetic data [22] and post 3D-2D processing [1] have also been used to improve 3D object detection on single image. To the best knowledge of the authors, no reliable monocular 3D detection method has been intro-

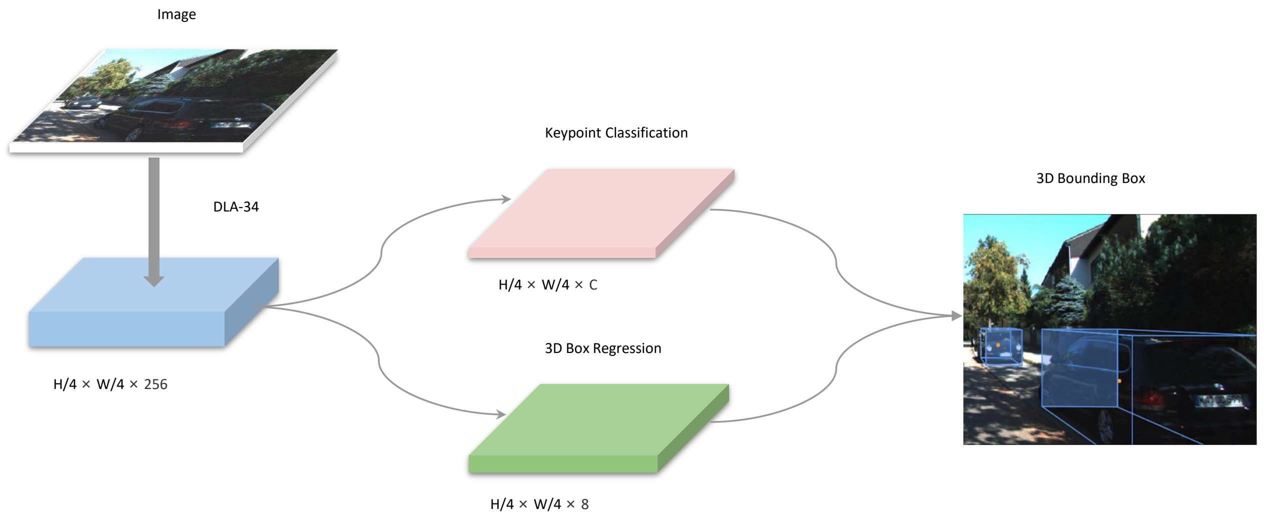

Figure 2. **Network Structure of SMOKE.** We leverage DLA-34 [41] to extract features from images. The size of the feature map is 1:4 due to downsampling by 4 of the original image. Two separate branches are attached to the feature map to perform keypoint classification (pink) and 3D box regression (green) jointly. The 3D bounding box is obtained by combining information from two branches.

duced so far to learn 3D information directly from the image plane avoiding the performance decrease that is inevitable with multi-stage methods.

In this paper, we propose an innovative single-stage 3D object detection method that pairs each object with a single keypoint. We argue and later show that a 2D detection, which introduces nonnegligible noise in 3D parameter estimation, is redundant to perform 3D object detection. Furthermore, 2D information can be naturally obtained if the 3D variables and camera intrinsic matrix are already known. Consequently, our designed network eliminates the 2D detection branch and estimates the projected 3D points on the image plane instead. A 3D parameter regression branch is added in parallel. This design results in a simple network structure with two estimation threads. Rather than regressing variables in a separate method by using multiple loss functions, we transform these variables together with projected keypoint to 8 corner representation of 3D boxes and regress them with a unified loss function. As in most single-stage 2D object detection algorithms, our 3D detection approach only contains one classification and regression branch. Benefiting from the simple structure, the network exhibits improved accuracy in learning 3D variables, and has better convergence and less overall computational needs.

The second contribution of our work is a multi-step disentanglement approach for 3D bounding box regression. Since all the geometry information is grouped into one parameter, it is difficult for the network to learn each variable accurately in a unified way. Our proposed method isolates the contribution of each parameter in both the 3D bounding box encoding phase and the regression loss function, which significantly helps to train the whole network effectively.

Our contribution is summarized as follows:

- We propose a one-stage monocular 3D object detection with a simple architecture that can precisely learn 3D geometry in an end-to-end fashion.
- We provide a multi-step disentanglement approach to improve the convergence of 3D parameters and detection accuracy.
- The resulting method outperforms all existing state-of-the-art monocular 3D object detection algorithms on the challenging KITTI dataset at the submission date November 12, 2019.

2. Related Work

In this section, we provide an in-depth overview of the state-of-the-art of 3D object detection based on the used sensor inputs. We first discuss LiDAR based and LiDAR-image fusion methods. After that, stereo image based methods are overviewed. Finally, we summarize approaches that only depend on single RGB images.

LiDAR/Fusion Based Methods: LiDAR-based 3D object detection methods achieve high detection precision by processing sparse point clouds into various representations. Some existing methods, e.g., [15, 39], project point clouds into a 2D Bird's eye view and equip standard 2D detection networks to perform object classification and 3D box regression. Other methods, like [43, 11, 13, 38], represent point clouds in voxel grid and then leverage 2D/3D CNNs to generate proposals. LiDAR-image fusion methods [17, 33, 5] learn relevant features from both the point clouds and the images together. These features are then combined and fed into a joint network trained for detection and classification.

Stereo Images Based Methods: The early work 3DOP [4] generates 3D proposals by exploring many handcrafted features such as stereo reconstruction, depth features, and object size priors. TLNet [26] introduces a triangulation based learning network to pair detected regions of interests between left and right images. Stereo R-CNN [16] creates 2D proposals simultaneously on stereo images. Then, the methods utilize keypoint prediction to generate a coarse 3D bounding box per region. A 3D box alignment w.r.t. stereo images is finally used on the object instance to improve the detection accuracy. Pseudo-LiDAR methods, e.g., [32], generate a "fake" point cloud and then feed these features into a point cloud based 3D detection network.

Monocular Image Based Methods: 3D object detection based on a single perspective image has been extensively studied and it is considered to be a challenging task. A common approach is to apply an additional 3D network branch to regress orientation and translation of object instances, see [3, 23, 37, 19, 14, 25, 22, 31]. Mono3D [3] generates 3D anchors by using a massive amount of features via semantic segmentation, object contour, and location priors. These features are then evaluated via an energy function to accommodate learning of relative information. Deep3DBox [23] introduces bin-based discretization for the estimation of local orientation for each object and 2D-3D bounding box constraint relationships to obtain the full 3D pose. MonoGRNet [25] subdivides the 3D object localization task into four tasks that estimate instance depth, 3D location of objects, and local corners respectively. These components are then stacked together to refine the 3D box in a global context. The network is trained in a stage-wise fashion and then trained end-to-end to obtain the final result. Some methods, like [36, 2, 10], rely on features detected in a 2D object box and leverage external data to pair information from 2D to 3D. DeepMANTA [2] proposes a coarse-to-fine process to generate accurate 2D object proposals, which are then used to match a 3D CAD model from an external annotated dataset. 3D-RCNN [10] also uses 3D models to pair the outputs from a 2D detection network. They then recover the 3D instance shape and pose by deploying a render-and-compare loss. Other approaches, like [21, 34, 9], generate hand-crafted features by transforming region of interest on images to other representations. AM3D transforms 2D imagery to a 3D point cloud plane by combining it with a depth map. A PointNet [24] is then used to estimate 3D dimensions, locations and orientations. The only one-stage method M3D-RPN [1] proposes a standalone network to generate 2D and 3D object proposals simultaneously. They further leverage a depth-aware network and post 3D-2D optimization technique to improve precision. OFTNet [29] maps the 2D feature map to bird's eye view by leveraging orthographic feature transform and regress each 3D variable independently. Consequently, none of the above methods can estimate 3D information accurately without generating 2D proposals.

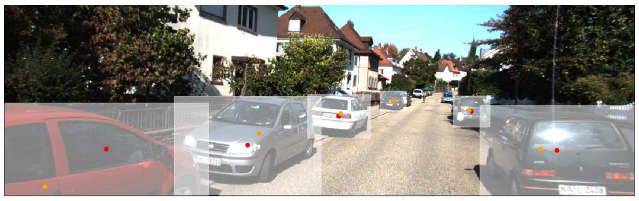
Figure 3. Visualization of difference between 2D center points (red) and 3D projected points (orange). Best viewed in color.

3. Detection Problem

We formulate the monocular 3D object detection problem as follows: given a single RGB image $I \in \mathbb{R}^{W \times H \times 3}$, with W being the width and H being the height of the image, find for each present object its category label C and its 3D bounding box B, where the latter is parameterized by 7 variables $(h, w, l, x, y, z, \theta)$. Here, (h, w, l) represent the height, weight, and length of each object in meters, and (x, y, z) is the coordinates (in meters) of the object center in the camera coordinate frame. Variable θ is the yaw orientation of the corresponding cubic box. The roll and pitch angles are set to zero by following the KITTI [6] annotation. Additionally, we make the mild assumption that the camera intrinsic matrix K is known for both training and inference.

4. SMOKE Approach

In this section, we describe the SMOKE network that directly estimates 3D bounding boxes for detected object instances from monocular imagery. In contrast to previous techniques that leverage 2D proposals to predict a 3D bounding box, our method can detect 3D information with a simple single stage. The proposed method can be divided into three parts: (i) backbone, (ii) 3D detection, (iii) loss function. First, we briefly discuss the backbone for feature extraction, followed by the introduction of the 3D detection network consisting of two separated branches. Finally, we discuss the loss function design and the multi-step disentanglement to compute the regression loss. The overview of the network structure is depicted in Fig. 2.

4.1. Backbone

We use a hierarchical layer fusion network DLA-34 [41] as the backbone to extract features since it can aggregate information across different layers. Following the same structure as in [42], all the hierarchical aggregation connections are replaced by a Deformable Convolution Network (DCN) [44]. The output feature map is downsampled 4 times with respect to the original image. Compared with the original implementation, we replace all BatchNorm (BN) [8] operation with GroupNorm (GN) [35] since it has been proven to be less sensitive to batch size and more robust to training noise. We also use this technique in the two prediction

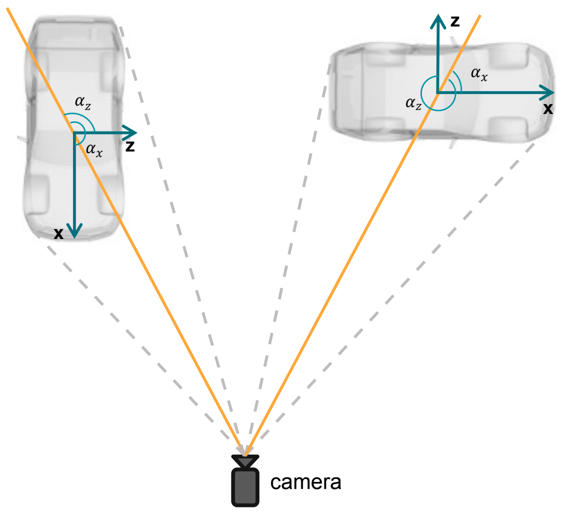

Figure 4. Relation of the observation angle α_x and α_z. α_x is provided in KITTI, while α_z is the value we choose to regress.

branches, which will be discussed in Sec. 4.2. This adjustment not only improves detection accuracy, but it also reduces considerably the training time. In Sec. 5.2, we provide performance comparison of BN and GN to demonstrate these properties.

4.2. 3D Detection Network

Keypoint Branch: We define the keypoint estimation network similar to [42] such that each object is represented by one specific keypoint. Instead of identifying the center of a 2D bounding box, the key point is defined as the projected 3D center of the object on the image plane. The comparison between 2D center points and 3D projected points is visualized in Fig. 3. The projected keypoints allow to fully recover 3D location for each object with camera parameters. Let $\begin{bmatrix} x & y & z \end{bmatrix}^\top$ represent the 3D center of each object in the camera frame. The projection of 3D points to points $\begin{bmatrix} x_\mathrm{c} & y_\mathrm{c} \end{bmatrix}^\top$ on the image plane can be obtained with the camera intrinsic matrix K in a homogeneous form:

$$\begin{bmatrix} z \cdot x_\mathrm{c} \\ z \cdot y_\mathrm{c} \\ z \end{bmatrix} = K_{3\times3} \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{1}$$

For each ground truth keypoint, its corresponding downsampled location on the feature map is computed and distributed using a Gaussian Kernel following [42]. The standard deviation is allocated based on the 3D bounding boxes of the ground truth projected to the image plane. Each 3D box on the image is represented by 8 2D points $\begin{bmatrix} x_{\mathrm{b},1\sim8} & y_{\mathrm{b},1\sim8} \end{bmatrix}^\top$ and the standard deviation is computed by the smallest 2D box with $\{x_\mathrm{b}^{\min}, y_\mathrm{b}^{\min}, x_\mathrm{b}^{\max}, y_\mathrm{b}^{\max}\}$ that encircles the 3D box.

Regression Branch: Our regression head predicts the essential variables to construct a 3D bounding box for each keypoint on the heatmap. Similar to other monocular 3D detection framework [22, 31], the 3D information is encoded as an 8-tuple $\tau = \begin{bmatrix} \delta_z & \delta_{x_\mathrm{c}} & \delta_{y_\mathrm{c}} & \delta_h & \delta_w & \delta_l & \sin\alpha & \cos\alpha \end{bmatrix}^\top$. Here δ_z denotes the depth offset, $\delta_{x_\mathrm{c}}, \delta_{y_\mathrm{c}}$ is the discretization offset due to downsampling, $\delta_h, \delta_w, \delta_l$ denotes the residual dimensions, $\sin(\alpha), \cos(\alpha)$ is the vectorial representation of the rotational angle α. We encode all variables to be learned in residual representation to reduce the learning interval and ease the training task. The size of feature map for regression results in $S_\mathrm{r} \in \mathbb{R}^{\frac{H}{R} \times \frac{W}{R} \times 8}$. Inspired by the lifting transformation described in [22], we introduce a similar operation $\mathcal{F}$ that converts projected 3D points to a 3D bounding box $B = \mathcal{F}(\tau) \in \mathbb{R}^{3\times8}$. For each object, its depth z can be recovered by pre-defined scale and shift parameters σ_z and μ_z as

$$z = \mu_z + \delta_z \sigma_z. \tag{2}$$

Given the object depth z, the location for each object in the camera frame can be recovered by using its discretized projected centroid $\begin{bmatrix} x_\mathrm{c} & y_\mathrm{c} \end{bmatrix}^\top$ on the image plane and the downsampling offset $\begin{bmatrix} \delta_{x_\mathrm{c}} & \delta_{y_\mathrm{c}} \end{bmatrix}^\top$:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = K_{3\times3}^{-1} \begin{bmatrix} z \cdot (x_\mathrm{c} + \delta_{x_\mathrm{c}}) \\ z \cdot (y_\mathrm{c} + \delta_{y_\mathrm{c}}) \\ z \end{bmatrix}. \tag{3}$$

This operation is the inverse of Eq. (1). In order to retrieve object dimensions $\begin{bmatrix} h & w & l \end{bmatrix}^\top$, we use a pre-calculated category-wise average dimension $\begin{bmatrix} \bar{h} & \bar{w} & \bar{l} \end{bmatrix}^\top$ computed over the whole dataset. Each object dimension can be recovered by using the residual dimension offset $\begin{bmatrix} \delta_h & \delta_w & \delta_l \end{bmatrix}^\top$:

$$\begin{bmatrix} h \\ w \\ l \end{bmatrix} = \begin{bmatrix} \bar{h} \cdot e^{\delta_h} \\ \bar{w} \cdot e^{\delta_w} \\ \bar{l} \cdot e^{\delta_l} \end{bmatrix}. \tag{4}$$

Inspired by [23], we choose to regress the observation angle α instead of the yaw rotation θ for each object. We further change the observation angle with respect to the object head α_x, instead of the commonly used observation angle value α_z, by simply adding $\frac{\pi}{2}$. The difference between these two angles is shown in Fig. 4. Moreover, each α is encoded as the vector $\begin{bmatrix} \sin(\alpha) & \cos(\alpha) \end{bmatrix}^\top$. The yaw angle θ can be obtained by utilizing α_z and the object location:

$$\theta = \alpha_z + \arctan\left(\frac{x}{z}\right). \tag{5}$$

Finally, we can construct the 8 corners of the 3D bounding box in the camera frame by using the yaw rotation matrix R_θ, object dimensions $\begin{bmatrix} h & w & l \end{bmatrix}^\top$ and location $\begin{bmatrix} x & y & z \end{bmatrix}^\top$:

$$B = R_\theta \begin{bmatrix} \pm h/2 \\ \pm w/2 \\ \pm l/2 \end{bmatrix} + \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{6}$$

4.3. Loss Function

Keypoint Classification Loss: We employ the penalty-reduced focal loss [12, 42] in a point-wise manner on the downsampled heatmap. Let $s_{i,j}$ be the predicted score at the heatmap location (i, j) and $y_{i,j}$ be the ground-truth value of each point assigned by Gaussian Kernel. Define $\breve{y}_{i,j}$ and $\breve{s}_{i,j}$ as:

$$\breve{y}_{i,j} = \begin{cases} 0 & \text{if } y_{i,j} = 1 \\ y_{i,j} & \text{otherwise} \end{cases}, \quad \breve{s}_{i,j} = \begin{cases} s_{i,j} & \text{if } y_{i,j} = 1 \\ 1 - s_{i,j} & \text{otherwise} \end{cases},$$

For simplicity, we only consider a single object class here. Then, the classification loss function is constructed as

$$L_{\text{cls}} = -\frac{1}{N} \sum_{i,j=1}^{h,w} (1 - \breve{y}_{i,j})^{\beta} (1 - \breve{s}_{i,j})^{\gamma} \log(\breve{s}_{i,j}), \tag{7}$$

where γ and β are tunable hyper-parameters and N is the number of keypoints per image. The term $(1 - y_{i,j})$ corresponds to penalty reduction for points around the groundtruth location.

Regression Loss: We regress the 8D tuple τ to construct the 3D bounding box for each object. We also add channel-wise activation to the regressed parameters of dimension and orientation at each feature map location to preserve consistency. The activation functions for the dimension and the orientation are chosen to be the sigmoid function σ and the ℓ_2 norm, respectively:

$$\begin{bmatrix} \delta_h \\ \delta_w \\ \delta_l \end{bmatrix} = \sigma\left(\begin{bmatrix} o_h \\ o_w \\ o_l \end{bmatrix}\right) - \frac{1}{2}, \quad \begin{bmatrix} \sin\alpha \\ \cos\alpha \end{bmatrix} = \begin{bmatrix} o_{\sin} / \sqrt{o_{\sin}^2 + o_{\cos}^2} \\ o_{\cos} / \sqrt{o_{\sin}^2 + o_{\cos}^2} \end{bmatrix},$$

Here o stands for the specific output of network. By adopting the keypoint lifting transformation introduced in Sec. 4.2, we define the 3D bounding box regression loss as the ℓ_1 distance between the predicted transform $\hat{B}$ and the groundtruth B:

$$L_{\text{reg}} = \frac{\lambda}{N} \|\hat{B} - B\|_1, \tag{8}$$

where λ is a scaling factor. This is used to ensure that neither the classification, nor the regression dominates the other. The disentangling transformation of loss has been proven to be an effective dynamic method to optimize 3D regression loss functions in [31]. Following this design, we extend the concept of loss disentanglement into a multi-step form. In Eq. (3), we use the projected 3D groundtruth points on the image plane $\begin{bmatrix} x_c & y_c \end{bmatrix}^\top$ with the network predicted discretization offset $\begin{bmatrix} \hat{\delta}_{x_c} & \hat{\delta}_{y_c} \end{bmatrix}^\top$ and depth $\hat{z}$ to retrieve the location $\begin{bmatrix} \hat{x} & \hat{y} & \hat{z} \end{bmatrix}^\top$ of each object. In Eq. (5), we use the groundtruth location $\begin{bmatrix} x & y & z \end{bmatrix}^\top$ and the predicted observation angle $\hat{\alpha}_z$ to construct the estimated yaw orientation $\hat{\theta}$. The 8 corners representation of the 3D bounding box is also isolated into three different groups following the concept of disentanglement, namely orientation, dimension and location. The final loss function can be represented by:

$$L = L_{\text{cls}} + \sum_{i=1}^{3} L_{\text{reg}}(\hat{B}_i), \tag{9}$$

where i represents the number of groups we define in the 3D regression branch. The multi-step disentangling transformation divides the contribution of each parameter group to the final loss. In Sec. 5.2, we show that this method significantly improves detection accuracy.

4.4. Implementation

In this section, we discuss the implementation of our proposed methodology in detail together with selection of the hyperparemeters.

Preprocessing: We avoid applying any complicated preprocessing method on the dataset. Instead, we only eliminate objects whose 3D projected center point on the image plane is out of the image range. Note that the total number of projected center points outside the image boundary for the *car* instance is 1582. This accounts for only the 5.5% of the entire set of 28742 labeled cars

Data Augmentation: Data augmentation techniques we used are random horizontal flip, random scale and shift. The scale ratio is set to 9 steps from 0.6 to 1.4, and the shift ratio is set to 5 steps from -0.2 to 0.2. Note that the scale and shift augmentation methods are only used for heatmap classification since the 3D information becomes inconsistent with data augmentation.

Hyperparameter Choice: In the backbone, the group number for GroupNorm is set to 32. For channels less than 32, it is set to be 16. For Eq. (7), we set $\gamma = 2$ and $\beta = 4$ in all experiments. Based on [31], the reference car size and depth statistics we use are $\begin{bmatrix} \bar{h} & \bar{w} & \bar{l} \end{bmatrix}^\top = [1.63 \;\; 1.53 \;\; 3.88]^\top$ and $\begin{bmatrix} \mu_z & \sigma_z \end{bmatrix}^\top = [28.01 \;\; 16.32]^\top$ (measured in meters).

Training: Our optimization schedule is easy and straightforward. We use the original image resolution and pad it to 1280×384. We train the network with a batch size of 32 on 4 Geforce TITAN X GPUs for 60 epochs. The learning rate is set at 2.5×10^{-4} and drops at 25 and 40 epochs by a factor of 10. During testing, we use the top 100 detected 3D projected points and filter it with a threshold of 0.25. No data augmentation method and NMS are used in the test procedure. Our implementation platform is Pytorch 1.1, CUDA 10.0, and CUDNN 7.5.

Method	Backbone	Runtime(s)	3D Object Detection			Birds' Eye View		
			Easy	Moderate	Hard	Easy	Moderate	Hard
OFTNet[29]	ResNet-18	0.50	1.32	1.61	1.00	7.16	5.69	4.61
GS3D[14]	VGG-16	2.00	4.47	2.90	2.47	8.47	6.08	4.94
MonoGR[25]	VGG-16	0.06	9.61	5.74	4.25	18.19	11.17	8.73
ROI-10D[22]	ResNet-34	0.20	4.32	2.02	1.46	9.78	4.91	3.74
MonoDIS[31]	ResNet-34	0.10	10.37	7.94	6.40	17.23	13.19	11.12
M3D-RPN[1]	DenseNet-121	0.16	**14.76**	9.71	7.42	**21.02**	13.67	10.23
Ours	DLA-34	**0.03**	14.03	**9.76**	**7.84**	20.83	**14.49**	**12.75**

Table 1. **Test set performance.** 3D object detection and Bird's eye view performance w.r.t. the car class on the official KITTI data set using the *test* split. Both metrics are evaluated by AP$|_{R_{40}}$ at 0.7 IoU threshold.

Method	3D Object Detection / Birds' Eye View		
	Easy	Moderate	Hard
CenterNet[42]	0.86 / 3.91	1.06 / 4.46	0.66 / 3.53
Mono3D[3]	2.53 / 5.22	2.31 / 5.19	2.31 / 4.13
OFTNet[29]	4.07 / 11.06	3.27 / 8.79	3.29 / 8.91
GS3D[14]	11.63 / -	10.51 / -	10.51 / -
MonoGR[25]	13.88 / -	10.19 / -	7.62 / -
ROI-10D[22]	9.61 / 14.50	6.63 / 9.91	6.29 / 8.73
MonoDIS[31]	18.05 / 24.26	14.98 / 18.43	13.42 / 16.95
M3D-RPN[1]	20.40 / 26.86	16.48 / 21.15	13.34 / 17.14
Ours	14.76 / 19.99	12.85 / 15.61	11.50 / 15.28

Table 2. **Validation set performance.** 3D object detection and Bird's eye view performance w.r.t. the car class on the official KITTI data set using the *val* split. Both metrics are evaluated by AP$|_{R_{11}}$ at 0.7 IoU threshold.

Method	2D Object Detection		
	Easy	Moderate	Hard
Mono3D[3]	94.52	89.37	79.15
OFTNet[29]	-	-	-
GS3D[14]	86.23	76.35	62.67
MonoGR[25]	88.65	77.94	63.31
ROI-10D[22]	76.56	70.16	61.15
MonoDIS[31]	94.61	89.15	78.37
M3D-RPN[1]	89.04	85.08	69.26
Ours	92.88	86.95	77.04

Table 3. **2D detection.** AP$|_{R_{40}}$ performance w.r.t. the car class on the official KITTI data set using the *test* split.

5. Performance Evaluation

We evaluate the performance of our proposed framework on the challenging KITTI dataset. The KITTI dataset is a broadly used publicly available dataset to evaluate visual algorithms on a driving scene considered representative for autonomous driving. It contains 7481 images for training and 7518 images for testing. The test metric is divided into *easy*, *moderate* and *hard* cases based on the height of the 2D bounding box of object instances, occlusion and truncation level. Frequently, the training set is split into 3712 training examples and 3769 validation examples as mentioned in [3]. For the 3D detection task of our proposed method, the 3D Object Detection and Bird's Eye View benchmarks are available for evaluation.

5.1. Detection on KITTI

3D Object Detection Performance: The 3D detection results of our proposed method on the split sets *test* and *val* are compared with the state-of-the-art single image-based methods in Tabs. 1 and 2. We principally focus on the *car* class since it has been at the focus of previous cooperative studies. For both tasks, the *average precision* (AP) with *Intersection over Union* (IoU) greater than 0.7 is used as the metric for evaluation. Note that as pointed out by [31], the official KITTI evaluation has been using 40 recall points instead of 11 recall points to measure the AP value since October 8, 2019. However, previous methods only report accuracy at 11 points on the *val* set. For fair comparison, we report the average precision on 40 points AP$|_{R_{40}}$ on the *test* set and AP$|_{R_{11}}$ on the *val* set.

Results on the *test* split, shown in Tab. 1, show that SMOKE outperforms all existing monocular methods on both 3D object detection and Bird's eye view evaluation metrics. We achieve improvement in the moderate and hard sets and comparable results on the easy set in the 3D object detection task. For Bird's eye view detection, we also achieve notable improvement on the moderate and hard sets. Compared with other methods that increase image size for better performance, our approach uses relatively low-resolution input and still achieves competitive results on the hard set in 3D detection. Next to these, SMOKE shows a significant improvement on detection speed. Without the time-consuming region proposal process and by the benefits of single-stage structure, our proposed method only needs 30ms to run on a TITAN XP. Note that we only compare our method with methods that directly learn features from images. Approaches based on hand-crafted features [34, 21] are not listed in the table. However, with respect to the *val* set of KITTI, the performance degrades as reported in Tab. 2. We argue that this is due to a lack of training objects. A similar problem has been reported in [42].

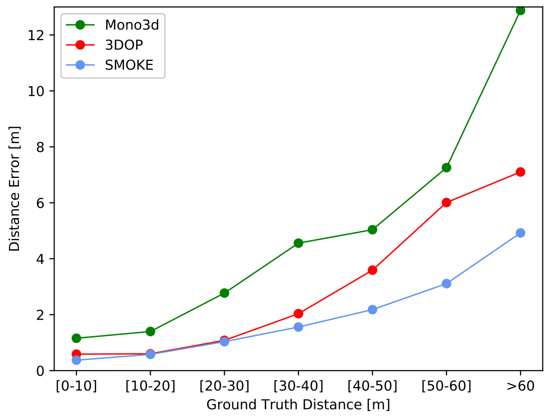

Figure 5. Average depth estimation error visualized in intervals of 10 meters. Best viewed in color.

Estimation of object location in a monocular image is difficult since the incompleteness of spatial information. We evaluate the depth estimation of SMOKE using two different distance measures. In Fig. 5, the achieved depth error is displayed in intervals of 10 meters. The error is computed if the 2D bounding box of a detection with any of the ground truth objects has an IoU larger than 0.7. As shown in the figure, the depth estimation error increases as the distance grows. This phenomenon has been observed in many monocular image-based detection algorithms since small objects have large distance distribution. We compare our method with two other methods Mono3D [3] and 3DOP [4] on the same *val* set. The curve indicates that our proposed SMOKE method outperforms both methods largely on depth error. Especially at distances larger than 40m, our method achieves more robust and accurate depth estimation.

2D Object Detection: The 2D detection performance on the official KITTI *test* set is depicted in Tab. 3. Although the 2D bounding box is not directly regressed in the SMOKE network, we observe that our method achieves comparable results on the 2D object detection task. The 2D detection box is obtained as the smallest rectangle that contains the projected 3D bounding box on the image plane. Unlike other approaches following a 2D→3D structure, our proposed method reverse this process in a 3D→2D fashion and outperforms many of the existing methods. This clearly shows that 3D object detection provides more abundant information than 2D detection, hence 2D proposals are redundant and not needed for 3D detection. Furthermore, our proposed method does not use extra data, complicated networks and high-resolution input compared to other methods.

5.2. Ablation Study

In this section, we show the results of experiments we conducted to compare different normalization choices, loss function, and rotation angle parameterizations. All experiments are performed on the *train/val* split on the KITTI dataset. Moreover, we use *car* class to evaluate our model.

Option	3D Object Detection / Bird's Eye View		
	Easy	Moderate	Hard
BN	8.20 / 17.85	8.27 / 15.46	6.50 / 15.21
GN	**10.60 / 18.06**	**8.33 / 16.07**	**6.98 / 15.39**

Table 4. **Normalization Strategy.** GN perfoms better than BN on all difficulty sets and in both evaluation metrics.

Option	3D Object Detection / Bird's Eye View		
	Easy	Moderate	Hard
Smooth ℓ_1	10.60 / 18.06	8.33 / 16.07	6.98 / 15.39
ℓ_1	11.03 / **20.90**	10.53 / 15.95	9.14 / **15.57**
Dis. ℓ_1	**14.76** / 19.99	**12.85 / 16.07**	**11.50** / 15.39

Table 5. **Regression Loss.** ℓ_1 loss gains better performance than Smooth ℓ_1 loss. The disentanglement form further improves detection result.

Option	3D Object Detection / Bird's Eye View		
	Easy	Moderate	Hard
Quaternion	13.36 / 17.81	12.52 / 15.16	11.31 / 15.00
Vectorial	**14.76 / 19.99**	**12.85 / 16.07**	**11.50 / 15.39**

Table 6. **Rotation Parametrization.** Vectorial representation of angles yileds better result than the quaternion representation.

Normalization Strategy: We chose GN as the normalization strategy since it is less sensitive to batch size and cross-GPU training issues. We compare the performance difference in the 3D detection task of BN and GN used in the backbone network. As illustrated in Tab. 4, GN achieves noticeable improvement over BN on the *val* set. In addition, we notice that GN can save considerable time in training. For each epoch, GN consumes around 5 minutes while BN needs 8 minutes which takes 60% more time compared to GN.

Regression Loss: As shown in Tab. 5, we compare different regression loss functions for 3D bounding box estimation performance. We observe that ℓ_1 loss performs better than Smooth ℓ_1 loss. Same phenomenon is also found in the keypoint estimation problem [42] where ℓ_1 loss yields better performance than ℓ_2 loss. Moreover, applying disentanglement to 3D bounding box regression achieves significantly better performance on both 3D object detection and Birds' eye view evaluation.

Rotation Parametrization: We compare the performance of SMOKE with respect to different representations of rotation. Following prior work [22, 31], the orientation can be encoded as a 4D quaternion to formulate 3D bounding box. The result with this representation is illustrated in Tab. 6. We observe that our simple vectorial representation yields slightly better result than the quaternion representation on both 3D detection and Bird's eye view evaluation.

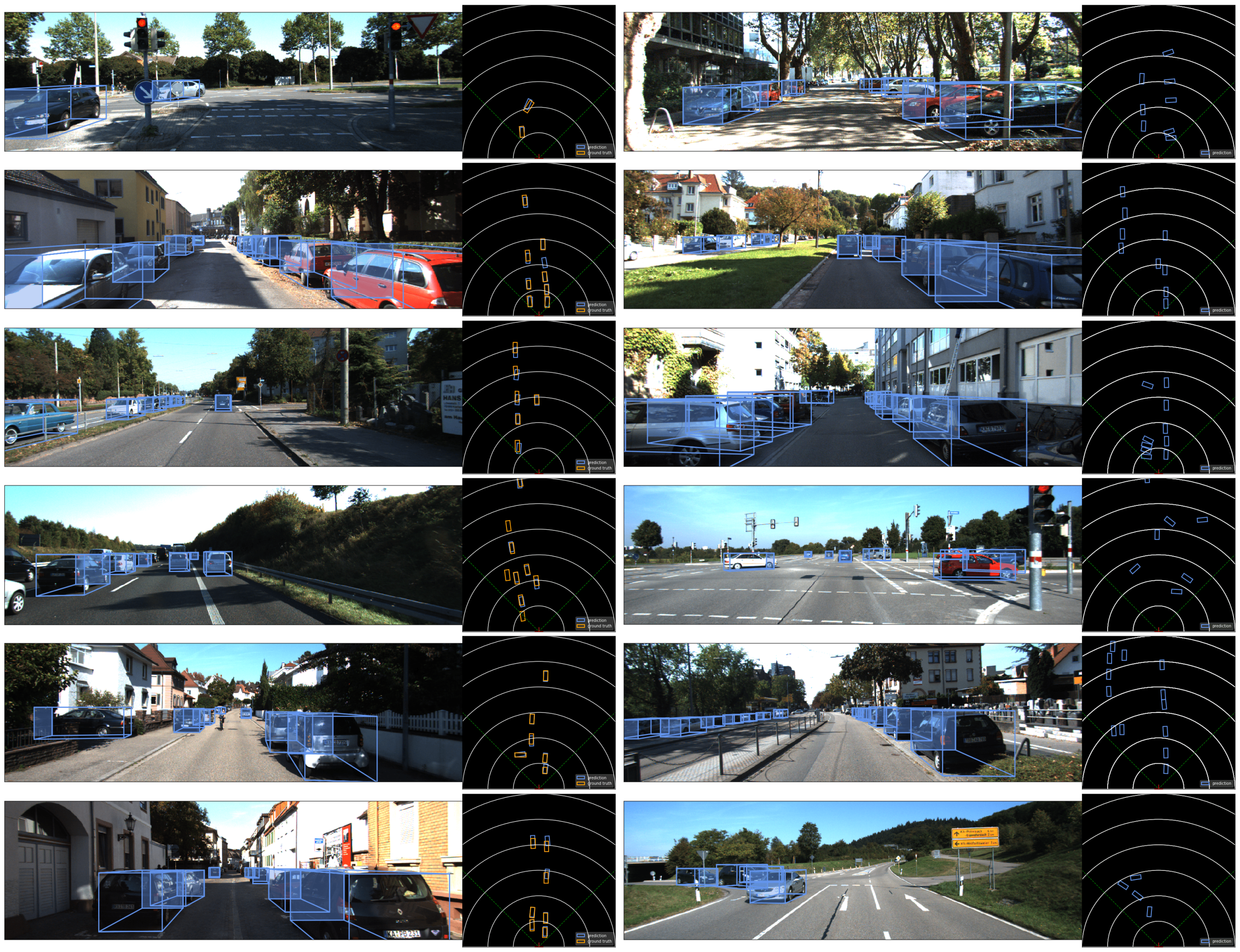

Figure 6. Qualitative examples from the validation (left) and test (right) sets in KITTI. The non-transparent side of the bounding box represents the front part of each car. Bird's eye view is also provided to show that SMOKE can recover object distances accurately. Note that all these images are not included in the training phase.

5.3. Qualitative Results

Qualitative results on both the *test* and *val* sets are displayed in Fig. 6. For better visualization and comparison, we also plot the object localization in Bird's eye view. The results clearly demonstrate that SMOKE can recover object distances accurately

6. Conclusion and Future Work

In this paper, we presented a novel single-stage monocular 3D object detection method based on projected 3D points on the image plane. Unlike previous methods, which depend on 2D proposals to estimate 3D information, our approach regresses 3D bounding boxes directly. This leads to a simple and efficient architecture. To further improve the convergence of regression loss, we proposed a multi-step disentanglement method to isolate the contribution of various parameter groups. In addition, our model does not need synthetic data, complicated pre/post-processing, and multi-stage training. Overall, we largely improve both the detection accuracy and speed on KITTI 3D object detection and Bird's eye view tasks.

Our proposed SMOKE 3D detection framework achieves promising accuracy and efficiency, which can be further extended and used on autonomous vehicles and in robotic navigation. In the future, we aim at extending our method to stereo images and further improving the estimation of projected 3D keypoints and their depth.

References

[1] Garrick Brazil and Xiaoming Liu. M3D-RPN: Monocular 3d region proposal network for object detection. In *ICCV*, 2019.

[2] Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Ce-

line Teuliere, and Thierry Chateau. Deep MANTA: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *CVPR*, 2017.

[3] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *CVPR*, 2016.

[4] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3D object proposals for accurate object class detection. In *NIPS*, 2015.

[5] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, 2017.

[6] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[7] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[8] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[9] Jason Ku, Alex D. Pon, and Steven L. Waslander. Monocular 3d object detection leveraging accurate proposals and shape reconstruction. In *CVPR*, 2019.

[10] Abhijit Kundu, Yin Li, and James M. Rehg. 3D-RCNN: Instance-level 3d object reconstruction via render-and-compare. In *CVPR*, 2018.

[11] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019.

[12] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *ECCV*, 2018.

[13] Bo Li. 3d fully convolutional network for vehicle detection in point cloud. In *IROS*, 2017.

[14] Buyu Li, Wanli Ouyang, Lu Sheng, Xingyu Zeng, and Xiaogang Wang. GS3D: An efficient 3d object detection framework for autonomous driving. In *CVPR*, 2019.

[15] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3d lidar using fully convolutional network. In *Robotics: Science and Systems*, 2016.

[16] Peiliang Li, Xiaozhi Chen, and Shaojie Shen. Stereo R-CNN based 3d object detection for autonomous driving. In *CVPR*, 2019.

[17] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *CVPR*, 2019.

[18] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *ICCV*, 2017.

[19] Lijie Liu, Jiwen Lu, Chunjing Xu, Qi Tian, and Jie Zhou. Deep fitting degree scoring network for monocular 3d object detection. In *CVPR*, 2019.

[20] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. In *ECCV*, 2016.

[21] Xinzhu Ma, Zhihui Wang, Haojie Li, Wanli Ouyang, and Zhang Pengbo. Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving. In *ICCV*, 2019.

[22] Fabian Manhardt, Wadim Kehl, and Adrien Gaidon. ROI-10D: Monocular lifting of 2d detection to 6d pose and metric shape. In *CVPR*, 2019.

[23] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3D bounding box estimation using deep learning and geometry. In *CVPR*, 2017.

[24] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

[25] Zengyi Qin, Jinglu Wang, and Yan Lu. MonoGRNet: A geometric reasoning network for monocular 3d object localization. In *AAAI*, 2019.

[26] Zengyi Qin, Jinglu Wang, and Yan Lu. Triangulation learning network: from monocular to stereo 3d object detection. *CVPR*, 2019.

[27] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[29] Thomas Roddick, Alex Kendall, and Roberto Cipolla. Orthographic feature transform for monocular 3d object detection. *BMVC*, 2019.

[30] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointR-CNN: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019.

[31] Andrea Simonelli, Samuel Rota Rota Bulò, Lorenzo Porzi, Manuel López-Antequera, and Peter Kontschieder. Disentangling monocular 3d object detection. In *ICCV*, 2019.

[32] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *CVPR*, 2019.

[33] Zhixin Wang and Kui Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. In *IROS*, 2019.

[34] Xinshuo Weng and Kris Kitani. Monocular 3D Object Detection with Pseudo-LiDAR Point Cloud. In *arXiv preprint arXiv:1903.09847*, 2019.

[35] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018.

[36] Yu Xiang, Wongun Choi, Yuanqing Lin, and Silvio Savarese. Subcategory-aware convolutional neural networks for object proposals and detection. In *WACV*, 2017.

[37] Bin Xu and Zhenzhong Chen. Multi-level fusion based 3D object detection from monocular images. In *CVPR*, 2018.

[38] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. In *Sensors*, 2018.

[39] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *CVPR*, 2018.

[40] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. STD: Sparse-to-dense 3d object detector for point cloud. In *ICCV*, 2019.

[41] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *CVPR*, 2018.

[42] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.

[43] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018.

[44] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *CVPR*, 2019.