Disentangling Monocular 3D Object Detection

Andrea Simonelli[△,⋆], Samuel Rota Bulò[△], Lorenzo Porzi[△], Manuel López-Antequera[△], Peter Kontschieder[△]
[△]Mapillary Research, [⋆]University of Trento
research@mapillary.com

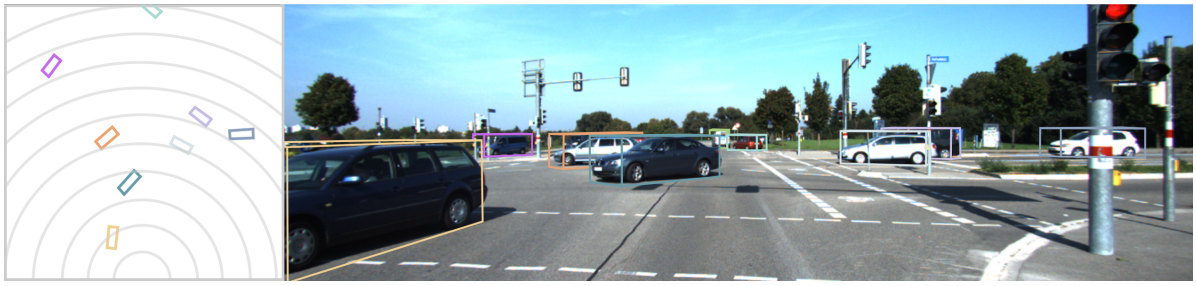

Figure 1: Results obtained from our single image, monocular 3D object detection network *MonoDIS* on a KITTI3D test image with corresponding birds-eye view, showing its ability to estimate size and orientation of objects at different scales.

Abstract

In this paper we propose an approach for monocular 3D object detection from a single RGB image, which leverages a novel disentangling transformation for 2D and 3D detection losses and a novel, self-supervised confidence score for 3D bounding boxes. Our proposed loss disentanglement has the twofold advantage of simplifying the training dynamics in the presence of losses with complex interactions of parameters, and sidestepping the issue of balancing independent regression terms. Our solution overcomes these issues by isolating the contribution made by groups of parameters to a given loss, without changing its nature. We further apply loss disentanglement to another novel, signed Intersection-over-Union criterion-driven loss for improving 2D detection results. Besides our methodological innovations, we critically review the AP metric used in KITTI3D, which emerged as the most important dataset for comparing 3D detection results. We identify and resolve a flaw in the 11-point interpolated AP metric, affecting all previously published detection results and particularly biases the results of monocular 3D detection. We provide extensive experimental evaluations and ablation studies on the KITTI3D and nuScenes datasets, setting new state-of-the-art results on object category car by large margins.

1. Introduction

Recent developments in object recognition [21] have led to near-human performance on monocular 2D detection tasks. For applications with given, realistic accuracy requirements or constraints on computational budget, it is possible to choose general-purpose 2D object detectors from a large pool [32, 22, 30, 39, 31, 19, 15].

The performance situation considerably changes in the 3D object detection case. Even though there are promising methods based on multi-sensor fusion (usually exploiting LIDAR information [17, 38, 37, 36] next to RGB images), 3D detection results produced from a single, monocular RGB input image lag considerably behind. This can be attributed to the ill-posed nature of the problem, where a lack of explicit knowledge about the unobserved depth dimension introduces ambiguities in 3D-to-2D mappings and hence significantly increases the task complexity.

To still enable 3D object detection from monocular images, current works usually make assumptions about the scene geometry, camera setup or the application (*e.g.* that cars cannot fly [29]). The implementation of such priors determines the encoding of extent and location/rotation of the 3D boxes, the corresponding 2D projections or their 3D box center depths. The magnitudes of these parameters have different units and therefore non-comparable meanings, which can negatively affect the optimization dynamics when error terms based on them are directly combined in a loss function. As a consequence, state-of-the-art, CNN-based monocular 3D detection methods [23, 29] report to train their networks in a stage-wise way. First the 2D detectors are trained until their performance stabilizes, before 3D reasoning modules can be integrated. While stage-wise training *per se* is not unusual in the context of deep learning, it could be an indication that currently used loss functions are yet sub-optimal.

A significant amount of recent works are focusing their experimental analyses on the KITTI3D dataset [8], and in

particular its *Car* category [23, 29, 33, 42]. The availability of suitable benchmark datasets confines the scope of experimental analyses and when only few datasets are available, progress in the research field is strongly tied to the expressiveness of used evaluation metrics. KITTI3D adopted the *11-point Interpolated Average Precision* metric [35] used in the PASCAL VOC2007 [7] challenge. We found a major flaw in the metric where using a single, confident detection result per difficulty category (KITTI3D distinguishes between *easy*, *moderate* and *hard* samples) suffices to obtain AP scores of $\approx 9\%$ on a dataset level, which is up to $3\times$ higher than the performance reported by recent works [5, 4, 11, 42].

The contributions of our paper disentangle the task of monocular 3D object detection at several levels. Our major technical contribution *disentangles* dependencies of different parameters by isolating and handling parameter groups individually at a loss-level. This overcomes the issue of non-comparability for parameter magnitudes, while preserving the nature of the final loss. Our loss disentanglement significantly improves losses on both, 2D and 3D tasks. It also enables us to effectively train the entire CNN architecture (2D+3D) together and end-to-end, without the need of hyperparameter-sensitive, stage-wise training or warm-up phases. As additional contributions we i) leverage 2D detection performance through a novel loss based on a *signed Intersection-over-Union* criterion and ii) introduce a loss term for predicting detection confidence scores of 3D boxes, learned in a self-supervised way.

Another major contribution is a critical review of the 3D metrics used to judge progress in monocular 3D object detection, with particular focus on the predominantly used KITTI3D dataset. We observe that a flaw in the definition of the 11-point, interpolated AP metric significantly biases 3D detection results at the performance level of current state-of-the-art methods. Our applied correction, despite bringing *all works evaluating on KITTI3D* back down to earth, more adequately describes their true performance.

For all our contributions, we provide ablation studies on the KITTI3D and the novel nuScenes [2] driving datasets. Fair comparisons indicate that our work considerably improves over current monocular 3D detection methods.

2. Related Work

We review the most recent, related works from 3D object detection and group them according to the data modalities used therein. After discussing RGB-only works just like ours, we list works exploiting also depth and/or synthetic data augmentation or 3D shape information, before finalizing with a high-level summary about LIDAR and/or stereo-based approaches.

RGB images only. Deep3DBox [24] proposed to estimate full 3D pose and object dimensions from a 2D box by exploiting constraints from projective geometry. The core idea is that the perspective projection of a 3D bounding box should fit tightly to at least one side of its corresponding 2D box detection. In SSD-6D [12] an initial 2D detection hypothesis is lifted to provide 6D pose of 3D objects by using structured discretizations of the full rotational space. 3D model information is learned by only training from synthetically augmented datasets. OFTNet [33] introduces an orthographic feature transform, mapping features extracted from 2D to a 3D voxel map. The voxel map's features are eventually reduced to 2D (birds-eye view) by integration along the vertical dimension, and detection hypotheses are efficiently processed by exploiting integral-image representations. Mono3D [4] emphasized on generation of 3D candidate boxes, scored by different features like class semantics, contour, shape and location priors. Even though at test time the results are produced based on single RGB images only, their method also requires semantic and instance segmentation results as input. The basic variant (w/o using depth) of ROI-10D [23] proposes a novel loss to lift 2D detection, orientation and scale into 3D space that can be trained in an end-to-end fashion. FQNet [20] infers a fitting quality criterion in terms of 3D IoU scores, allowing them to filter estimated 3D box proposals based on using only 2D object cues. MonoGRNet [29] is the current state-of-the-art for RGB-only input, using a CNN comprised of four sub-networks for 2D detection, instance depth estimation, 3D location estimation and local corner regression, respectively. The three latter sub-networks emphasize on geometric reasoning, *i.e*. instance depth estimation predicts the central 3D depth of the nearest object instance, 3D location estimation seeks for the 3D bounding box center by exploiting 3D to 2D projections at given instance depth estimations, and local corner regression directly predicts the eight 3D bounding box corners in a local (or allocentric [13, 23] way). It is relevant to mention that [29] reports that training was conducted stage-wise: First, the backbone is trained together with the 2D detector using Adam. Next, the geometric reasoning modules are trained (also with Adam). Finally, the whole network is trained end-to-end using stochastic gradient descent. The work in [1] learns to estimate correspondences between detected 2D keypoints and 3D counterparts. However, this requires manual annotations on the surface of 3D CAD models and is limited in dealing with occluded objects.

Including depth. An expansion stage of ROI-10D [23] takes advantage of depth information provided by SuperDepth [27], which itself is learned in a self-supervised manner. In [42], a multi-level fusion approach is proposed, exploiting disparity estimation results from a pre-trained module during both, the 2D box proposal generation stage as well as the 3D prediction part of their network.

Including 3D shape information. 3D-RCNN [13] exploits the idea of using inverse graphics for instance-level, amodal 3D shape and pose estimation of all object instances per image. They propose a differentiable Render-and-Compare loss, exploiting available 2D annotations in existing datasets for guiding optimization of 3D object shape and pose. In [25], the recognition task is tackled by jointly reasoning about the 3D shape of multiple objects. Deep-MANTA [3] uses 3D CAD models and annotated 3D parts in a coarse-to-fine localization process. The work in [26] encodes shape priors using keypoints for recovering the 3D pose and shape of a query object. In Mono3D++ [11], the 3D shape and pose for cars is provided by using a morphable wireframe, and it optimizes projection consistency between generated 3D hypotheses and corresponding, 2D pseudo-measurements.

LIDAR and/or stereo-based. 3DOP [5] exploits stereo images and prior knowledge about the scene to directly reason in 3D. Stereo R-CNN [16] tackles 3D object detection by exploiting stereo imagery and produces stereo boxes, keypoints, dimensions and viewpoint angles, summarized in a learned 3D box estimation module. In MV3D [6], a sensor-fusion approach for LIDAR and RGB images is presented, approaching 3D object proposal generation and multi-view feature fusion via individual sub-networks. Conversely, Frustrum-PointNet [28] directly operates on LIDAR point clouds and aligns candidate points provided from corresponding 2D detections for estimating the final, amodal 3D bounding boxes. PointRCNN [36] describes a 2-stage framework where the first stage provides bottom-up 3D proposals and the second stage refines them in canonical coordinates. RoarNet [37] applies a 2D detector to first estimate 3D poses of objects from a monocular image before processing corresponding 3D point clouds to obtain the final 3D bounding boxes.

3. Task Description

We address the problem of monocular 3D object detection, where the input is a single RGB image and the output consists in a 3D bounding box, expressed in camera coordinates, for each object that is present in the image (see, Fig. 1). As opposed to other methods in the literature, we do *not* take additional information as input like depth obtained from LIDAR or other supervised or self-supervised monocular depth estimators. Also the training data consists solely of RGB images with corresponding annotated 3D bounding boxes. Nonetheless, we require a calibrated setting so we assume that per-image calibration parameters are available both at training and test time.

4. Proposed Architecture

We adopt a two-stage architecture that shares a similar structure with the state-of-the-art [23]. It consists of a single-stage 2D detector (*first stage*) with an additional 3D detection head (*second stage*) constructed on top of features pooled from the detected 2D bounding boxes. Details of the architecture are given below.

4.1. Backbone

The backbone we use is a ResNet34 [10] with a Feature Pyramid Network (FPN) [18] built on top of it. The FPN network has the same structure as in [19] with 3+2 scales, connected to the output of modules conv3, conv4 and conv5 of ResNet34, corresponding to downsampling factors of $\times 8$, $\times 16$ and $\times 32$, respectively. Our ResNet34 differs from the standard one by replacing BatchNorm+ReLU layers with the synchronized version of InPlaceABN (iABN$^{\text{sync}}$) activated with LeakyReLU with negative slope 0.01 as proposed in [34]. This modification does not affect the performance of the network, but allows to free up a significant amount of GPU memory, which can be exploited to scale up the batch size or input resolution. All FPN blocks depicted in Fig. 2 correspond to 3×3 convolutions with 256 channels, followed by iABN$^{\text{sync}}$.

Inputs. The input x to the backbone is a single RGB image.

Outputs. The backbone provides 5 output tensors $\{f_1, \ldots, f_5\}$ corresponding to the 5 different scales of the FPN network, covering downsampling factors of $\times 8$, $\times 16$, $\times 32$, $\times 64$, and $\times 128$, each with 256 feature channels (see, Fig. 2).

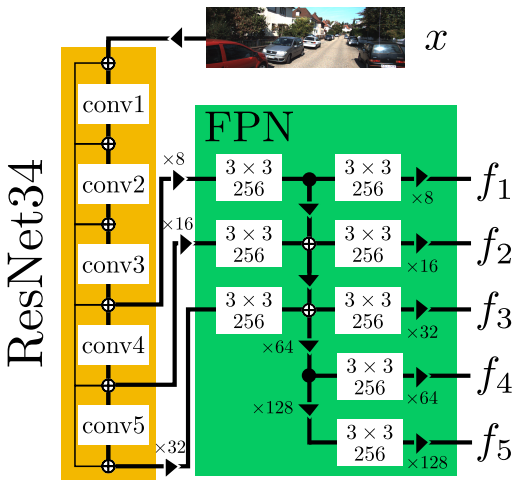

Figure 2: Backbone architecture. Rectangles in the "FPN" block represent convolutions followed by iABN$^{\text{sync}}$.

4.2. 2D Detection Head

We consider the head of the single-stage 2D detector implemented in RetinaNet [19], which applies a detection

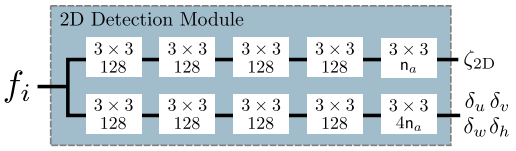

Figure 3: 2D detection module. Rectangles represent convolutions. All convolutions but the last per row are followed by iABN$^{\text{sync}}$.

module independently to each output f_i of the backbone described above. The detection modules share the same parameters but work inherently at different scales, according to the scale of the features that they receive as input. As opposed to the standard RetinaNet, we employ iABN$^{\text{sync}}$ also in this head. The head, depicted in Fig. 3, is composed of two parallel stacks of 3×3 convolutions, and is parametrized by n_a reference bounding box sizes (anchors) per scale level.

Inputs. The inputs are the 5 outputs $\{f_1, \ldots, f_5\}$ of the backbone, where f_i has a spatial resolution of $\mathsf{h}_i \times \mathsf{w}_i$.

Outputs. For each image, and each input tensor f_i, the 2D detection head generates n_a bounding box proposals (one per anchor) for each spatial cell g in the $\mathsf{h}_i \times \mathsf{w}_i$ grid. Each proposal for a given anchor a with size (w_a, h_a) is encoded as a 5-tuple $(\zeta_{\text{2D}}, \delta_u, \delta_v, \delta_w, \delta_h)$ such that

- $p_{\text{2D}} = (1 + e^{-\zeta_{\text{2D}}})^{-1}$ gives the confidence of the 2D bounding box prediction,
- $(u_b, v_b) = (u_g + \delta_u w_a, v_g + \delta_v h_a)$ gives the center of the bounding box with (u_g, v_g) being the image coordinates of cell g, and
- $(w_b, h_b) = (w_a e^{\delta_w}, h_a e^{\delta_h})$ gives the bounding box size.

Fig. 5 gives a visual description of the head's outputs.

Losses. We employ the focal loss [19] to train the bounding box confidence score. This loss takes the following form, for a given cell g and anchor a with target confidence $y \in \{0, 1\}$ and predicted confidence $p \in [0, 1]$:

$$L_{\text{2D}}^{\text{conf}}(p_{\text{2D}}, y) = -\alpha y (1 - p_{\text{2D}})^{\gamma} \log p_{\text{2D}} - \bar{\alpha} \bar{y} p_{\text{2D}}^{\gamma} \log(1 - p_{\text{2D}}) \,,$$

where $\alpha \in [0, 1]$ and $\gamma > 0$ are hyperparameters that modulate the importance of errors and positives, respectively, $\bar{\alpha} = 1 - \alpha$ and $\bar{y} = 1 - y$. The confidence target y does not depend on the regressed bounding box, but only on the cell g and the anchor a. It takes value 1 if the reference bounding box centered in (u_g, v_g) with size (w_a, h_a) exhibits an Intersection-over-Union (IoU) with a ground-truth bounding box larger than a given threshold τ_{iou}. For each cell g and anchor a that matches a ground-truth bounding box $\hat{\boldsymbol{b}}$ with predicted bounding box $\boldsymbol{b} = (u_b - \frac{w_b}{2}, v_b - \frac{h_b}{2}, u_b + \frac{w_b}{2}, v_b + \frac{h_b}{2})$ we consider the following detection loss:

$$L_{\text{2D}}^{\text{bb}}(\boldsymbol{b}, \hat{\boldsymbol{b}}) = 1 - \text{sIoU}(\boldsymbol{b}, \hat{\boldsymbol{b}}) \,, \tag{1}$$

where sIoU represents an extension of the common IoU function, which prevents gradients from vanishing in case of non-overlapping bounding boxes. We call it *signed* IoU function, as, intuitively, it creates negative intersections in case of disjoint bounding boxes (see, Appendix A). In Sec. 5, we discuss a disentangling transformation of the loss in Eq. (1) that allows to isolate the contribution of each network's output to the loss, while preserving the fundamental nature of the loss.

Output Filtering. The dense output of the 2D head is filtered as in [19]: first, detections with scores lower than 0.05 are discarded, then Non-Maxima Suppression (NMS) with IoU threshold 0.5 is performed on the 5000 top-scoring among the remaining ones, and the best 100 are kept.

4.3. 3D Detection Head

The 3D detection head (Fig. 4) regresses a 3D bounding box for each 2D bounding box returned by the 2D detection head (surviving the filtering step). It starts by applying ROIAlign [9] to pool features from FPN into a 14×14 grid for each 2D bounding box, followed by 2×2 average pooling, resulting in feature maps with shape $7 \times 7 \times 128$. The choice of which FPN output is selected for each bounding box $\boldsymbol{b}$ follows the same logic as in [18], namely the features are pooled from the output f_k, where $k = \min(5, \max(1, \lfloor 2 + \log_2(\sqrt{w_b h_b}/224) \rfloor))$. On top of this, two parallel branches of fully connected layers with 512 channels compute the outputs detailed below. Each fully connected layer but the last one per branch is followed by iABN (non-synchronized).

Input. The inputs are a 2D bounding box proposal $\boldsymbol{b}$ returned by the 2D detection head and features f_k from the backbone.

Output. The head returns for each 2D proposal $\boldsymbol{b}$ with center (u_b, v_b) and dimensions (w_b, h_b) a 3D bounding box encoded in terms of a 10-tuple $\boldsymbol{\theta} = (\delta z, \Delta_u, \Delta_v, \delta_W, \delta_H, \delta_D, q_r, q_i, q_j, q_k)$ and an additional output ζ_{3D} such that

- $p_{\text{3D|2D}} = (1 + e^{-\zeta_{\text{3D}}})^{-1}$ represents the confidence of the 3D bounding box prediction given the 2D proposal,
- $z = \mu_z + \sigma_z \delta_z$ represents the depth of the center $\boldsymbol{C}$ of the predicted 3D bounding box, where μ_z and σ_z are given, dataset-wide depth statistics,
- $\boldsymbol{c} = (u_b + \Delta_u, v_b + \Delta_v)$ gives the position of $\boldsymbol{C}$ projected on the image plane (in image coordinates),
- $\boldsymbol{s} = (W_0 e^{\delta_W}, H_0 e^{\delta_H}, D_0 e^{\delta_D})$ is the size of the 3D bounding box, where (W_0, H_0, D_0) is a given, dataset-wide reference size, and
- $\boldsymbol{q} = q_r + q_i \mathtt{i} + q_j \mathtt{j} + q_k \mathtt{k}$ is the quaternion providing the pose of the bounding box with respect to an *allocentric* [13], local coordinate system.

Fig. 5 gives a visual description of the head's outputs.

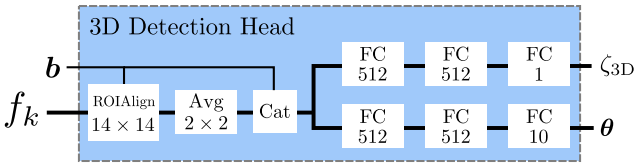

Figure 4: 3D detection head. "FC" rectangles represent fully connected layers. All FCs except the last of each row are followed by iABN.

Losses. Let θ be the 10-tuple representing the regressed 3D bounding box and let $\hat{B} \in \mathbb{R}^{3\times 8}$ be the ground-truth 3D bounding box in camera coordinates. By applying the lifting transformation $\mathcal{F}$ introduced in [23] and reviewed in Appendix B, we obtain the predicted 3D bounding box B given the network's output $\boldsymbol{\theta}$, *i.e.* $B = \mathcal{F}(\boldsymbol{\theta})$. The loss on the 3D bounding box regression is then given by

$$L_{\text{3D}}^{\text{bb}}(B, \hat{B}) = \frac{1}{8}\|B - \hat{B}\|_{\text{H}}\,, \tag{2}$$

where $\|\cdot\|_{\text{H}}$ denotes the Huber loss with parameter δ_H applied component-wise to each element of the argument matrix. The loss for the confidence $p_{\text{3D|2D}}$ about the predicted 3D bounding box is self-supervised by the 3D bounding box loss remapped into a probability range via the transformation $\hat{p}_{\text{3D|2D}} = e^{-\frac{1}{T}L_{\text{3D}}^{\text{bb}}(B,\hat{B})}$, where $T > 0$ is a temperature parameter. The confidence loss for the 3D bounding box is then the standard binary cross entropy loss:

$$L_{\text{3D}}^{\text{conf}}(p_{\text{3D|2D}}, \hat{p}_{\text{3D|2D}}) = -\hat{p}\log p - (1-\hat{p})\log(1-p)\,,$$

where we have omitted the subscripts for the sake of readability. This loss allows to obtain a more informed confidence about the quality of the returned 3D bounding box than just using the 2D confidence. Akin to the 2D case, we employ also a different variant of Eq. (2) that disentangles the contribution of groups of parameters in order to improve the stability and effectiveness of the training. Yet, the confidence computation will be steered by Eq. (2).

Output Filtering. The final output will be filtered based on a combination of the 2D and 3D confidences, following a Bayesian rule. The 3D confidence $p_{\text{3D|2D}}$ is implicitly conditioned on having a valid 2D bounding box and the latter probability is reflected by p_{2D}. At the same time the confidence of a 3D bounding box given an invalid 2D bounding box defaults to 0. Hence, the unconditioned 3D confidence can be obtained by the law of total probability as

$$p_{\text{3D}} = p_{\text{3D|2D}} p_{\text{2D}}\,.$$

This is the final confidence that our method associates to each 3D detection and that is used to filter the predictions via a threshold τ_{conf}. We do not perform further NMS steps on the regressed 3D bounding boxes nor filtering based on 3D prior knowledge (*e.g.* one could reduce false positives by dropping "flying" cars).

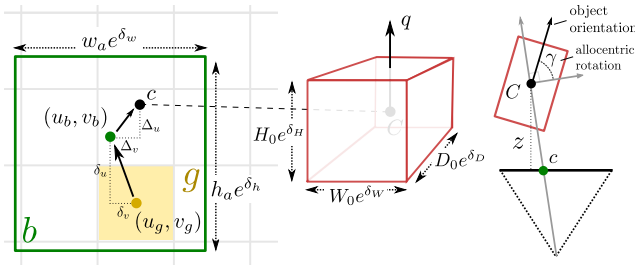

Figure 5: Visualization of the semantics of the outputs of the 2D and 3D detection heads. Left: 2D bounding box regression on image plane. Center: 3D bounding box regression. Right: allocentric angle from bird-eye view.

5. Disentangling 2D and 3D Detection Losses

In this section we propose a transformation that can be applied to the 2D bounding box loss L_{2D}^{bb} and the 3D counterpart L_{3D}^{bb}, as well as a broader set of loss functions. We call it *disentangling* transformation because it isolates the contribution of groups of parameters to a given loss, while preserving its inherent nature. Each parameter group keeps its independent loss term, but they are all made comparable, thus sidestepping the difficulty of finding a proper weighting. While losses that combine parameters in a single term, such as those in Eq. (1) and Eq. (2), are immune to the balancing issue, they might exhibit bad dynamics during the optimization as we will show with a toy experiment. The transformation we propose, instead, retains the best of both worlds.

5.1. Disentangling Transformation

Let $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ be a loss function defined on a space $\mathcal{Y}$ (*e.g.* the space of 3D bounding boxes) such that $L(y, \hat{y}) = 0$ if $\hat{y} = y$. Let $\Theta \subset \mathbb{R}^{\mathsf{d}}$ be a set of possible network outputs that can be mapped to elements of $\mathcal{Y}$ via a function ψ that we assume to be one-to-one. This property holds for 2D bounding boxes via the common 4D parametrization (center + dimensions), as well as for the 3D bounding boxes via the 10D representation described in Sec. 4.3. In the latter case, ψ coincides with the lifting transformation $\mathcal{F}$. Let $\hat{y}$ be a fixed output element (*e.g.* a ground-truth bounding box) and consider a partitioning of the d dimensions of Θ into k groups. To give a concrete example, in case of 2D bounding boxes we can have 2 groups of parameters: one for the dimensions, and one for the center. In the case of $3D$ bounding boxes we consider 4 groups related intuitively to depth, projected center, rotation and dimensions. Given $\boldsymbol{\theta} \in \Theta$ we denote by $\boldsymbol{\theta}_j$ the sub-vector corresponding to the jth group and by $\boldsymbol{\theta}_{-j}$ the sub-vector corresponding to all but the jth group. Moreover, given $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \Theta$, we denote by $\psi(\boldsymbol{\theta}_j, \boldsymbol{\theta}'_{-j})$ the mapping of a parametrization that takes the jth group from $\boldsymbol{\theta}$ and the rest of the parameters from $\boldsymbol{\theta}'$. The disentanglement of loss L given $\hat{y}$, the mapping ψ and

Figure 6: Sample frames from the toy experiment's video on both, entangled (top) and disentangled (bottom) runs. Optimization process at iteration 0 (left) and 150 (right). Green is the ground-truth target. Red is the current prediction. The face with thick lines represents the front of the car. The face with a cross represents the bottom of the car. The birds-eye view on the left shows the projection of the crossed face.

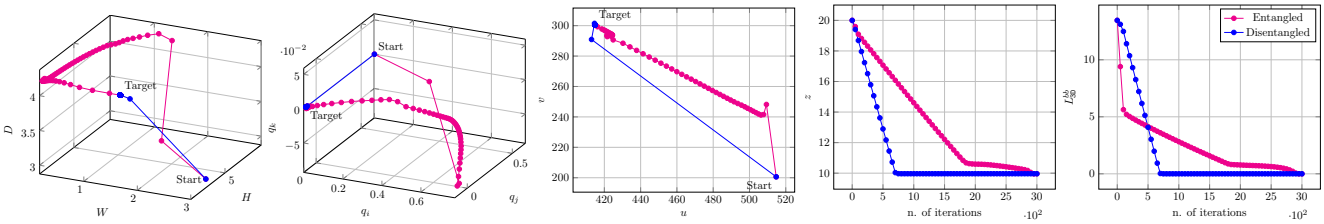

Figure 7: Trajectories of the optimization process for each group of parameters (dimensions, rotation quaternion, projected center, depth), when using the entangled (magenta) and disentangled (blue) 3D detection losses. Left-to-right: trajectories of dimensions, rotation quaternion (last 3 coordinates), projection of the 3D bounding box center on the image and depth of the 3D bounding box center. The last plot shows the evolution of the *entangled* L_{3D}^{bb} loss for both cases.

a decomposition of parameters into k groups is defined as:

$$L_{\text{dis}}(y, \hat{y}) = \sum_{j=1}^{\mathsf{k}} L(\psi(\boldsymbol{\theta}_j, \hat{\boldsymbol{\theta}}_{-j}), \hat{y}) \,,$$

where $\boldsymbol{\theta} = \psi^{-1}(y)$ and $\hat{\boldsymbol{\theta}} = \psi^{-1}(\hat{y})$. The idea behind the transformation is very intuitive besides the mathematical formalism. We simply replicate k times the loss L, each copy having only a group of parameters that can be optimized, the other being fixed to the ground-truth parametrization, which can be recovered via ψ^{-1}. We have applied the disentangling transformation to both the 2D loss in Eq. (1) and to the 3D loss in Eq. (2) and used them to conduct our experiments, unless otherwise stated.

5.2. Explanatory Toy Experiment

The toy experiment consists in comparing the optimization trajectories when we employ the (entangled) 3D object detection loss L_{3D}^{bb} and the disentangled counterpart, which is obtained by applying the disentangling transformation described in Sec. 5. We took a ground-truth detection case from KITTI3D and picked an illustrative initialization for the 3D box for optimization (see, Fig. 6 green and red boxes, respectively).

We perform the experiment using stochastic gradient descent with learning rate 0.001, momentum 0.9 and no weight decay. We run the experiment for 3000 iterations. We report in Fig. 7 (first 4 plots from the left) the trajectories of the optimization process for each group of parameters when the entangled and disentangled losses are used. The parameter groups describe box dimensions, rotation quaternion, projected center of the 3D bounding box on the image, and the depth of the 3D bounding box center. The benefits deriving from the use of the disentangled loss can be clearly seen in the plots. Convergence is much faster and smoother. We can see that the trajectories induced by the entangled loss are suboptimal, since they explore multiple configurations of parameters before approaching the correct one, sometimes with considerable deviations (see, *e.g.* the dimensions of the bounding box). As an example, we report in Fig. 6 (right) the point where the entangled version attains the largest deviation in terms of bounding box dimensions from the ground-truth, which happens at iteration 150, while at this stage the optimization dynamics using the disentangled loss fixed already all parameters but the depth. Despite the quaternion being aligned with the ground-truth rotation axis from the beginning, the optimization dynamics with the entangled loss starts diverging from it, producing unnatural poses and sizes that are not properly penalized by the entangled loss as can be seen by the loss values reported for the two configurations. Such unstable supervision delivered by the entangled loss harms the generalization ca-

pabilities of the network. Interestingly, even though the optimization process that uses the disentangled loss does not directly optimize L_{3D}^{bb}, it can minimize it more quickly than the counterpart directly optimizing it (see, Fig. 7 last).

We provide also a video on our project website that shows the evolution of the optimization process described above. Fig. 6 gives an overview of the first frame (left column). For each optimized loss (entangled on top and disentangled on the bottom) we provide the ground truth 3D bounding box in green and the currently predicted one in red. The faces with thick lines and showing a cross represent the front of the car and the bottom of the car, respectively, while the white line connects the respective centers. We also show the birds-eye view, where we projected the bottom face (the one with the cross) on the ground plane. There we also report the value of the entangled loss L_{3D}^{bb} for both approaches for direct comparison and the iteration number. The video has been rendered with a logarithmic time scale in order to emphasize the initial part of the dynamics, which is also the most informative one.

6. Critical Review on the KITTI3D AP Metric

The KITTI3D benchmark dataset [8] significantly determines developments and general progress on 3D object detection, and has emerged as the most decisive benchmark for monocular 3D detection algorithms like ours. It contains a total of 7481 training and 7518 test images and has no official validation set. However, it is common practice to split the training data into 3712 training and 3769 validation images as proposed in [5], and then report validation results. On the official test split, there is no common agreement which of the training sets to use, but in case validation data is used for snapshot cherry-picking, it is imperative to provide test data scores from the same model.

Each 3D ground truth detection box is assigned to one out of three difficulty classes (*easy, moderate, hard*), and the used 11-point Interpolated Average Precision metric is separately computed on each difficulty class. This metric was originally proposed in [35], and was used in the PASCAL VOC challenges [7] between 2007 and 2010. It approximates the shape of the Precision/Recall curve as

$$\text{AP}|_R = \frac{1}{|R|} \sum_{r \in R} \rho_{interp}(r) \,,$$

averaging the precision values provided by $\rho_{interp}(r)$. In the current setting, KITTI3D applies exactly eleven equally spaced recall levels, *i.e.* $R_{11} = \{0, 0.1, 0.2, \ldots, 1\}$. The interpolation function is defined as $\rho_{interp}(r) = \max_{r':r' \geq r} \rho(r')$, where $\rho(r)$ gives the precision at recall r, meaning that instead of averaging over the actually observed precision values per point r, the maximum precision at recall value greater or equal than r is taken. The recall intervals start at 0, which means that a single, correctly matched prediction (according to the applied IoU level) is sufficient to obtain 100% precision at the bottom-most recall bin. In other words, if for each difficulty level a single, but correct prediction is provided to the evaluation, this produces an $\text{AP}|_{R_{11}}$ score of $1/11 \approx 0.0909$ for the entire dataset, which as shown in our experimental section already outperforms a number of recent methods while it clearly does not properly assess the quality of an algorithm.

In light of KITTI3Ds importance, we propose a simple but effective fix that essentially exploits more of the information provided by the official evaluation server and evaluation scripts. Instead of sub-sampling 11 points from the provided 41 points, we approximate the area under the curve by simply replacing R_{11} with $R_{40} = \{1/40, 2/40, 3/40, \ldots, 1\}$ thus averaging precision results on 40 recall positions but not at 0. This eliminates the glitch encountered at the lowest recall bin, and allows to post-process all currently provided test server results on 2D and 3D AP scores.

7. Experiments on KITTI3D

We focus the validation of our method on the KITTI3D benchmark dataset that we described in Sec. 6, using the 0.7 IoU threshold for calculating AP.

7.1. Pre-processing

We provide some observations about the annotations that can be found in the dataset, and some simple filtering steps that we have applied to the annotations of the training split defined in [5].

DontCare areas. Besides standard classes such as *Car*, *Pedestrian* and *Cyclist*, KITTI3D provides *DontCare* annotations. This class is used to label portions of the image that potentially include positive instances which have not been labeled under the proper class for reasons such as high distance. Accordingly, we avoid harvesting negatives in the 2D detection head if an anchor has IoU above 50% with those areas.

DontCare overlap. Some positive bounding boxes, such as cars that were too near to the camera, have an IoU with a *DontCare* bounding box greater than 50%. We decided to set those bounding boxes as *DontCare*. This adjustment converted 729 cars (5.0%) to *DontCare*.

Full occlusion. Some valid bounding boxes are actually fully occluded by a nearer object. Keeping those bounding boxes as positive instances might harm the learning process, so we decided to delete them. This adjustment deleted 218 (1.5%) cars.

From a total number of 14357 cars that were annotated, the valid number of *Car* bounding boxes was 13410 (93.4%).

7.2. Implementation Details

We give more details about our implementation and instantiation of hyperparameters, in order to enable the reproducibility of our results.

2D Detection Head. For each FPN level f_i and each spatial cell g we employ a total of 15 anchors spanning on five aspect ratios $\{\frac{1}{3}, \frac{1}{2}, 1, 2, 3\}$ and three scales $\{4s_i 2^{\frac{j}{3}} : j \in \{0, 1, 2\}\}$, where s_i is the downsampling factor of f_i. Each anchor is considered positive if its IoU with a ground truth instance is greater than $\tau_{iou} = 0.5$.

3D Detection. We used a reference *Car* size of $W_0 = 1.53m$, $H_0 = 1.63m$, $D_0 = 3.88m$ and depth statistics of $\mu_z = 28.01m$ and $\sigma_z = 16.32m$. We filtered the final 3D detections with a score threshold of $\tau_{\text{conf}} = 0.05$.

Losses. We applied the same weighting policies in all our experiments. We set weight 1.0 to all losses in the 2D detection head and 0.5 to all losses in the 3D detection head. The Huber parameters is set to $\delta_H = 3.0$ and the 3D confidence temperature of $T = 1$.

Optimization. Our training schedule is the same for all experiments, and it does not involve any multi-step or warm-up procedures. We used SGD with a learning rate set at 0.01 and apply weight decay of 0.0001 to all parameters but scale and biases of iABN. We also freeze conv1 and conv2 of ResNet34 in the backbone. We trained with batch size of 96 on 4 NVIDIA V-100 GPUs for a total of 20k iterations, scaling the learning rate by a 0.1 factor at 12k and 16k iterations. Our input resolution is set according to [23]. We applied horizontal flipping as the only form of training-data augmentation. No augmentation was performed for test/validation.

7.3. 2D Detection

In a first set of experiments, we study the signed IoU loss function (Sec. 4.2) in isolation. To do this, we train our backbone + 2D head to perform pure 2D detection of cars in KITTI3D, comparing between the original RetinaNet regression loss, signed IoU and signed IoU with disentanglement. For this simpler task we reduce the training schedule to 3.5k iterations, with learning rate steps after 2k and 3k, while keeping all other parameters as in Sec. 7.2. As shown in Tab. 1, using signed IoU leads to a modest performance increase, which improves considerably when adding disentanglement.

Method	Easy	Moderate	Hard
RetinaNet	87.77	83.74	74.02
RetinaNet + IoU	88.37	84.05	74.32
RetinaNet + IoUDIS	**89.35**	**85.38**	**76.26**

Table 1: Ablation results on KITTI3D with 2D detection networks, $\text{AP}|_{R_{40}}$ scores.

7.4. 3D Detection

In this section we focus on our main task and perform a detailed ablation of our contributions, comparing the results with most relevant state-of-the-art algorithms for monocular 3D detection. Keeping the network architecture and training schedule fixed, we evaluate different loss functions and detection scoring strategies. Following the discussion in Sec. 6, we report both, our revised $\text{AP}|_{R_{40}}$ metric (Tab. 3) and the original $\text{AP}|_{R_{11}}$ (Tab. 5).

Ablation study. First, we turn our attention to the *3D BB* loss in Eq. (2), comparing it to the direct *Regression* of the 10D parameters $\boldsymbol{\theta}$ [23] (first two lines of both tables). Confirming the findings in [23], we observe increased 3D detection scores when tying all parameters together in a single (entangled) loss function in metric space. Perhaps surprisingly, *3D BB* also leads to better 2D detection performance: we suppose this could be due to more informative gradients propagating from the 3D head improving the backbone features. Adding our disentangled 2D detection loss based on the signed IoU (Eq. (1)) and the 3D confidence prediction (Sec. 4.3), consistently improves performance for both *Regression* and *3D BB* (third and fourth lines in the tables). Similarly, applying disentangling to the *3D BB* loss improves 3D detection performance, and has an even larger impact on the 2D side. Bringing all our contributions together leads to noticeable performance increases under all considered metrics (*MonoDIS*). In Tab. 2 we conduct an additional ablation study on the validation set in [5] to assess the importance of the 3D confidence prediction. To this end, we take our best model trained and evaluated with the 3D confidence prediction ($p_{\text{3D}}, \text{AP}|_{R_{xx}}$) and compare against the same model when the 2D confidence is returned ($p_{\text{2D}}, \text{AP}|_{R_{xx}}$) and when it is randomly sampled (random, $\text{AP}|_{R_{xx}}$) . The ability of computing a reliable estimation of the confidence about the prediction is of utmost importance as can be inferred by the drastic drop of performance that we get when replacing p_{3D} with p_{2D}, or with a random confidence. This is a direct consequence of the important role that the returned confidence plays in the AP metric.

Comparison with SOTA. In Tab. 3, 4 and 5 we report validation and test set results, respectively, of many recent monocular 3D detection approaches. When evaluating on the validation set, we consider the split defined in [5], as is done in all the baselines. Please note that the works in [40, 24, 41] are using yet another training/validation split, rendering their results incomparable to ours while yielding numerically comparable ranges to *e.g.* [20]. For the test set, we consider both the split in [5], which is shared with OFTNet [33] and ROI-10D [23], and a larger training split[1], since the setting used for MonoGRNet [29] is not clear. In

[1] https://github.com/MarvinTeichmann/KittiBox

Method, metric	2D detection			3D detection			Bird's eye view		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
p_{3D}, AP$\|_{R_{11}}$	90.23	88.64	79.10	18.05	14.98	13.42	24.26	18.43	16.95
p_{2D}, AP$\|_{R_{11}}$	77.61	82.33	74.52	5.46	4.91	4.27	9.03	7.25	7.05
random, AP$\|_{R_{11}}$	19.01	28.83	31.65	2.70	2.37	1.82	3.82	2.75	2.90
p_{3D}, AP$\|_{R_{40}}$	94.96	89.22	80.58	11.06	7.60	6.37	18.45	12.58	10.66
p_{2D}, AP$\|_{R_{40}}$	81.92	82.91	75.98	5.07	3.72	3.44	8.61	6.73	6.00
random, AP$\|_{R_{11}}$	18.12	28.15	29.88	1.72	1.51	1.36	2.96	2.42	2.58

Table 2: Results on KITTI3D when using p_{2D} or $p_{3D} = p_{3D|2D}p_{2D}$ as the final confidence score to rank predictions. In addition, we report the performance when the confidence is sampled from a uniform distribution.

Tab. 3 we show AP$|_{R_{40}}$ scores[2] for the test set results, and in Tb. 4 the corresponding AP$|_{R_{11}}$ scores. Nonetheless, we would like to stress that the AP$|_{R_{11}}$ is biased by the issue reported in Section 6 and we invite to rather consider AP$|_{R_{40}}$ as the reference metric for fair comparison. With a single exception, our approach beats all baselines on all 3D and bird's eye view metrics, often by a large margin, despite the fact that some of the outperformed methods rely on additional data, such as synthetic images (ROI-10D [23]), or a pre-trained monocular depth prediction network (ROI-10D [23], Xu *et al*. [42]). Interestingly, from the validation set results in Tab. 5, many existing approaches score lower than the "single correct hypothesis" baseline (see Sec. 6) on 3D detection AP$|_{R_{11}}$, highlighting the need for an improved AP metric.

Results on additional KITTI3D classes. In Tab. 6 we provide the AP$|_{R_{11}}$ and AP$|_{R_{40}}$ scores (at IoU treshold 0.5, see official evaluation scripts) obtained on the validation set in [5] for classes *Pedestrian* and *Cyclist* (trained independently). If compared to the results on class *Car*, it can be seen that performances on these two particular classes are in general lower. The performance degradation on classes *Pedestrian* and *Cyclist* compared to *Car* is due to i) the reduced number of annotations which is $\approx 6\times$ and $\approx 20\times$ lower than *Car* for class *Pedestrian* and *Cyclist*, respectively, and ii) the higher impact that errors on localization have on the AP scores since the object xz-extent is typically smaller. For these reasons, similarly to [23, 29, 33, 42], we put a larger focus on class *Car* in the main paper.

Qualitative results. In Fig. 10 we show qualitative results on a set of images taken from the validation set for the classes *Car* (top), *Pedestrian* (middle) and *Cyclist* (bottom). We also provide a video[3] showing detection results obtained on a sequence from the validation set. The structure of the frames is similar to the one in Fig. 10, where detections are shown on the right side and the corresponding birds-eye view on the left. For simplicity, we decided to display all the detections with the same color.

[2]We calculated these from the precision-recall values published in the KITTI3D leaderboard page.

[3]https://research.mapillary.com/publication/MonoDIS

8. Experiments on nuScenes

We conduct additional experiments on the novel *nuScenes* dataset [2].

About the dataset. The nuScenes dataset provides multimodal, street-level data collected with a car equipped with 6 cameras, 1 LiDAR, 5 Radars and IMU. It contains 15h of driving data (242 km at average speed of 16 km/h) covering parts of the areas of Boston (Seaport and South Boston) and Singapore (One North, Holland Village and Queenstown). These two cities have been chosen due to their known dense traffic and highly challenging driving situations and driving routes are selected to capture a diverse set of locations, times and weather conditions. The dataset provides 360°, synchronized sensor coverage, calibration of sensor intrinsics and extrinsics parameters, and objects annotations for 23 different classes from 1000 selected scenes of $20s$ duration each. Annotated objects in the scenes come with a semantic category, 3D bounding box, tracking information, and attributes (visibility, activity and pose) for each frame they occur in.

Detection task. The nuScenes detection tasks requires detecting 10 object classes in terms of full 3D bounding boxes, attributes and velocities. In this work, we will focus on detecting the full 3D bounding box of object belonging to class *car*, because the only available baselines at the time of writing are OFTNet (monocular RGB image-based) and PointPillar [14] (LiDAR-based). Fair comparison can only be made to OFTNet, where results are reported only for category *car* (see, [2]).

Evaluation metric. The authors of nuScenes propose an alternative metric called *nuScenes detection score* (NDS) that combines a measure of the detection performance with quality terms of box location (ATE, average translation error), size (ASE, average scale error), orientation (AOE, average orientation error), attributes (AAE, average attribute error) and velocity (AVE, average velocity error). The detection performance is measured in terms of Average Precision (AP), but with matches determined based on 2D center distance on the ground plane. Also the AP score is calculated as the normalized area under the precision/recall curve by excluding the $[0 - 10\%]$ range. The final score averages

Method	2D detection			3D detection			Bird's eye view		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
Regression	70.10	73.20	66.80	1.30	0.90	0.70	2.60	1.90	1.70
3D BB	74.30	77.10	69.50	3.90	2.70	2.50	6.90	5.10	4.40
Regression w/ IoUDIS, 3DConf	70.10	75.10	66.90	2.60	1.70	1.40	5.40	3.80	3.00
3D BB w/ IoUDIS, 3DConf	**95.10**	88.90	78.60	8.80	6.10	5.00	14.60	10.10	8.30
3D BB w/ disentangling	80.50	80.80	74.40	4.10	3.00	2.70	7.10	5.40	4.80
MonoDIS	94.96	**89.22**	**80.58**	**11.06**	**7.60**	**6.37**	**18.45**	**12.58**	**10.66**
OFTNet [33]	–	–	–	1.61	1.32	1.00	1.28	0.81	0.51
FQNet [20]	**94.72**	**90.17**	76.78	2.77	1.51	1.01	5.40	3.23	2.46
ROI-10D w/ Depth, Synthetic [23]	76.56	70.16	61.15	4.32	2.02	1.46	9.78	4.91	3.74
MonoGRNet [29]	88.65	77.94	63.31	9.61	5.74	4.25	**18.19**	11.17	8.73
MonoDIS	93.11	85.86	73.61	7.03	4.89	4.08	12.18	9.13	7.38
MonoDIS, larger training split	94.61	89.15	**78.37**	**10.37**	**7.94**	**6.40**	17.23	**13.19**	**11.12**

Table 3: $\text{AP}|_{R_{40}}$ scores on KITTI3D: ablation results (white background), test set results of SOTA (grey background) and ours (green background).

Method	2D detection			3D detection			Bird's eye view		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
OFTNet [33]	–	–	–	3.28	2.50	2.27	9.50	7.99	7.51
FQNet [20]	**90.45**	**88.83**	**77.55**	3.48	2.42	1.96	6.51	4.62	3.99
ROI-10D w/ Depth, Synthetic [23]	75.33	69.64	61.18	**12.30**	10.30	9.39	16.77	12.40	11.39
MonoGRNet [29]	87.23	77.46	61.12	11.29	12.90	11.34	**20.55**	16.37	15.16
MonoDIS	89.61	83.80	70.84	8.26	6.15	6.06	13.10	11.12	9.35
MonoDIS, larger training split	90.31	87.58	76.85	11.81	**15.12**	**12.71**	18.88	**19.08**	**17.41**

Table 4: $\text{AP}|_{R_{11}}$ scores on KITTI3D: test set results of SOTA (grey background) and ours (green background).

AP over matching thresholds of $\mathbb{D} = \{0.5, 1, 2, 4\}$ meters and the set of classes $\mathbb{C}$:

$$\text{mAP} = \frac{1}{|\mathbb{C}||\mathbb{D}|} \sum_{c \in \mathbb{C}} \sum_{d \in \mathbb{D}} \text{AP}_{c,d} \,,$$

where $\text{AP}_{c,d}$ is the AP score on class c with matching threshold d.

Obtained results. We present in Fig. 8 the results obtained on the car class in terms of Precision/Recall curves (for all distance thresholds in $\mathbb{D}$), as well as error curves for translation, scale and orientation true positive metrics (at distance threshold 2m), produced by the official nuScenes evaluation scripts. For direct comparison to available OFTNet and PointPillar results from [2], we also provide Tab. 7. It is important to stress that direct comparison is only fair to OFTNet which is also purely image-based, unlike PointPillar, which is LiDAR-based. We are not reporting the NDS score as it also requires predictions for attributes and velocities. Since that would imply modifications of the network design it would also render results inconsistent with those obtained on KITTI3D in Sec. 7.

The results in Tab. 7 show that our approach improves by **42%** over OFTNet (in absolute terms), considering the primary AP metric at a distance threshold of 2m. In addition, MonoDIS improves on all available True Positive metrics over OFTNet and even on 2/3 metrics when compared to PointPillar (LiDAR-based). Despite obtaining better (lower) TP metrics ASE and AOE compared to PointPillar, the main advantage of LiDAR-based methods are shown in their lower translation errors (and therefore also in the corresponding AP scores at various distances). We provide some qualitative results in Fig. 11, demonstrating promising 3D recognition performance without using LiDAR and therefore actively sensed depth information.

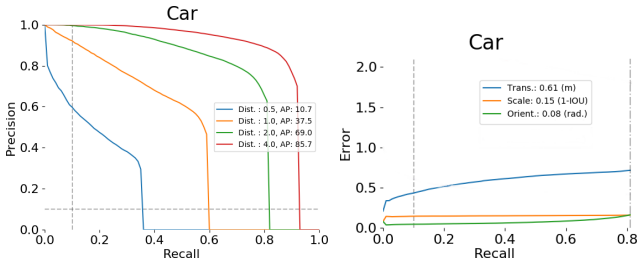

Figure 8: Performance plots for class *Car* in nuScenes. Left: Precision/Recall curves for AP metric at multiple distance thresholds in $\mathbb{D}$. Right: Error/Recall curves for relevant TP errors metrics on translation (ATE), scale (ASE) and orientation (AOE).

9. Conclusions

We proposed a new loss disentangling transformation that allowed us to effectively train a 3D object detection network end-to-end without the need of stage-wise training or warm-up phases. Our solution isolates the contribution

Method	2D detection			3D detection			Bird's eye view		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
Regression	66.50	72.30	66.00	1.60	1.50	1.20	2.70	2.10	2.30
3D BB	70.80	77.10	66.50	4.70	3.00	2.90	7.80	5.40	5.80
Regression w/ IoUDIS, 3DConf	67.20	73.60	65.50	3.20	2.90	2.00	5.80	4.80	4.30
3D BB w/ IoUDIS, 3DConf	90.20	88.40	78.40	15.40	13.60	12.00	20.50	16.20	15.70
3D BB w/ disentangling	76.40	80.30	73.20	4.90	3.40	3.10	7.30	5.70	6.30
MonoDIS	90.23	88.64	79.10	**18.05**	**14.98**	**13.42**	**24.26**	**18.43**	**16.95**
Single correct hypothesis per difficulty	9.09	9.09	9.09	9.09	9.09	9.09	9.09	9.09	9.09
OFTNet [33]	–	–	–	4.07	3.27	3.29	11.06	8.79	8.91
Xu *et al*. [42]	–	–	–	7.85	5.39	4.73	19.20	12.17	10.89
FQNet [20]	–	–	–	5.98	5.50	4.75	9.50	8.02	7.71
Mono3D [4]	**93.89**	**88.67**	**79.68**	2.53	2.31	2.31	5.22	5.19	4.13
Mono3D++ [11]	–	–	–	10.60	7.90	5.70	16.70	11.50	10.10
ROI-10D [23]	78.57	73.44	63.69	10.12	1.76	1.30	14.04	3.69	3.56
ROI-10D w/ Depth [23]	89.04	88.39	78.77	7.79	5.16	3.95	10.74	7.46	7.06
ROI-10D w/ Depth, Synthetic [23]	85.32	77.32	69.70	9.61	6.63	6.29	14.50	9.91	8.73
MonoGRNet [29]	–	–	–	13.88	10.19	7.62	–	–	–
Best in [1]	–	–	–	13.96	7.37	4.54	–	–	–

Table 5: $AP|_{R_{11}}$ scores on KITTI3D (0.7 IoU threshold): Ablation results (white background), val set results of SOTA (grey background).

Method, metric, class	2D detection			3D detection			Bird's eye view		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
MonoDIS, $AP|_{R_{11}}$, pedestrian	72.16	64.93	56.89	10.79	10.39	9.22	11.04	10.94	10.59
MonoDIS, $AP|_{R_{40}}$, pedestrian	72.78	65.56	56.50	3.20	2.28	1.71	4.04	3.19	2.45
MonoDIS, $AP|_{R_{11}}$, cyclist	67.81	49.15	47.26	5.27	4.55	4.55	5.52	4.66	4.55
MonoDIS, $AP|_{R_{40}}$, cyclist	68.12	47.45	45.60	1.52	0.73	0.71	1.87	1.00	0.94

Table 6: Results on the classes *Pedestrian* and *Cyclist* on the KITTI3D validation set (0.5 IoU threshold).

Method	AP_{Car} ↑ [%]				TP_{Car} ↓		
	0.5m	1.0m	**2.0m**	4.0m	ATE [m]	ASE [1-IoU]	AOE [rad]
PointPillar	55.5	71.8	76.1	78.6	0.27	0.17	0.19
OFTNet	–	–	27.0	–	0.65	0.16	0.18
MonoDIS	10.7	37.5	**69.0**	85.7	**0.61**	**0.15**	**0.08**

Table 7: Performance comparison for results on category car in nuScenes dataset [2]. Top row: LiDAR-based PointPillar results (listed for completeness). Bottom: Available OFTNet results vs. MonoDIS.

made by groups of parameters to a given loss into separate terms that retain the same nature of the original loss, thus being compatible without the need of further, cumbersome loss balancing steps. We proposed two further loss functions where i) is based on a novel signed Intersection-over-Union criterion to improve 2D detection results and ii) is used to predict a detection confidence for the 3D bounding box predictions, learned in a self-supervised way. Besides the methodological contributions, we reveal a flaw in the primary detection metric used in KITTI3D, where a single, correctly predicted bounding box yields overall AP scores of 9.09% on validation or test splits. Our simple fix corrects performance results of previously published methods in general, and shows how significantly it was biasing monocular 3D object detection results in particular. In our extensive experimental results and ablation studies we demonstrated the effectiveness of our proposed model, and significantly improved over previous state-of-the-art on both, KITTI3D and the novel nuScenes dataset.

A. Signed Intersection over Union

Let $\hat{\boldsymbol{b}} = (\hat{u}_1, \hat{v}_1, \hat{u}_2, \hat{v}_2)$ and $\boldsymbol{b} = (u_1, v_1, u_2, v_2)$ be two bounding boxes, where (u_1, v_1) denotes the top-left corner and (u_2, v_2) denotes the bottom-right corner. We define the signed intersection-over-union as follows:

$$\text{sIoU}(\boldsymbol{b}, \hat{\boldsymbol{b}}) = \frac{|\boldsymbol{b} \sqcap \hat{\boldsymbol{b}}|_{\pm}}{|\boldsymbol{b}| + |\hat{\boldsymbol{b}}| - |\boldsymbol{b} \sqcap \hat{\boldsymbol{b}}|_{\pm}}, \tag{3}$$

where

$$\boldsymbol{b} \sqcap \hat{\boldsymbol{b}} = \begin{pmatrix} \max(u_1, \hat{u}_1) \\ \max(v_1, \hat{v}_1) \\ \min(u_2, \hat{u}_2) \\ \min(v_2, \hat{v}_2) \end{pmatrix}$$

provides an extended intersection operation between bounding boxes, $|\boldsymbol{b}|$ gives the area of bounding box $\boldsymbol{b}$ and

$$|\boldsymbol{b}|_{\pm} = \begin{cases} +|\boldsymbol{b}| & \text{if } u_2 > u_1 \text{ and } v_2 > v_1, \\ -|\boldsymbol{b}| & \text{otherwise,} \end{cases}$$

gives the *signed* area of $\boldsymbol{b}$, which corresponds to the standard area with positive sign only if the first corner of $\boldsymbol{b}$ is the top-left one, while the second corner is the bottom-right one. To give a better intuition we provide some examples in Fig. 9, where green and red colors encode positive and negative areas, respectively: Left-to-right, the first two examples boil down to standard IoU yielding positive values, while the last ones are examples yielding negative values. The sIoU score is bounded in $[-1, 1]$.

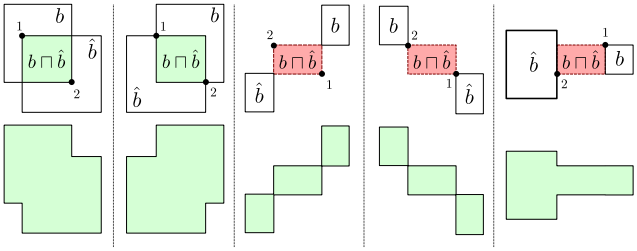

Figure 9: Five examples of computation of the proposed signed IoU. Top: Colored areas represent the numerator of the sIoU formula, where green denotes positive area, red denotes negative area; numbers represent the corner ordering. Bottom: Areas represent the denominator, which is always positive.

B. Lifting Transformation

We review the lifting transformation used in [23]. Let $\boldsymbol{\theta}$ be the 10D network's output from which we compute the depth z of the 3D bounding box's center, its projection on the image place $\boldsymbol{c} = (u_c, v_c)$, the dimensions of the 3D bounding box $\boldsymbol{s} = (W, H, D)$ and the unit quaternion $\boldsymbol{q}$ as described in Sec. 4.3 of the main paper. Let K be the 3×3 matrix of intrinsics with entries:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

and let

$$\boldsymbol{C} = \left(\tfrac{u_c - c_x}{f_x} z, \quad \tfrac{v_c - c_y}{f_y} z, \quad z \right)^\top = (C_x, C_y, C_z)^\top$$

be the position of the center of the 3D bounding box. The lifting transformation is defined as:

$$\mathcal{F}(\boldsymbol{\theta}) = \frac{1}{2} R_{\boldsymbol{q_c}} S \, B_0 + \boldsymbol{C}$$

where B_0 holds the corners of the unit cube $[-1, 1]^3$, S is the diagonal matrix with entries $\boldsymbol{s}$, and $R_{\boldsymbol{q}_c}$ is the 3×3 rotation matrix corresponding to quaternion

$$\boldsymbol{q}_c = \boldsymbol{q} \left[\cos \frac{\beta}{2} + sin \frac{\beta}{2} \mathtt{j} \right]$$

with $\beta = \tan^{-1}(\frac{C_x}{C_z})$.

References

[1] I. Barabanau, A. Artemov, E. Burnaev, and V. Murashkin. Monocular 3d object detection via geometric reasoning on keypoints. *CoRR*, abs/1905.05618, 2019. 2, 11

[2] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuScenes: A multimodal dataset for autonomous driving. *CoRR*, abs/1903.11027, 2019. 2, 9, 10, 11

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

[4] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3d object detection for autonomous driving. In *(CVPR)*, 2016. 2, 11

[5] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun. 3d object proposals for accurate object class detection. In *(NIPS)*, 2015. 2, 3, 7, 8, 9

[6] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In *(CVPR)*, July 2017. 3

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal visual object classes (VOC) challenge. *(IJCV)*, 88(2):303–338, 2010. 2, 7

[8] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *(CVPR)*, 2012. 1, 7

[9] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. In *(ICCV)*, 2017. 4

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 3

[11] T. He and S. Soatto. Mono3d++: Monocular 3d vehicle detection with two-scale 3d hypotheses and task priors. *CoRR*, abs/1901.03446, 2019. 2, 3, 11

[12] 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 *(ICCV)*, October 2017. 2

[13] A. Kundu, Y. Li, and J. M. Rehg. 3D-RCNN: Instance-level 3d object reconstruction via render-and-compare. In *(CVPR)*, June 2018. 2, 3, 4

[14] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *(CVPR)*, 2019. 9

[15] H. Law and J. Deng. Cornernet: Detecting objects as paired keypoints. In *(ECCV)*, September 2018. 1

[16] P. Li, X. Chen, and S. Shen. Stereo r-cnn based 3d object detection for autonomous driving. In *(CVPR)*, 2019. 3

[17] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *(CVPR)*, 2019. 1

[18] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016. 3, 4

[19] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017. 1, 3, 4

[20] L. Liu, J. Lu, C. Xu, Q. Tian, and J. Zhou. Deep fitting degree scoring network for monocular 3d object detection. *CoRR*, abs/1904.12681, 2019. 2, 8, 10, 11

[21] L. Liu, W. Ouyang, X. Wang, P. W. Fieguth, J. Chen, X. Liu, and M. Pietikäinen. Deep learning for generic object detection: A survey. *CoRR*, abs/1809.02165, 2018. 1

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *(ECCV)*, 2016. 1

[23] F. Manhardt, W. Kehl, and A. Gaidon. Roi-10d: Monocular lifting of 2d detection to 6d pose and metric shape. In *(CVPR)*, 2019. 1, 2, 3, 5, 8, 9, 10, 11, 12

[24] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3d bounding box estimation using deep learning and geometry. In *(CVPR)*, July 2017. 2, 8

[25] K. S. Muhammad Zeeshan Zia, Michael Stark. Are cars just 3d boxes? jointly estimating the 3d shape of multiple objects. In *(CVPR)*, 2014. 3

[26] J. K. Murthy, G. V. S. Krishna, F. Chhaya, and K. M. Krishna. Reconstructing vehicles from a single image: Shape priors for road scene understanding. In *(ICRA)*, 2017. 3

[27] S. Pillai, R. Ambrus, and A. Gaidon. Superdepth: Self-supervised, super-resolved monocular depth estimation. In *(ICRA)*, 2019. 2

[28] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *(CVPR)*, June 2018. 3

[29] Z. Qin, J. Wang, and Y. Lu. Monogrnet: A geometric reasoning network for 3d object localization. In *(AAAI)*, 2019. 1, 2, 8, 9, 10, 11

[30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *(CVPR)*, June 2016. 1

[31] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *(CVPR)*, 2017. 1

[32] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *(NIPS)*, 2015. 1

[33] T. Roddick, A. Kendall, and R. Cipolla. Orthographic feature transform for monocular 3d object detection. *CoRR*, abs/1811.08188, 2018. 2, 8, 9, 10, 11

[34] S. Rota Bulò, L. Porzi, and P. Kontschieder. In-place activated batchnorm for memory-optimized training of DNNs. In *(CVPR)*, 2018. 3

[35] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986. 2, 7

[36] S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *(CVPR)*, 2019. 1, 3

[37] K. Shin, Y. P. Kwon, and M. Tomizuka. Roarnet: A robust 3d object detection based on region approximation refinement. *CoRR*, abs/1811.03818, 2018. 1, 3

[38] Z. Wang and K. Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. *CoRR*, abs/1903.01864, 2019. 1

[39] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer. Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In *(CVPR) Workshops*, July 2017. 1

[40] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Data-driven 3d voxel patterns for object category recognition. In *(CVPR)*, 2015. 8

[41] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Subcategory-aware convolutional neural networks for object proposals and detection. In *(WACV)*, 2017. 8

[42] B. Xu and Z. Chen. Multi-level fusion based 3d object detection from monocular images. In *(CVPR)*, June 2018. 2, 9, 11

Figure 10: Example results for classes *Car* (top), *Pedestrian* (middle) and *Cyclist*(bottom) with corresponding birds-eye view.

Figure 11: Example results for class *Car* on nuScenes dataset for images taken at different weather and illumination conditions.