Disentangling Monocular 3D Object Detection

Andrea Simonelli[△,⋆], Samuel Rota Bulò[△], Lorenzo Porzi[△], Manuel López-Antequera[△], Peter Kontschieder[△]
[△]Mapillary Research [⋆]University of Trento, Fondazione Bruno Kessler
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 and set a new state-of-the-art on the KITTI3D Car class.

1. Introduction

Recent developments in object recognition [17] 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 [26, 18, 25, 16, 12].

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 [14, 33, 30] 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 [24]). 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 [19, 24] 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 [6], and in

particular its *Car* category [19, 24, 27, 34]. 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 [29] used in the PASCAL VOC2007 [5] 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 [3, 2, 9, 34].

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 Car category of the KITTI3D dataset. 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 [20] 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 [10] 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 [27] 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 [2] 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 in input. The basic variant (w/o using depth) of ROI-10D [19] proposes a novel loss to lift 2D detection, orientation and scale into 3D space that can be trained in an end-to-end fashion. MonoGRNet [24] 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 [11, 19] way). It is relevant to mention that [24] 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.

Including depth. An expansion stage of ROI-10D [19] takes advantage of depth information provided by SuperDepth [22], which itself is learned in a self-supervised manner. In [34], 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 [11] 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 [35], the recognition task is tackled by jointly reasoning

about the 3D shape of multiple objects. Deep-MANTA [1] uses 3d CAD models and annotated 3d parts in a coarse-to-fine localization process. The work in [21] encodes shape priors using keypoints for recovering the 3D pose and shape of a query object. In Mono3D++ [9], 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 [3] exploits stereo images and prior knowledge about the scene to directly reason in 3D. Stereo R-CNN [13] 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 [4], 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 [23] directly operates on LIDAR point clouds and aligns candidate points provided from corresponding 2D detections for estimating the final, amodal 3D bounding boxes. PointRCNN [30] describes a 2-stage framework where the first stage provides bottom-up 3D proposals and the second stage refines them in canonical coordinates. RoarNet [31] 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 [19]. 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 [8] with a Feature Pyramid Network (FPN) [15] built on top of it. The FPN network has the same structure as in [16] 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 [28]. 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).

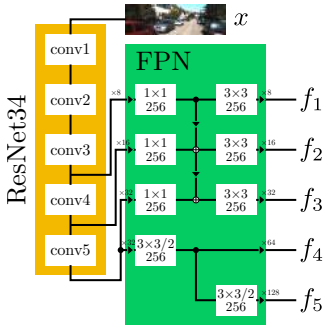

Figure 2: Backbone architecture. Rectangles in the "FPN" block represent convolutions followed by iABN$^{\text{sync}}$.

4.2. 2D Detection Head

We consider the head of the single-stage 2D detector implemented in RetinaNet [16], which applies a detection 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

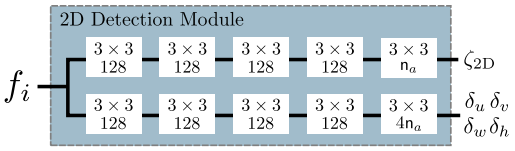

Figure 3: 2D detection module. Rectangles represent convolutions. All convolutions but the last per row are followed by iABN$^{\text{sync}}$.

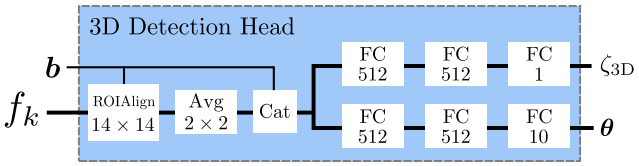

Figure 4: 3D detection head. "FC" rectangles represent fully connected layers. All FCs except the last of each row are followed by iABN.

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 $(\zeta_{2D}, \delta_u, \delta_v, \delta_w, \delta_h)$ such that

- $p_{2D} = (1 + e^{-\zeta_{2D}})^{-1}$ 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 [16] 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 $p \in [0, 1]$:

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

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:

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

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 (please refer to [32] for further discussions). 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 [16]: 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 [7] 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 [15], 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

- $p_{3D|2D} = (1 + e^{-\zeta_{3D}})^{-1}$ 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* [11], local coordinate system.

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

Losses. Let $\boldsymbol{\theta}$ 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 [19] and reviewed in [32], given the network's output $\boldsymbol{\theta}$ we obtain the predicted 3D bounding box B, *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).

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.

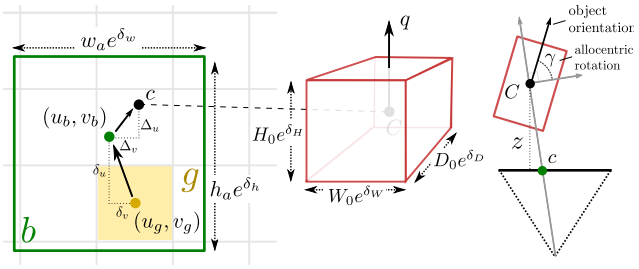

Figure 5: Visualization of the semantics of the outputs of the 2D and 3D detection heads. Left: 2D bounding box regression on image plane. Center: 3D bounding box regression. Right: allocentric angle from bird-eye view.

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 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

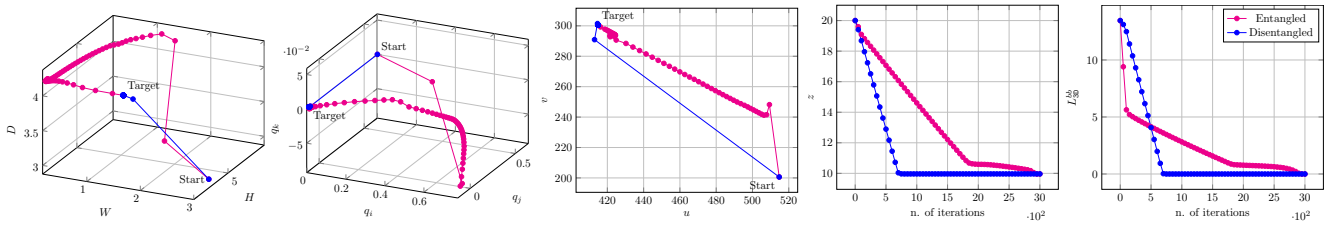

Figure 6: Trajectories of the optimization process for each group of parameters (dimensions, rotation quaternion, projected center, depth), when using the entangled (magenta) and disentangled (blue) 3D detection losses. Left-to-right: trajectories of dimensions, rotation quaternion (last 3 coordinates), 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.

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.

Explanatory Toy Example. We conduct a toy experiment where we fix a ground-truth 3D bounding box from the KITTI3D training set and optimize the 10 parameters $\boldsymbol{\theta}$ directly using Stochastic Gradient Descent (SGD) using both the 3D bounding box loss from Eq. (2) (called *entangled loss*) and the disentangled counterpart that we obtain when grouping together the parameters related to dimensions, rotation, projection of the center, and depth of the center. We start both optimizations from the same perturbed 3D bounding box and report in Fig. 6 the trajectories that we obtain in terms of bounding box dimensions, the last 3 quaternion components, image coordinates of the projected bounding box center and the depth of the center. We indicate the initial and target values for each group of parameters, and mark with a bullet each iteration of the optimization process. As we can see, the use of the disentangled loss leads to more efficient and stable trajectories than the entangled loss. Notably, the rotation parameters converge almost immediately with the disentangled loss, while they follow a long and convoluted trajectory when optimizing the entangled one. Something similar also happens with the bounding box dimensions. In particular, the entangled loss optimization process first reduces the distance to the target by flattening some dimensions (the height in this case). This flattening persists until the target and predicted boxes start overlapping, after which the optimization dynamics can finally converge to the target dimensions. This sub-optimal behaviour is entirely avoided by the disentangled loss, since the predicted and target boxes are, by construction, always centered. More details about this toy experiment are provided in [32].

6. Critical Review on the KITTI3D AP Metric

The KITTI3D benchmark dataset [6] 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 [3], 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 [29], and was used in the PASCAL VOC challenges [5] 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.* $R_{11} = \{0, 0.1, 0.2, \dots, 1\}$. 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

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**

Table 1: $AP|_{R_{40}}$ validation set ablation results on KITTI3D (0.7 IoU threshold).

Method	2D detection			3D detection			Bird's eye view		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
OFTNet [27]	–	–	–	1.61	1.32	1.00	1.28	0.81	0.51
ROI-10D w/ Depth, Synthetic [19]	76.56	70.16	61.15	4.32	2.02	1.46	9.78	4.91	3.74
MonoGRNet [24]	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 2: $AP|_{R_{40}}$ test set SOTA results on KITTI3D (0.7 IoU threshold)

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

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. Implementation Details

In this section we provide details about our implementation and instantiation of hyperparameters.

2D and 3D Detection Heads. For details regarding the *FPN*, 2D anchors as well as *Car* class reference size and depth statistics, please refer to [32].

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 [19]. We applied horizontal flipping as the only form of training-data augmentation. No augmentation was performed for test/validation.

7.2. 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. 7.1. As shown in Tab. 3, 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 3: Ablation results on KITTI3D with 2D detection networks, $AP|_{R_{40}}$ scores.

7.3. 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 $AP|_{R_{40}}$ metric (Tab. 1 and 2) and the original $AP|_{R_{11}}$ (Tab. 4).

Ablation study. First, we make a comparison between a direct *regression* of the the 10D parameters $\boldsymbol{\theta}$ [19] with the

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
Single correct hypothesis per difficulty	9.09	9.09	9.09	9.09	9.09	9.09	9.09	9.09	9.09
OFTNet [27]	–	–	–	4.07	3.27	3.29	11.06	8.79	8.91
Xu *et al*. [34]	–	–	–	7.85	5.39	4.73	19.20	12.17	10.89
Deep3DBox [20]	–	–	–	5.85	4.10	3.84	9.99	7.71	5.30
Mono3D [2]	**93.89**	**88.67**	**79.68**	2.53	2.31	2.31	5.22	5.19	4.13
Mono3D++ [9]	–	–	–	10.60	7.90	5.70	16.70	11.50	10.10
ROI-10D [19]	78.57	73.44	63.69	10.12	1.76	1.30	14.04	3.69	3.56
ROI-10D w/ Depth [19]	89.04	88.39	78.77	7.79	5.16	3.95	10.74	7.46	7.06
ROI-10D w/ Depth, Synthetic [19]	85.32	77.32	69.70	9.61	6.63	6.29	14.50	9.91	8.73
MonoGRNet [24]	–	–	–	13.88	10.19	7.69	–	–	–
MonoDIS	90.23	88.64	79.10	**18.05**	**14.98**	**13.42**	**24.26**	**18.43**	**16.95**

Table 4: $AP|_{R_{11}}$ validation set scores on KITTI3D (0.7 IoU): ablation results (top), SOTA results (bottom).

3D BB loss in Eq. (2). The results of this comparison can be found in 1^{st} and 2^{nd} row of Tab. 1 and 4. Confirming the findings in [19], we observe increased 3D detection scores when tying all parameters together in the (entangled) *3D BB* loss function in metric space. Perhaps surprisingly, this loss 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 *IoUDIS* based on the signed IoU (Eq. (1)) and the 3D confidence prediction *3DConf* (Sec. 4.3), consistently improves performance for both *regression* and *3D BB* (3^{rd} and 4^{th} row). Similarly, applying *disentangling* to the *3D BB* loss improves 3D detection performance, and has an even larger impact on the 2D side (5^{th} row). Finally, bringing all the contributions together in our method *MonoDIS* leads to noticeable performance increases under all considered metrics (last row in Tab. 1 and 4).

Comparison with SOTA. In Tab. 2 and 4 we report test and validation set results of many recent monocular 3D detection approaches. When evaluating on the validation set, we consider the split defined in [3], as is done in all the baselines. For the test set, we consider both the split in [3], which is shared with OFTNet [27] and ROI-10D [19], and a larger training split[1], since the setting used for MonoGRNet [24] is not clear. For the sake of space, we only show $AP|_{R_{40}}$ scores[2] for the test set results, and report the corresponding $AP|_{R_{11}}$ scores in [32]. With a single exception, our approach beats all baselines on all 3D and bird's eye view metrics, often by a large margin. Note that some of the outperformed methods rely on additional data, such as synthetic images (ROI-10D [19]), or a pre-trained monocular depth prediction network (ROI-10D [19], Xu *et al*. [34]). Interestingly, 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.

8. 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 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 improve over previous state-of-the-art.

Acknowledgments

We would like to thank Fabian Manhardt, Wadim Kehl and Adrien Gaidon at Toyota Research Institute for helpful discussions.

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

[2] Calculated from the PR-curves in the KITTI3D leaderboard page.

References

[1] Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Celine 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)*, July 2017. 3

[2] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *(CVPR)*, 2016. 2, 8

[3] 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. 2, 3, 6, 8

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

[5] 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, 6

[6] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The kitti vision benchmark suite. In *(CVPR)*, 2012. 1, 6

[7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *(ICCV)*, 2017. 4

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 3

[9] Tong He and Stefano Soatto. Mono3d++: Monocular 3d vehicle detection with two-scale 3d hypotheses and task priors. *CoRR*, abs/1901.03446, 2019. 2, 3, 8

[10] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *(ICCV)*, October 2017. 2

[11] Abhijit Kundu, Yin Li, and James M. Rehg. 3D-RCNN: Instance-level 3d object reconstruction via render-and-compare. In *(CVPR)*, June 2018. 2, 4

[12] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *(ECCV)*, September 2018. 1

[13] Peiliang Li, Xiaozhi Chen, and Shaojie Shen. Stereo r-cnn based 3d object detection for autonomous driving. In *(CVPR)*, 2019. 3

[14] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *(CVPR)*, 2019. 1

[15] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016. 3, 4

[16] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017. 1, 3, 4

[17] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul W. Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *CoRR*, abs/1809.02165, 2018. 1

[18] 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. 1

[19] Fabian Manhardt, Wadim Kehl, and Adrien Gaidon. Roi-10d: Monocular lifting of 2d detection to 6d pose and metric shape. In *(CVPR)*, 2019. 1, 2, 3, 5, 7, 8

[20] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *(CVPR)*, July 2017. 2, 8

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

[22] Sudeep Pillai, Rares Ambrus, and Adrien Gaidon. Superdepth: Self-supervised, super-resolved monocular depth estimation. In *(ICRA)*, 2019. 2

[23] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *(CVPR)*, June 2018. 3

[24] Zengyi Qin, Jinglu Wang, and Yan Lu. Monogrnet: A geometric reasoning network for 3d object localization. In *(AAAI)*, 2019. 1, 2, 7, 8

[25] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *(CVPR)*, June 2016. 1

[26] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *(NIPS)*, 2015. 1

[27] Thomas Roddick, Alex Kendall, and Roberto Cipolla. Orthographic feature transform for monocular 3d object detection. *CoRR*, abs/1811.08188, 2018. 2, 7, 8

[28] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. In-place activated batchnorm for memory-optimized training of DNNs. In *(CVPR)*, 2018. 3

[29] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986. 2, 6

[30] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *(CVPR)*, 2019. 1, 3

[31] Kiwoo Shin, Youngwook Paul Kwon, and Masayoshi Tomizuka. Roarnet: A robust 3d object detection based on region approximation refinement. *CoRR*, abs/1811.03818, 2018. 3

[32] Andrea Simonelli, Samuel Rota Bulò, Lorenzo Porzi, Manuel Lòpez-Antequera, and Peter Kontschieder. Disentangling monocular 3d object detection. *CoRR*, abs/1905.12365, 2019. 4, 5, 6, 7, 8

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

[34] Bin Xu and Zhenzhong Chen. Multi-level fusion based 3d object detection from monocular images. In *(CVPR)*, June 2018. 2, 8

[35] Muhammad Zeeshan Zia, Michael Stark, and Konrad Schindler. Are cars just 3d boxes? Jointly estimating the 3d shape of multiple objects. In *(CVPR)*, 2014. 2