Dynamic Depth Fusion and Transformation for Monocular 3D Object Detection

Erli Ouyang[1], Li Zhang[1⋆], Mohan Chen[1], Anurag Arnab[2], and Yanwei Fu[1⋆⋆]

[1] School of Data Science, and MOE Frontiers Center for Brain Science,
Shanghai Key Lab of Intelligent Information Processing, Fudan University
[2] University of Oxford
{eouyang18,lizhangfd,mhchen19,yanweifu}@fudan.edu.cn

Abstract. Visual-based 3D detection is drawing a lot of attention recently. Despite the best efforts from the computer vision researchers visual-based 3D detection remains a largely unsolved problem. This is primarily due to the lack of accurate depth perception provided by LiDAR sensors. Previous works struggle to fuse 3D spatial information and the RGB image effectively. In this paper, we propose a novel monocular 3D detection framework to address this problem. Specifically, we propose to primary contributions: (i) We design an *Adaptive Depth-guided Instance Normalization* layer to leverage depth features to guide RGB features for high quality estimation of 3D properties. (ii) We introduce a *Dynamic Depth Transformation* module to better recover accurate depth according to semantic context learning and thus facilitate the removal of depth ambiguities that exist in the RGB image. Experiments show that our approach achieves state-of-the-art on KITTI 3D detection benchmark among current monocular 3D detection works.

Keywords: 3D object detection, Monocular

1 Introduction

3D object detection from images plays an essential role in self-driving cars and robotics. Powered by the effective deep point clouds processing techniques [1, 2], recent LiDAR-based 3D detectors [3–8] have achieved superior performance through exploiting accurate depth information scanned by sensors. However, LiDAR is too expensive for some low cost scenarios and has a limited perception range, i.e., usually less than $100m$. On the other hand, 3D from 2D is a fundamentally ill-posed problem, and estimating 3D bounding boxes from images remains a challenging task, due to the difficulty in drawing missing spatial information in 2D images. In spite of this, recent image-based 3D detectors have made some progress with the help of carefully designed network architectures.

⋆ The first two authors contributed equally to this paper.

⋆⋆ Corresponding author. This work was supported in part by NSFC Projects (U62076067) , Science and Technology Commission of Shanghai Municipality Projects (19511120700, 19ZR1471800).

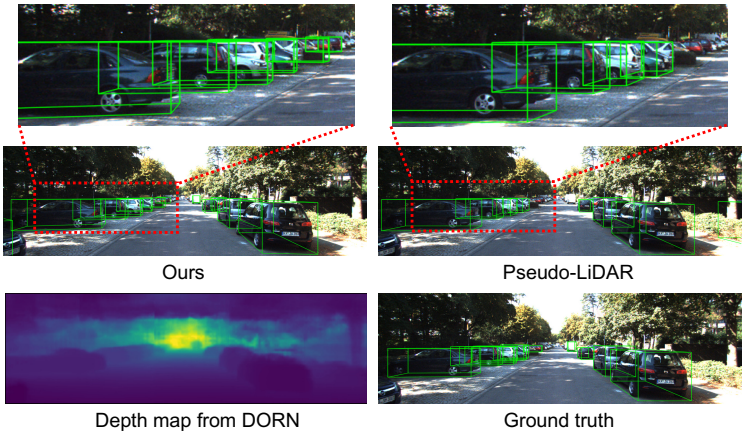

Fig. 1. We propose a novel monocular 3D detection approach. Pseudo-LidAR point cloud-based methods, i.e., Pseudo-LidAR [19], rely much on the quality of depth map (DORN [23]) and our methods fuse depth information and color context for more accurate object 3D properties. Our approach also handle occlusion issue well. (As shown in zoomed in region, where Pseudo-LidAR is affected by serious occlusion problem.)

However, there still exists a huge performance gap between them and LiDAR-based approaches due to the ambiguities of 2D color images. Thus, predicted depth maps are introduced to help resolve context vagueness.

Some early monocular image 3D detection approaches [9–11] either utilize additional input features for more context information such as instance/semantic segmentation [12–18] or directly regress the 3D bounding boxes by 2D convolutional layers. However, it is still hard for them to recover 3D from 2D inputs, which leads to relatively poor results. Recent work [19, 20] transfers the generated depth map into point clouds, and show that the depth data representation matters in the 3D detection task. However, they are sensitive to input depth quality and the procedure is complex as a result of point clouds processing, e.g., segmenting an object's point cloud from its surroundings.

The dense disparity (inverse of depth) map could be inferred by stereo matching equipped with convolutional neural networks, which motivates some stereo-based 3D detection works [21, 22]. However, their accuracy still falls behind LiDAR-methods and camera calibration is also needed, i.e., stereo cameras must be maintained at the same horizontal level. Therefore, monocular methods [19, 20, 9–11] fit in more various scenarios where stereo is not available or practical.

In this paper, we propose a novel image-based 3D detection framework, aiming to address the following key issues in monocular 3D detection field: (1) Inefficient utilization for generated depth maps. For methods [19, 20] using pseudo-LiDAR point clouds, they rely heavily on the accuracy of depth maps. Moreover, depth maps generated from state-of-the-art deep networks still tend to be blurry on the objects boundary and thus re-projected point clouds are noisy,

which makes results sensitive. (2) Inaccurate spatial location estimation for occluded objects. Occlusion happens in typical autonomous driving scenes, and these cases affect detector performance greatly because it is difficult for traditional 2D convolution to capture correct depth information of occluded objects. To solve these aforementioned problems, we propose two effective modules respectively for each of the issues as shown in figure 1. Specifically, we first propose an Adaptive Depth-guided Instance Normalization (AdaDIN) layer to fuse depth features and color features in a more effective way, where the depth features are utilized as an adaptive guidance to normalize color features to recover hidden depth message from the 2D color map. Secondly, we design a novel Dynamic Depth Transformation (DDT) module to address the object occlusion problem, in which we sample and transfer depth values dynamically from tje depth map in the target objects' region by using deformable kernels generated over fused features.

To summarize, this works makes the following contributions:

- We propose an AdaDIN layer, where color features are adaptively normalized by depth feature to recover 3D spatial information.
- We design a novel Dynamic Depth transformation module to sample depth value from target region to determine object spatial location properly.
- Evaluation on KITTI datasets [24] shows that our proposed method achieves the state-of-the-art among all monocular approaches on 3D detection and Bird's eye view detection.

2 Related work

For 3D objects detection task, the methods could be grouped into two classes: LiDAR-based and image-based. Here we briefly review relevant works.

Image-based 3D object detection. For lack of accurate depth information, 3D object detectors using only monocular/stereo image data is generally worse than those using LiDAR data. For monocular input, early works [9, 10] take a strategy of aligning 2D-3D bounding boxes, predicting 3D proposals based on the 2D detection results and additional features extracted from segmentation, shape, context and location. Since the image-based 3D detection is an ill-posed problem, more recent works [25–27] utilize objects prior like objects shape and geometry constraints to predict the results. GS3D [25] generates refined detection results by an estimated coarse basic cuboid from 2D results. M3D-RPN[26] is a one stage detector, where a depth-aware convolutional layer is designed to learn features related to depth. In contrast, our approach adopt a depth map estimated from monocular as well as color image to fully utilize depth information for better results.

As for stereo, there are a small number of arts compared with monocular so far. 3DOP [21] generates 3D box proposals by exploiting object size priors, ground plane and a variety of depth informed features (e.g., point cloud density). Finally they combine a CNN to score the proposal boxes. Stereo R-CNN [22]

first estimates 2D bounding box and key-point information and solves the 3D boxes by optimizing a group of geometry constraints equations.

Nevertheless, the traditional 2D convolution does not have enough capability to resolve the 3D spatial message from 2D features, and there is no effective signal transformation ways from depth to color image, which limits the 3D detection performance. Pseudo-LiDAR [19] brings another important option to imaged-based detectors, in which the generated depth map from monocular are re-projected into point clouds and then existing LiDAR-based approaches are applied to point clouds data for 3D results. AM3D [20] further aggregates RGB information into re-projected point clouds for a higher performance. However, point clouds-based methods are sensitive to the quality of input depth maps. In contrast, we normalize color features with depth features, which helps 3D spatial information transfer from depth to color.

LiDAR-based 3D object detection. Most of state-of-the-art 3D object detection methods use LiDAR data. VoxelNet [3] learns a feature representation from point clouds and predict the results. The general point clouds processing architectures [1, 2] provide the basic tools for LiDAR-based approaches [4, 5] to generate accurate 3D proposals. However, the high price device and large space consumption limit LiDAR-based methods in many scenarios. In this paper, our proposed method takes easily available monocular image and depth map estimated from the same color image as input to produce superior 3D detection results.

3 Methodology

We describe our proposed one-stage monocular 3D detection method in this section. Compared with two-stage approach [20] that also takes the monocular input and depth map as input, our method facilitates a simplified detection procedure while also achieving a higher performance. We first introduce our overall framework, and then we give the details of each key module of our approach.

3.1 Approach overview

The overall framework of our approach is shown in figure 2. The network mainly consists of these modules: image and depth feature extractors, Adaptive Depth-guide instance normalization, Dynamic Depth transformation and 3D detection heads. Our network takes monocular RGB image $\mathcal{I} \in \mathbb{R}^{H \times W \times 3}$ and the corresponding generated depth map $\mathcal{D} \in \mathbb{R}^{H \times W \times 1}$ as input, then extracts the features of both depth map and image, and the depth map feature is utilized to guide feature representation of RGB image by our Depth-guide instance normalization module, which could effectively transfer the depth message to the color feature for accurate 3D bounding box estimation. Afterwards, the depth map is further transformed by our Dynamic Depth transformation to solve occlusion issue which often occurs in autonomous scenes. Our network outputs are generated from 5 main independent heads (shown in figure 2) and 2 center point offsets heads,

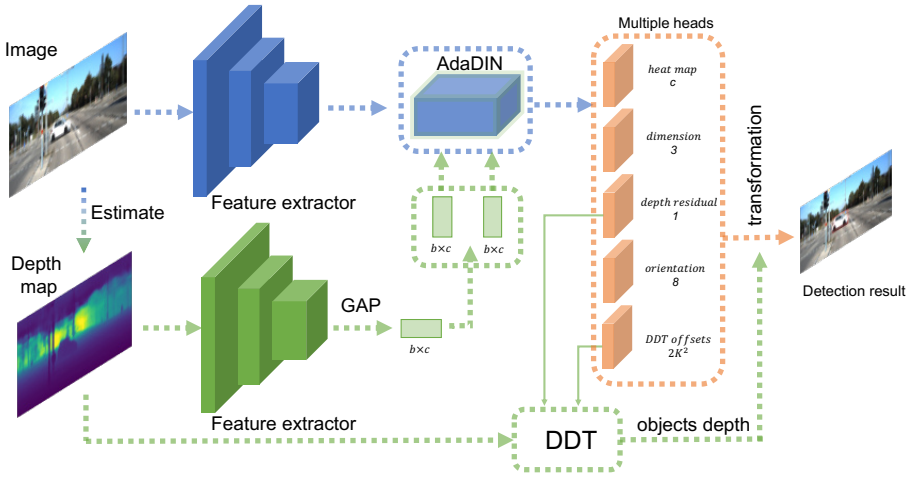

Fig. 2. Overall framework of proposed approach. The color image and the depth map are feed into two feature extractor, and then AdaDIN layer is applied to fuse depth information into color context feature. Multi-heads are attached for 3D bounding box estimation with the help of DDF which address occlusion issue effectively. Note that for simplicity, we omit the two offsets heads (2D center offset and 2D-3D offset) in above figure. The numbers below the multiple heads names are channels of each head.

and each main head stands for a property of 3D bounding box, e.g., dimension, location, rotation, object center position and its depth. During testing, we parse all the outputs of multiple heads to obtain the final 3D objects bounding boxes as described in section 3.4.

3.2 Adaptive depth-guided instance normalization

In this section, we introduce our Adaptive Depth-guided Instance Normalization (AdaDIN).

AdaDIN layer is designed to normalize color features by exploiting depth map features. Due to the lack of depth information, it is hard for color feature to estimate 3D properties of objects (i.e., 3D dimension and depth), and how to fuse depth information from depth feature into color features effectively is a key issue for monocular 3D detection. In this work, inspired by [28, 29], we normalize the color feature across the spatial dimensions independently for each channel and instance, then the normalized feature is modulated with learned scale γ and bias β generated from the depth feature. Specifically, assume that $\mathcal{F}_I^i \in \mathbb{R}^{C' \times H' \times W'}$ is the extracted feature for image i, where $H' = H/4, W' = W/4$ is feature size, C' is number of feature channel, and note that we omit the mini-batch size for simplicity. $\mathcal{F}_D^i$ is the corresponding depth map feature with the same shape of $\mathcal{F}_I^i$. Then we apply the following normalization ($j \in \{1, \ldots, H'\}, k \in$

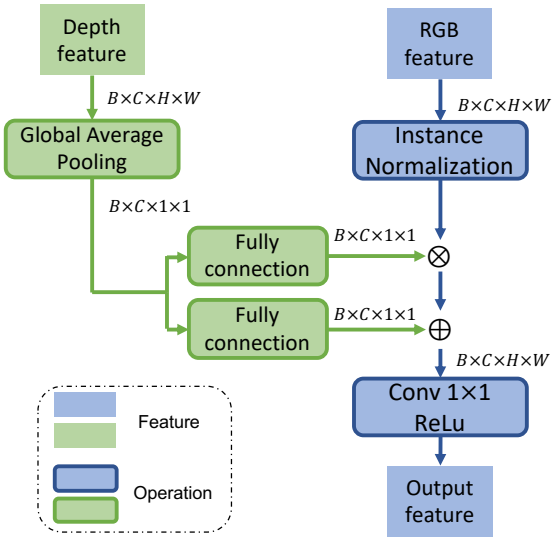

Fig. 3. Adaptive Depth-guided Instance Normalization. The feature from depth map are applied to generate channel-wise affine parameters γ and β for color feature after instance normalization. We first apply a Global Average Pooling to the depth feature, then two independent fully connection layers are utilized to generate two affine parameters for every RGB channel.

$\{1,\ldots,W'\}, c \in \{1,\ldots,C'\}, f^i_{c,j,k} \in \mathcal{F}^i_I,$):

$$\mu^i_c = \frac{1}{H' \times W'} \sum_{j,k} f^i_{c,j,k} \tag{1}$$

$$\sigma^i_c = \sqrt{\frac{1}{H' \times W'} \sum_{j,k} \left(f^i_{c,j,k} - \mu^i_c\right)^2 + \epsilon} \tag{2}$$

$$\mathrm{AdaDIN}(f^i_{c,j,k}, \mathcal{F}^i_D) = \gamma_c(\mathcal{F}^i_D) \left(\frac{f_{c,j,k} - \mu^i_c}{\sigma^i_c}\right) + \beta_c(\mathcal{F}^i_D) \tag{3}$$

Where affine parameters γ, β are generated from depth map feature $\mathcal{F}^i_D$. As shown in figure 3, the depth map feature is fed into two fully connection layers after a Global Average Pooling (GAP), and the outputs of these two fully connections layers are applied to as affine parameters γ and β for color feature, respectively.

3.3 Dynamic depth transformation

Another crucial module of our proposed approach is Dynamic Depth Transformation (DDT). The depth value ($Z-$coordinate in camera coordinate system, in meters) estimation of 3D object is challenging for image-based 3D detectors.

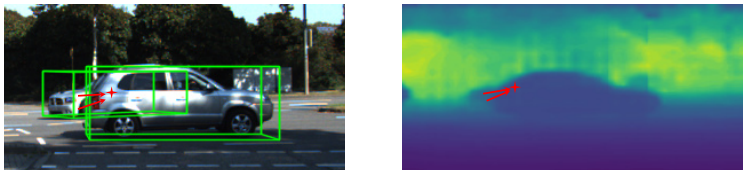

Fig. 4. The intuitive explanation for DDT. When the target object (the car behind) is occluded, we hope to perceive depth value of its center position (red cross) from unoccluded region.

The difficulty lies in the domain gap between 2D RGB context and 3D spatial location. Our AdaDIN module effectively transforms the spatial message from depth map to RGB feature for learning more accurate 3D spatial properties, i.e., 3D object dimension and rotation. Furthermore, to determine the exact depth value of a 3D bounding box, we design a novel Dynamic Depth Transformation module.

To fully utilize the depth map which is the most explicit representation of 3D data to estimate the depth value of a target object, we first learn to generate a depth residual map by a head branch attached to the image features, then we sum this estimated depth residual and the input depth map together. To obtain the depth of 3D bounding box center, we select the depth value in summed map indexed by the corresponding object location. The intuition behind this summation is that, when we have the relatively accurate dense depth map that stands for the depth values of object surface, we only need to an additional "depth residual" from the surface of objects to their 3D bounding box centers. However, in real world scenes, objects like cars are often occluded by another one, and this means that the depth value of the target center point is inaccurately represented by the one who occludes it, which makes it hard to learn an accurate depth residual and would harm the performance of center depth estimation, see figure 4.

To address this problem, we propose a Dynamic Depth Transformation module, which can dynamically sample and transfer the proper depth values for the target object from surrounding positions who are not occluded, see figure 5. Inspired by the [30], we learn a group dynamic offsets conditional on the local input to help a sampling kernel grasp proper depth values.

To start with simple case, for a non-dynamic uniform sampling, we apply a regular grid $\mathcal{R}$ over the dense depth map $D = \{d(p)\}$, where p is 2D index of depth map, for example,

$$\mathcal{R} = \{(-1,-1),(-1,0),\ldots,(0,1),(1,1)\} \tag{4}$$

is a 3×3 sampling kernel with dilation 1. We can get the estimated depth value $\hat{d}(p_0)$ by this uniform kernel at the position p_0:

$$\hat{d}\,(p_0) = \sum_{p_n \in \mathcal{R}} w\,(p_n) \cdot d\,(p_0 + p_n) \tag{5}$$

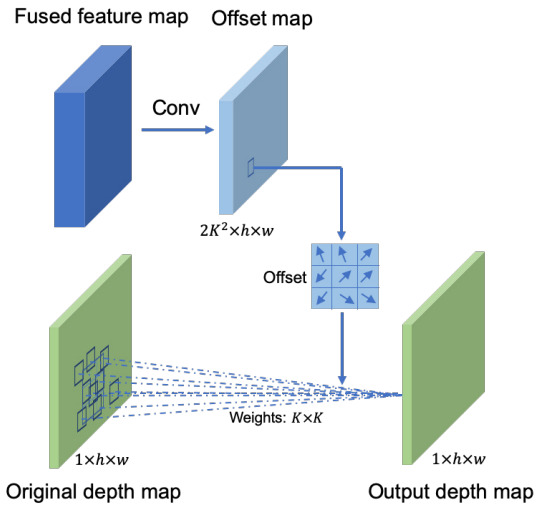

Fig. 5. Dynamic Depth Transformation (DDT). The fused color feature generates offsets for the kernel at every local position and input depth map sample.

where w are learnable weights shared by all position p_0.

We can further augment this regular grid $\mathcal{R}$ with offsets $\{\Delta p_n\}$:

$$\hat{d}\left(p_0\right) = \sum_{p_n \in \mathcal{R}} w\left(p_n\right) \cdot d\left(p_0 + p_n + \Delta p_n\right) \tag{6}$$

Δp_n is dynamically generated conditional on specific sample and local context, which enable network to avoid incorrect depth value in occluded region. The final estimated depth value $d_{obj}(p_0)$ of object 3D bounding box center is:

$$d_{obj}\left(p_0\right) = \hat{d}\left(p_0\right) + d_{res}\left(p_0\right) \tag{7}$$

where d_{res} is estimated depth residual from a 1-channel head. Different from [30], in which the dynamic offsets are obtained by a convolutional layer over the same input feature map, instead, our offsets are estimated over the final RGB features by an attached head. After fusing with depth information by our AdaDIN layers, the final RGB features provide more sufficient local semantic content as well as spatial depth information than raw depth map for obtaining kernel offsets when encountering occlusion.

Note that the offsets of a single location need $K \times K \times 2$ scalars, where K is the size of sample kernel, hence our dynamic offsets head makes a $2K^2$-channel offsets prediction (18 for $K = 3$).

3.4 3D bounding box estimation

To estimate 3D objects bounding boxes, we need the following properties: locations, dimensions and orientation. Locations is determined by the depth and

projected 3D location in image plane; Dimensions is the size of bounding boxes (height, width and length); Orientation is the rotation angle of bounding box around $y-$axis (perpendicular to ground). Therefore, we attach multiple heads to the feature extractors to produce a group of 3D predictions, see figure 2. Specifically, in addition to the object distance $D \in \mathbb{R}^{H' \times W' \times 1}$ and depth transformation offset $O_{depth} \in \mathbb{R}^{H' \times W' \times 18}$, which have been discussed in section 3.3, the complete model produce a objects 2D center heat-map $\mathcal{H} \in \mathbb{R}^{H' \times W' \times C}$, 2D center offset $O_{2D} \in \mathbb{R}^{H' \times W' \times 2}$, 2D-3D offsets $O_{2D-3D} \in \mathbb{R}^{H' \times W' \times 2}$, objects dimensions $S \in \mathbb{R}^{H' \times W' \times 3}$ and objects rotations $R \in \mathbb{R}^{H' \times W' \times 8}$, where $H' = H/4, W' = W/4$ are the size of output feature and C is the number of objects classes. Next we explain these predictions and corresponding loss objective one by one.

Firstly, to determine the position of object i that belongs to class c ($c = 1, \cdots, C$) in the image plane, we regard 2D centers (x_i^c, y_i^c) of objects as key-points and solve these key-points through estimating a C-channel heat-map $\mathcal{H}$, similar to human pose estimation [31–33]. The 2D center offset O_{2D} is predicted to recover the discretization error caused by the output stride of feature extractor networks. It's worth noting that the 2D center and the projected 3D center on the image plane is not necessarily the same point, i.e., there is an offset from object 2D center to its corresponding projected 3D center, therefore, another output head is desired to predict this offset O_{2D-3D}. In particular, the projected 3D center may not lie in the actual image (i.e., for objects which partially missed lying on the edge of image), and our offset O_{2D-3D} could lead to the correct out-of-image 3D center. All classes share the same offsets O_{2D} and O_{2D-3D}. For an object i, our final prediction of projected 3D center on the image plane is obtained by:

$$(\hat{x}_i^c, \hat{y}_i^c) = ({}_{int}\hat{x}_i^c + \hat{o}_i^x + \hat{o'}_i^x, {}_{int}\hat{y}_i^c + \hat{o}_i^y + \hat{o'}_i^y) \tag{8}$$

where $({}_{int}\hat{x}_i^c, {}_{int}\hat{y}_i^c)$ is integer image coordinate generated from heat-map $\mathcal{H}$, $(\hat{o}_i^x, \hat{o}_i^y)$ is the estimated 2D offset obtained from O_{2D} and $(\hat{o'}_i^x, \hat{o'}_i^y)$ is 2D-3D offset obtained from O_{2D-3D}.

Note that the above object 3D center coordinate is represented on the 2D image plane, to lift it to the 3D space, we can simply re-project the center point to the 3D space by its depth Z_i (obtained from our dynamic depth transformation module) and camera intrinsics which is assumed known from the datasets:

$$Z_i \cdot [x_i, y_i, 1]^\top = K \cdot [X_i, Y_i, Z_i, 1]^\top \tag{9}$$

where $K \in \mathbb{R}^{3 \times 4}$ is camera intrinsics assumed to be known as in [33, 20], $[x_i, y_i, 1]^\top$ and $[X_i, Y_i, Z_i, 1]^\top$ are homogeneous coordinates of 3D object center in 2D image plane and 3D space, respectively.

In addition to 3D coordinate of object center, we still need the orientation and object dimension to get the exact 3D bounding boxes. For orientation α, we actually estimate the viewpoint angle, which is more intuitive for humans. Since it is not easy to regress a single orientation value directly, we encode it into 8 scalars lying in 2 bins, with 4 scalars for each bin and then apply a in-bin

regression, like [10]. Our orientation head thus outputs a 8-channel prediction R. For object dimension, we regress the height, width and length (h_i, w_i, l_i) for each object with a 3-channel head prediction S.

Loss objective. Our overall loss is

$$\mathcal{L} = \mathcal{L}_{\mathcal{H}} + \mathcal{L}_{O,2D} + \mathcal{L}_{O,3D-2D} + \mathcal{L}_{depth} + \mathcal{L}_{dim} + \mathcal{L}_{rotation} \tag{10}$$

For simplicity, we omit the weighting factors for each loss term. Where heatmap loss $\mathcal{L}_{\mathcal{H}}$ is a penalty-reduced pixel-wise logistic regression with focal loss, following [33, 34] $\mathcal{L}_{rotation}$ is orientation regression loss. 2D offset loss $\mathcal{L}_{O,2D}$, 3D-2D offset loss $\mathcal{L}_{O,3D-2D}$, object depth loss $\mathcal{L}_{depth}$ and object dimension loss $\mathcal{L}_{dim}$ are defined as L1-loss ($\mathcal{L}_{dim}$ is in meters and $\mathcal{L}_{O,3D-2D}, \mathcal{L}_{O,2D}$ are in pixels):

$$\begin{aligned}
&\mathcal{L}_{O,2D} = \tfrac{1}{N}\textstyle\sum_{i=1}^{N}(o_i^x - \hat{o}_i^x) + (o_i^y - \hat{o}_i^y) \\
&\mathcal{L}_{O,3D-2D} = \tfrac{1}{N}\textstyle\sum_{i=1}^{N}(o'^x_i - \hat{o'^x_i}) + (o'^y_i - \hat{o'^y_i}) \\
&\mathcal{L}_{depth} = \tfrac{1}{N}\textstyle\sum_{i=1}^{N}(d_i - \hat{d}_i) \\
&\mathcal{L}_{dim} = \tfrac{1}{N}\textstyle\sum_{i=1}^{N}(h_i - \hat{h}_i) + (w_i - \hat{w}_i) + (l_i - \hat{l}_i)
\end{aligned} \tag{11}$$

where $(o_i^x, o_i^y), (o'^x_i, o'^y_i)$ is ground truth 2D offset and 3D-2D offset of object i, $\hat{h}_i, \hat{w}_i, \hat{l}_i$ are estimated size of object i and N is the number of objects.

3.5 Feature extraction

In principle, any deep network is suitable for our feature extractor. In our experiments, we adopt Deep Layer Aggregation [35] (DLA-34) architecture as our image and depth feature extractors because DLA balances speed and accuracy well. Original DLA network is designed for image classification task with hierarchical skip connections and we adapt it to our 3D detection framework. Inspired by [33], we adopt a fully convolutional upsampling version of DLA-34 with network stride of 4, and the 3×3 deformable convolution [30] is applied at upsampling layers to replace normal convolution. We have two similar feature extraction networks for RGB image and depth map respectively, and the image and depth map share the same input size $H \times W \times 3$. Note that we tile the original 1-channel depth map three times to form a 3-channel input. The size of output feature is $H/4 \times W/4 \times 64$.

After RGB image and depth map are fed into two DLA-34 feature extractors separately, the feature of depth map is utilized as a guidance for image feature through fusing 3D spatial message from depth map by our Adaptive Depth-guided instance normalization module, which is introduced in section 3.2.

3.6 Implementation details

Our proposed approach is implemented in PyTorch framework and takes about 12 hours to train on 2 NIVIDIA TITAN X GPUs for KITTI. We train our networks for 70 epochs with batch size of 14. For input, we normalize each RGB channel of color image with means and standard deviations calculated over all

Fig. 6. Qualitative results of our 3D detection results on KITTI. Bounding boxes from different class are drawn in different color.

training data. The input depth maps are inferenced by DORN [23], and are tiled into 3-channel images before fed into feature extractor. The input depth are also normalized with depth mean and standard deviation. When training, encoder of our feature extractor is initialized with ImageNet pretrained weights, and the Adam optimizer is applied with $\beta = [0.9, 0.999]$. Learning rate is initialized with 1.25e-4 and decays at epoch 45 and 60 with $0.1\times$. All loss weighting factors are set to 1. When testing, we apply non-maximal suppression (NMS) on center point heat-map with the threshold of 0.2.

Table 1. Bird's eye view and 3D detection results: Average Precision (in %) of bird's eye view boxes and 3D bounding boxes on KITTI *validation* set at IoU ≥ 0.5. In Data column, **M** means only taking monocular image as input; **M+D** means taking both monocular image and generated depth map as input.

Method	Data	AP_{BEV}			AP_{3D}		
		easy	moderate	hard	easy	moderate	hard
Mono3D [9]	M	30.50	22.39	19.16	25.19	18.20	15.52
Deep3DBox [10]	M	30.02	23.77	18.83	27.04	20.55	15.88
Monogrnet[36]	M	54.21	39.69	33.06	50.51	36.97	30.82
Multi-Fusion [11]	M	55.02	36.73	31.27	47.88	29.48	26.44
M3D-RPN [26]	M	55.37	**42.49**	35.29	48.96	**39.57**	33.01
Ours	M	**56.8**	42.3	**35.9**	**51.60**	38.9	**33.7**
Pseudo-LiDAR [19]	M+D	70.8	49.4	42.7	66.30	42.30	38.50
AM3D [20]	M+D	**72.64**	51.82	44.21	68.86	49.19	42.24
Ours	M+D	71.35	**53.54**	**45.74**	**67.01**	**49.77**	**43.09**

4 Experiments

Datasets. We evaluate our approach on the widely used KITTI 3D detection benchmark [24]. The **KITTI** datasets contains 7,481 RGB images sampled from different scenes with corresponding 3D objects annotations and LiDAR data for training and 7,518 for testing. The calibration parameters are also provided for each frame and the objects are labeled into three classes for evaluation: Car, Pedestrian and Cyclist. To compare with previous works, we split out 3,769 images for validation and remaining 3,712 for training our networks, following [21]. Samples from the same sequence are avoided being included in both training and validation set.

Evaluation metric. For KITTI, average precision (AP) calculated from precision-recall curves of two tasks are evaluated in our experiments: Bird's Eye View (BEV) and 3D Object Detection. According to the occlusion/truncation and the size of an object in the 2D image, the evaluation has three difficulty setting of easy, moderate and hard under IoU ≥ 0.5 or 0.7 per class. We show the major results on Car to compare with previous works.

4.1 Results on KITTI

We conduct our experiments on KITTI split [21]. The results on KITTI *validation* set are shown in table 1 and table 2(IoU ≥ 0.5 and IoU ≥ 0.7, respectively). We only list the monocular image-based methods here for fair comparison. For our model without depth map input, we just remove our DDT (Dynamic Depth transformation) module and replace Adaptive Depth-guided Instance Normalization (AdaDIN) layer with normal Instance Normalization. Then our results still outperforms all approaches who take only single image as input under easy and hard difficulty, and we also show a close accuracy with M3D-RPN [26] under moderate difficulty.

For our full model, we achieve state-of-the-art among all methods under moderate and hard difficulty at IoU ≥ 0.5, and also performs closely with AM3D [20]. For results at IoU ≥ 0.7, we can observe that comparing with previous works, our method improves the performance by a large margin from table 2. Some qualitative examples are shown in figure 6. We also report our full model results on KITTI *test* set at IoU ≥ 0.7 in fugure 3, showing superior performance to previous works.

Table 2. Bird's eye view and 3D detection results: Average Precision (in %) of bird's eye view boxes and 3D bounding boxes on KITTI *validation* set at IoU ≥ 0.7

Method	AP_{BEV}			AP_{3D}		
	easy	moderate	hard	easy	moderate	hard
MonoDIS [37]	18.45	12.58	10.66	11.06	7.60	6.37
M3D-RPN [26]	20.85	15.62	11.88	14.53	11.07	8.65
Ki3D [38]	27.83	19.72	15.10	19.76	14.10	10.47
Ours	**34.97**	**26.01**	**21.78**	**23.12**	**17.10**	**14.29**

Table 3. Evaluation results on KITTI *test* set at IoU ≥ 0.7.

Method	AP_{BEV}			AP_{3D}		
	easy	moderate	hard	easy	moderate	hard
FQNet [39]	5.40	3.23	2.46	2.77	1.51	1.01
ROI-10D [27]	9.78	4.91	3.74	4.32	2.02	1.46
GS3D [25]	8.41	6.08	4.94	4.47	2.90	2.47
MonoPSR [40]	18.33	12.58	9.91	10.76	7.25	5.85
Ours	**18.71**	**13.03**	**11.02**	**11.52**	**8.26**	**6.97**

4.2 Ablation study

We conduct our ablation study and experiment analysis on KITTI split [21] on Car class. We adopt moderate setting on Bird's eye view detection and 3D detection task to show our analysis results.

Adaptive depth-guided instance normalization. AdaDIN is designed to adaptively transfer spatial depth information to the color context feature. We compare three versions of our methods to verify its effectiveness: (1). **Base** model. The baseline model of our approach, where the AdaDIN and DDT are removed. (2). **Base+AdaDIN**. Our baseline model with AdaDIN layer, and this model needs generated monocular depth map as input for AdaDIN layer. From table 4, we can observe that our AdaDIN greatly increases the performance of 3D detection performance thanks to the information transferred from depth feature.

Table 4. Comparisons of models with each component. The validation results on KITTI 3D detection results are shown.

Method	easy	moderate	hard
Base	51.6	38.9	33.7
Base+DDT	59.53	44.30	40.82
Base+AdaDIN	62.37	45.08	37.61
Full model	**67.01**	**49.77**	**43.09**

Table 5. Comparison of our dynamic offsets generation strategy and deformable convolution.

Method	easy	moderate	hard
Deformable [30]	33.34	27.58	23.49
Ours	67.01	49.77	43.09

Dynamic depth transformation. Our Dynamic depth transformation (DDT) module is able to address occlusion issue in very common urban scenes. From table 4, we can see that DDT also shows improvements for 3D detection.

Offsets in DDT module. As elaborated in section 3.3, to tackle occlusion problem, we apply a dynamic offset to a uniform sampling kernel for recovering correct object depth. Different to Deformable Convolution [30], our kernel offsets are generated from image feature and then apply to another source input – raw depth map. We compare these two strategies and show the result in table 5.

We can observe from table 5 that our offset generation strategy outperforms Deformable convolution with a large margin. The reason is that our RGB normalized feature affined with parameters generated from depth map feature contains not only high level color context but also 3D depth information. On the other hand, very limited information could be exacted by a few convolution layers from raw depth map. Therefore, more accurate local depth offset could be estimated by our approach.

5 Conclusion

In this paper, we proposed a novel monocular 3D detection approach. One of our key components is Adaptive Depth-guided Instance Normalization, which could effectively fuse 3D spatial information obtained from depth map features with the color context message from RGB features for accurate 3D detection. Another crucial module is Dynamic Depth transformation, which is helpful when the detector encounters occlusions. Extensive experiments show our method achieves state-of-the-art performance on the KITTI 3D detection benchmark among other monocular image-based methods.

References

1. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: CVPR. (2017)
2. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: NeurIPS. (2017)
3. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: CVPR. (2018)
4. Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum pointnets for 3d object detection from rgb-d data. In: CVPR. (2018)
5. Shi, S., Wang, X., Li, H.: Pointrcnn: 3d object proposal generation and detection from point cloud. In: CVPR. (2019)
6. Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3d object detection network for autonomous driving. In: CVPR. (2017)
7. Liang, M., Yang, B., Wang, S., Urtasun, R.: Deep continuous fusion for multi-sensor 3d object detection. In: ECCV. (2018)
8. Yang, B., Luo, W., Urtasun, R.: Pixor: Real-time 3d object detection from point clouds. In: CVPR. (2018)
9. Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., Urtasun, R.: Monocular 3d object detection for autonomous driving. In: CVPR. (2016)
10. Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J.: 3d bounding box estimation using deep learning and geometry. In: CVPR. (2017)
11. Xu, B., Chen, Z.: Multi-level fusion based 3d object detection from monocular images. In: CVPR. (2018)
12. Zhang, L., Li, X., Arnab, A., Yang, K., Tong, Y., Torr, P.H.: Dual graph convolutional network for semantic segmentation. In: BMVC. (2019)
13. Zhang, L., Xu, D., Arnab, A., Torr, P.H.: Dynamic graph message passing networks. In: CVPR. (2020)
14. Hou, Q., Zhang, L., Cheng, M.M., Feng, J.: Strip pooling: Rethinking spatial pooling for scene parsing. In: CVPR. (2020)
15. Li, X., Zhang, L., You, A., Yang, M., Yang, K., Tong, Y.: Global aggregation then local distribution in fully convolutional networks. In: BMVC. (2019)
16. Li, X., Li, X., Zhang, L., Cheng, G., Shi, J., Lin, Z., Tan, S., Tong, Y.: Improving semantic segmentation via decoupled body and edge supervision. In: ECCV. (2020)
17. Wang, Q., Zhang, L., Bertinetto, L., Hu, W., Torr, P.H.: Fast online object tracking and segmentation: A unifying approach. In: CVPR. (2019)
18. Zhu, F., Zhang, L., Fu, Y., Guo, G., Xie, W.: Self-supervised video object segmentation. In: arXiv preprint. (2020)
19. Wang, Y., Chao, W.L., Garg, D., Hariharan, B., Campbell, M., Weinberger, K.: Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In: CVPR. (2019)
20. Ma, X., Wang, Z., Li, H., Zhang, P., Ouyang, W., Fan, X.: Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving. In: ICCV. (2019)
21. Chen, X., Kundu, K., Zhu, Y., Berneshawi, A.G., Ma, H., Fidler, S., Urtasun, R.: 3d object proposals for accurate object class detection. In: NeurIPS. (2015)
22. Li, P., Chen, X., Shen, S.: Stereo r-cnn based 3d object detection for autonomous driving. In: CVPR. (2019)
23. Fu, H., Gong, M., Wang, C., Batmanghelich, K., Tao, D.: Deep Ordinal Regression Network for Monocular Depth Estimation. In: CVPR. (2018)

24. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: CVPR. (2012)
25. Li, B., Ouyang, W., Sheng, L., Zeng, X., Wang, X.: Gs3d: An efficient 3d object detection framework for autonomous driving. In: CVPR. (2019)
26. Brazil, G., Liu, X.: M3d-rpn: Monocular 3d region proposal network for object detection. In: ICCV. (2019)
27. Manhardt, F., Kehl, W., Gaidon, A.: Roi-10d: Monocular lifting of 2d detection to 6d pose and metric shape. In: CVPR. (2019)
28. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: CVPR. (2017)
29. Park, T., Liu, M.Y., Wang, T.C., Zhu, J.Y.: Semantic image synthesis with spatially-adaptive normalization. In: CVPR. (2019)
30. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: CVPR. (2017)
31. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: ECCV. (2016)
32. Wei, S.E., Ramakrishna, V., Kanade, T., Sheikh, Y.: Convolutional pose machines. In: CVPR. (2016)
33. Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. In: arXiv preprint. (2019)
34. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: CVPR. (2017)
35. Yu, F., Wang, D., Shelhamer, E., Darrell, T.: Deep layer aggregation. In: CVPR. (2018)
36. Qin, Z., Wang, J., Lu, Y.: Monogrnet: A geometric reasoning network for monocular 3d object localization. In: AAAI. (2019)
37. Simonelli, A., Bulo, S.R., Porzi, L., López-Antequera, M., Kontschieder, P.: Disentangling monocular 3d object detection. In: CVPR. (2019)
38. Brazil, G., Pons-Moll, G., Liu, X., Schiele, B.: Kinematic 3d object detection in monocular video. In: ECCV. (2020)
39. Liu, L., Lu, J., Xu, C., Tian, Q., Zhou, J.: Deep fitting degree scoring network for monocular 3d object detection. In: CVPR. (2019)
40. Ku, J., Pon, A.D., Waslander, S.L.: Monocular 3d object detection leveraging accurate proposals and shape reconstruction. In: CVPR. (2019)