Task-Aware Monocular Depth Estimation for 3D Object Detection

Xinlong Wang,[1]* Wei Yin,[1] Tao Kong,[2] Yuning Jiang,[2] Lei Li,[2] Chunhua Shen[1]

[1]The University of Adelaide, Australia, [2]Bytedance AI Lab
{xinlong.wang, wei.yin, chunhua.shen}@adelaide.edu.au,
taokongcn@gmail.com, {jiangyuning, lileilab}@bytedance.com

Abstract

Monocular depth estimation enables 3D perception from a single 2D image, thus attracting much research attention for years. Almost all methods treat foreground and background regions ("things and stuff") in an image equally. However, not all pixels are equal. Depth of foreground objects plays a crucial role in 3D object recognition and localization. To date how to boost the depth prediction accuracy of foreground objects is rarely discussed. In this paper, we first analyze the data distributions and interaction of foreground and background, then propose the foreground-background separated monocular depth estimation (ForeSeE) method, to estimate the foreground and background depth using separate optimization objectives and decoders. Our method significantly improves the depth estimation performance on foreground objects. Applying ForeSeE to 3D object detection, we achieve 7.5 AP gains and set new state-of-the-art results among other monocular methods. Code will be available at: https://github.com/WXinlong/ForeSeE.

1 Introduction

Depth bridges the gap between 2D and 3D perception in computer vision. A precise depth map of an image provides rich 3D geometry information like locations and shapes for objects and stuff in a scene, thus attracting more and more attention in both 2D and 3D understanding fields. Monocular depth estimation, which aims to predict the depth map from a single image, is an ill-posed problem, as infinite number of 3D scenes can be projected to the same 2D image. With the development of deep convolutional neural networks (Krizhevsky, Sutskever, and Hinton 2012; Simonyan and Zisserman 2014; He et al. 2016), recent works have made great progress (Xu et al. 2018; Fu et al. 2018; Li et al. 2018). They typically consist of an encoder for feature extraction and a decoder for generating the depth of the whole scene, either by regressing the depth values or predicting the depth range categories. Plausible results have been shown.

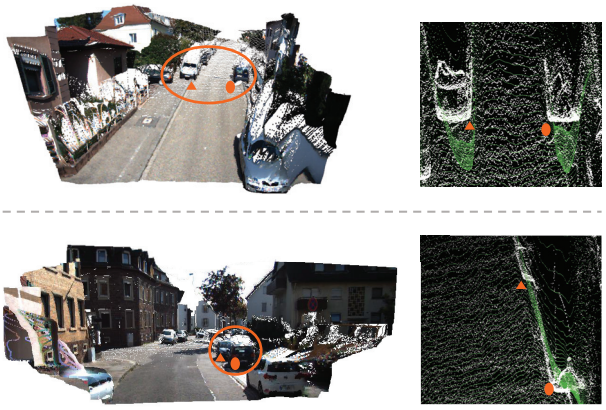


Figure 1: Examples of low precision prediction of foreground depth. For each row, the left picture is the projected point cloud transformed from ground truth depth map and RGB image; the right picture is the bird's-eye-view close-up to compare the depth (in green) predicted by the baseline depth estimation method with the ground truth (in white). The inaccurate object location and shape pose challenges for 3D recognition, localization and orientation estimation.

When the monocular methods are applied to other tasks focusing on foreground object analysis, *e.g.*, 3D object detection, there are two main obstacles from the low precision of foreground depth: (1) Poor estimate of the object center location; (2) Distorted or faint object shapes. We show some examples in Figure 1. The inaccurate object location and shape make the downstream localization and recognition challenging. The above issues could be handled by enhancing the depth estimation performance on foreground regions. However, all these state-of-the-art methods treat foreground depth and background depth equally, which leads to sub-optimal performance on foreground objects.

In fact, foreground depth and background depth show different data distributions. We make qualitative and quantitative comparisons in Figure 1, Figure 2 and Table 1. Foreground pixels tend to gather into clusters, bring more and bigger depth change and look like frustums in 3D space rather than flat surfaces like road and buildings. Second,

*This work is done when Xinlong Wang is an intern at Bytedance AI Lab.

foreground pixels account for only a small part of the whole scene. For instance, in the KITTI-Object dataset (Geiger et al. 2013), 90.6% pixels belong to background, while only 9.4% pixels fall within foreground. Furthermore, not all pixels are equal. As just described, foreground pixels play a more crucial role in downstream applications, *e.g.*, autonomous driving and robotic grasping. For example, an estimation error on a car is much different from the same error on a building. The inaccurate shape and location of the car could be catastrophic for 3D object detectors.

The observations make one wonder how to boost the estimation accuracy of foreground while do no harm to background. First of all, it is neither a hard example mining problem, nor a self-learned local attention problem. Different from the former one, here we want to further enhance the performance on specific regions in the scene, which does not have to be harder example. As we can see in Figure 3, foreground is indeed not harder than background. Attention mechanism is widely used to focus on more discriminative local regions in semantic classification problems, *e.g.*, semantic segmentation and fine-grained classification. But this is not the case for depth estimation. Given a close-up of a car in a scene, one could classify the semantic categories, but could not tell the depth. Another choice is to separately train the foreground and background regions, since the data distributions are different. However, we show that the foreground and background are interdependent to each other for inferring the depth and boosting the performance.

Instead, we formulate it as a multi-objective optimization problem. The objective functions of foreground and background depth are separated. So do the depth decoders. Thus, the foreground depth decoder could fit the foreground depth as well as possible while do no harm to background.

To summarize, our contributions are as follows:

- We conduct pioneering discussion about difference and interaction of foreground and background in monocular depth estimation. We show that the different patterns of foreground and background depth lead to sub-optimal results on foreground pixels.
- We propose ForeSeE, to learn and predict foreground and background depth separately. Specifically, it contains separate depth decoders for foreground and background regions, an objective sensitive loss funcion to optimize corresponding decoders, and a simple yet effective foreground-background merging strategy.
- With the proposed ForeSeE, we are able to predict much superior foreground depth, whereas background depth is not affected. Furthermore, utilizing the predicted depth maps, our model achieves 7.5 AP gains on 3D object detection task, which effectively verifies our motivation.

2 Related Work

Monocular Depth Estimation. Monocular depth estimation (MDE) is a long-lasting problem in computer vision and robotics. Early works (Saxena, Sun, and Ng 2009; Saxena, Chung, and Ng 2005) mainly leverage non-parametric optimization to predict the depth from handcrafted features (Hoiem, Efros, and Hebert 2007; Ladicky, Shi, and Pollefeys 2014). Recent powerful deep convolutional neural networks (DCNN) boost the performance of MDE significantly. Most methods formulate MDE as a pixel-wise supervised learning problem. Eigen *et al*. (Eigen, Puhrsch, and Fergus 2014) is the first to utilize the multi-scale DCNN to regress the depth map from a single image. Then, various innovative network architectures (Liu et al. 2016; Li, Klein, and Yao 2017; Xu et al. 2019; Fu et al. 2018) are proposed to leverage strong high-level features. Furthermore, several methods (Zhao et al. 2019; Qi et al. 2018b; Wei et al. 2019) propose to explicitly enforce geometric constraints for the optimizing process. In this work, we focus on boosting the depth prediction accuracy of foreground objects with the proposed ForeSeE optimization strategy.

Not All Pixels are Equal. Some prior works noticed that it is sub-optimal to treat all pixels equally in dense prediction tasks. Sevilla *et al*. (Sevilla-Lara et al. 2016) tackle optic flow estimation by defining different models of image motion for different regions. Li *et al*. (Li et al. 2017) use deep layer cascade to first segment the easy pixels then the harder ones. Sun *et al*. (Sun et al. 2019) select and weight synthetic pixels which are similar with real ones for learning semantic segmentation. Yuan *et al*. (Yuan et al. 2019) introduce an instance-level adversarial loss for video frame interpolation problem. Shen *et al*. (Shen et al. 2019) propose an instance-aware image-to-image translation framework. However, different from the above works, we focus on depth estimation problem and aim at improving the accuracy of 3D object detection.

Monocular 3D Object Detection. The lack of depth information poses a substantial challenge for estimating 3D bounding boxes from a single image. Many works seek help from geometry priors and estimated depth information. Deep3DBox (Mousavian et al. 2017) proposes to generate 3D proposals based on 2D-3D bounding box consistency constraint. ROI-10D (Manhardt, Kehl, and Gaidon 2019) introduces RoI lifting to extract fused feature maps from input image and estimated depth map, before the 3D bounding box regression. MonoGRNet (Qin, Wang, and Lu 2019a) estimates the depth of the targeting 3D bounding box's center to aid the 3D localization. Recently, some works (Xu and Chen 2018; Wang et al. 2019; Weng and Kitani 2019) propose to convert estimated depth map to lidar-like point cloud to help localize 3D objects. Wang *et al*. (Wang et al. 2019) directly applies 3D object detection methods on the generated pseudo-lidar, and claim 3D point cloud is a much superior representation than 2D depth map for better utilizing depth information. In these methods, a reliable depth map, especially the precise foreground depth, is the key to a successful 3D object detection framework. We perform 3D object detection using the pseudo-LiDAR generated by our depth estimation model. The proposed method largely improves the performance and outperforms state-of-the-art methods.

3 Observation and Analysis

3.1 Preliminaries

Dataset. KITTI dataset (Geiger et al. 2013) has witnessed inspiring progress in the field of depth estimation. As most

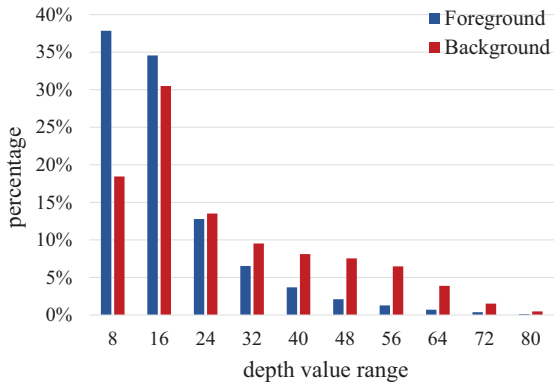


Figure 2: Comparison of depth value distribution between foreground pixels and background pixels. Percentage of pixels with depth value within [x-8, x] meters is reported.

	I	II	III
Foreground	96.77	1.99	1.24
Background	98.63	0.94	0.43

Table 1: Comparison of depth gradient distribution between foreground and background pixels. The gradients are uniformly discretized into three bins: I, II and III, from small to large. Percentage of pixels at each level is reported.

of scenes in KITTI-Raw data have limited foreground objects, we construct a new benchmark which is based on KITTI-Object dataset. We collect the corresponding ground-truth depth map for each image in KITTI-Object training set, and term it as KITTI-Object-Depth (KOD) dataset. A total of $7,481$ image-depth pairs are divided into training and testing subsets with $3,712$ and $3,769$ samples respectively (Chen et al. 2015), which makes sure that images in the two subsets belong to different video clips. 2D bounding boxes are used to distinguish the foreground and background pixels. Pixels fall within the foreground bounding boxes are designated as foreground pixels, while the other pixels are assigned to be background.

Baseline Method. We adopt the same DCNN-based baseline method (Wei et al. 2019) which has already shown state-of-the-art performance on several benchmarks. The main structure falls into the typical encoder-decoder style. Given an input image, the encoder extracts the dense features, then the decoder fetches the features and predicts the quantized depth range categories. Specifically, the depth values are discretized into 100 discrete bins in the log space. The quantized labels are assigned to each of the pixels as their classification labels.

3.2 Analysis on Data Distribution

Few works (Jiao et al. 2018) have analysed the depth distribution, not to mention the foreground and background depth distributions. Here we investigate two kinds of data distribution of foreground and background pixels in training subset. Figure 2 shows the depth value distributions. As shown, more than 75% foreground pixels have depth less than $16m$,

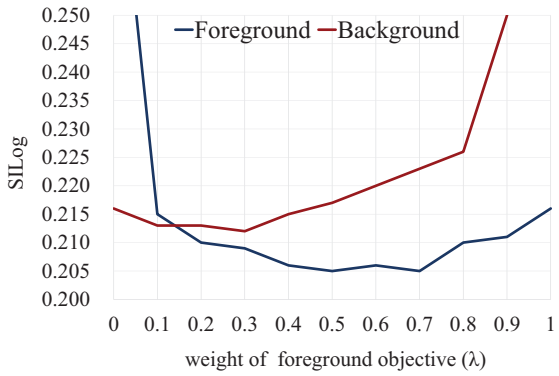


Figure 3: Interaction of foreground and background samples. The depth estimation results (SILog) on foreground and background regions are reported (lower is better). The weight of foreground objective is on x-axis.

while it is about 50% for background. The foreground depth also shows a heavier long-tail distribution. Depth gradient distributions are shown in Table 1. We use the Laplacian of the depth images as the depth gradient, which calculates the second order spacial derivatives. The Laplacian image highlights areas of rapid depth change. The outputs are scaled to [0, 255] and uniformly discretized into three bins: I, II and III, from small to large. In this way, all pixels are divided into three levels according to their gradient values. The foreground has much higher proportion than background at level II and III. Besides the depth range and depth gradient, the difference of shapes should also be noted. Generally, depth provides two kinds of information: location and shape. The foreground objects share similar shapes and look like frustums in 3D space, as shown in Figure 1. Based on the above analysis, we propose to consider the foreground and background separately when estimate their depth.

3.3 Separate Objectives

In dense prediction tasks, generally the loss function can be formulated as:

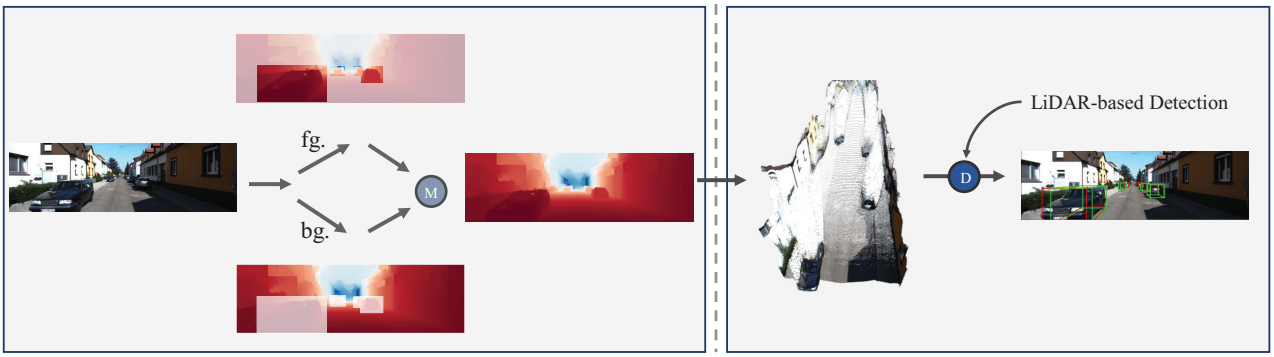
$$L = \frac{1}{N}\sum_{i}^{N} E(y_i, \hat{y}_i), \tag{1}$$

where N is the number of pixels, y_i and $\hat{y}_i$ are the prediction and ground truth of i^{th} pixel. E is the error function, *e.g.*, the widely used cross-entropy error function.

After the analysis in Section 3.2, we further investigate the interaction of foreground and background by splitting the optimization objective. The modified loss function is defined as:

$$L = \lambda \times \frac{1}{N_f}\sum_{i}^{N_f} E(y_i, \hat{y}_i) + (1-\lambda) \times \frac{1}{N_b}\sum_{i}^{N_b} E(y_i, \hat{y}_i), \tag{2}$$

where N_f is the number of foreground pixels, N_b is the number of background pixels and λ acts as the weight to balance the two loss terms. Figure 3 shows our results with different settings of λ. The results are generated by a CNN that



(a) ForeSeE (b) 3D Object Detection

Figure 4: Illustration of the overall pipeline. (a) Foreground-background separated depth estimation. (b) 3D object detection.

has a single depth prediction decoder but the separated objective function. When λ is set to 0, which means only the background samples are used to supervise the training, the result on foreground becomes very poor. Similarly, the performance on background drops sharply when λ is set to 1.0. It verifies that the foreground depth and background depth are distributed differently. When we increase the foreground weight λ from 0 to 0.1, the result on background improves, which indicates that the foreground and background to some extend could help each other. Further, it should be noted that the optimal λ values for foreground and background are different. For instance, the model shows its best performance on foreground when $\lambda = 0.7$, but meanwhile the result on background is much poorer. It indicates that the optimization objectives for foreground and background are not consistent. To address these issues, in Section 4, the foreground-background separated depth estimation method is proposed to achieve the optimum points at the same time.

3.4 Analysis Summary

We highlight three observations:

- The foreground and background depth have different depth value distributions, depth gradient distributions and shape patterns;
- The foreground and background depth reinforce each other due to their shared similarities;
- The optimization objectives of foreground and background depth estimation are mismatched.

4 ForeSeE

In this section we first introduce the network architecture of our method, then present the proposed loss function, and finally show how the mask used to distinguish foreground and background could be dropped during the inference. The whole pipeline is illustrated in Figure 4.

4.1 Separate Depth Decoders

We construct an additional decoder based on the baseline method (Wei et al. 2019), thus there are two parallel decoders which have the same structure. One of the decoders is for foreground depth prediction, while the other one aims to estimate the background depth. Specifically, for an image of size $H \times W \times 3$, each decoder outputs a tensor of size $H \times W \times C$, where C is the number of depth range categories.

Foreground regions are cropped from the output of foreground depth decoder. The background depth range predictions are obtained in the same way. The global depth range predictions are generated by a seamless merge of foreground and background regions. Then the depth range predictions are converted to the final depth map using the soft-weighted-sum strategy (Li, Dai, and He 2018).

4.2 Foreground-background Sensitive Loss Function

As observed in Section 3, although the foreground depth and background depth show different patterns, they do share some similarities and could reinforce each other under an appropriate ratio. Thus, we further weight the foreground and background samples. For either foreground or background branch, the loss function is a weighted average of foreground samples and background samples, but with different bias. Here we define the loss function which supervises the foreground branch as:

$$L_{fg} = \lambda_f \times E_{fg} + (1 - \lambda_f) \times E_{bg}, \tag{3}$$

where E_{fg} represents mean errors calculated on foreground predictions; E_{bg} is the mean error of foreground predictions; λ_f is the weight to balance the foreground and background samples during the training of foreground branch. Larger λ_f leads to more preference for foreground samples. Similarly, the loss function of background branch is formulated as:

$$L_{bg} = \lambda_b \times {E'}_{bg} + (1 - \lambda_b) \times {E'}_{fg}, \tag{4}$$

where λ_b is the weight; ${E'}_{bg}$ and ${E'}_{fg}$ are the mean errors of background predictions and foreground predictions on this background branch.

4.3 Inference without Mask

Here we propose a mask-free merge method such that the binary mask is no longer needed once the training is finished. A max-pooling operation is applied on the foreground

Method	FSL	SD	SO	Foreground		Background		Global	
				absRel	SILog	absRel	SILog	absRel	SILog
ForeSeE	✓	✓	✓	**0.118**	**0.205**	**0.141**	0.210	**0.138**	0.210
		✓	✓	0.120	0.208	**0.141**	**0.209**	0.139	**0.209**
			✓	0.120	**0.205**	0.147	0.217	0.144	0.216
Baseline				0.129	0.216	0.143	0.210	0.141	0.211

Table 2: Ablation study of depth estimation on the KOD dataset. FSL refers to foreground-background sensitive loss; SD refers to separate decoders; SO means separate objectives.

and background outputs before the softmax operation, which represent the confidence scores of being each range category. For each range category of each pixel, the highest confidence score between foreground and background output is retained, to serve as the final prediction. Formally, for the $H \times W \times C$ shaped outputs $P = p_1, p_2, ..., p_{H \times W}$ ($p_i \subseteq \mathbb{R}^C$), The final predictions are calculated as:

$$p'_i = \mathrm{Max}(p_i^f, p_i^b), \tag{5}$$

where p_i^f, p_i^b represent the i^{th} output of foreground and background branch, and $\mathrm{Max}(\cdot)$ is an element-wise maximum operator which takes two vectors as input and outputs a new vector. Then the $H \times W \times C$ shaped output is fed to $softmax$ and soft-weighted-sum operations to produce the final depth map. The results only drop slightly compared with the mask-based merge method (from 0.117 to 0.118 absRel).

5 Experiments

5.1 Experiment Settings

Datasets. We carry out experiments on KITTI dataset, which contains large-scale road scenes captured on driving cars, and serves as a popular benchmark for many computer vision problems related to self-driving cars. Specifically, we construct the KITTI-Object-Depth (KOD) dataset for evaluating the foreground depth estimation, as described in Section 3.1. The KOD dataset will be public available for convenience of future researches. Besides, we also apply our method on KITTI-Object dataset to perform monocular 3D object detection.

Evaluation Metrics. For evaluation of depth estimation, we follow common practice (Li et al. 2018; Wei et al. 2019) and use the mean absolute relative error (absRel) and scale invariant logarithmic error (SILog) as the main metrics. We also report mean relative squared error (sqRel), mean $\log_{10}$ error ($\log_{10}$) and accuracy under threshold (δ_i). As for 3D object detection, we follow the prior works (Liu et al. 2019; Qin, Wang, and Lu 2019b) and focus on the "car" class. We report the results of 3D and bird's-eye-view (BEV) object detection on the validation set. The commonly used average precision (AP) with the IoU thresholds at 0.7 is calculated. The results on KITTI easy, moderate and hard difficulty levels are reported.

Implementation Details. For depth estimation, we follow the most of settings in baseline method (Wei et al. 2019). The ImageNet pretrained ResNeXt-101 (Xie et al. 2017) is used as the backbone model. We train the network for 20 epochs, with batch size 4 and base learning rate set to 0.001. The Stochastic Gradient Descent (SGD) solver is adopted to optimize the network on a single GPU. λ_f and λ_b in foreground-background sensitive loss function are set to 0.2. Given a predicted depth map, the point cloud can be reconstructed based on the pinhole camera model. We transform each pixel (u_i, v_i) with depth value d_i to a 3D point (x_i, y_i, z_i) in left camera coordinate as follows:

$$z_i = d_i, \tag{6}$$

$$x_i = \frac{d_i \times (u_i - c_U)}{f_U}, \tag{7}$$

$$y_i = \frac{d_i \times (v_i - c_V)}{f_V}, \tag{8}$$

where f_U and f_V are the focal length along the x and y coordinate axis; c_U and c_V are the 2D coordinate of the optical center. Following (Wang et al. 2019), we set the reflectance to 1 for each point and remove the points higher than 1 m above the LiDAR source. The resulting point cloud is termed as pseudo-LiDAR. Afterwards, any existing LiDAR-based 3D object detection methods can be applied.

5.2 Depth Estimation

Quantitative Results. We show the ablative results in Table 2. Our ForeSeE outperforms the baseline over all metrics evaluated on foreground, background and global levels. Specifically, when evaluate on foreground level, our method improves the baseline performance by up to 8.5% (from 0.129 to 0.118 absRel). It is in accordance with our intention that ForeSeE is specifically designed to enhance the ability of estimating the foreground depth. We further analyse the effect of each component. When equipped with the separate objectives (SO) described in Section 3.3, the baseline achieves better results on foreground while performs worse when evaluate on background pixels. Directly using the separate decoders (SD) could avoid the harm on background. Finally, the performance on foreground is further improved by applying the foreground-background sensitive loss (FSL).

To compare with other state-of-the-art methods, we apply DenseDepth (Alhashim and Wonka 2018) to the KOD benchmark, which reports the best performance on KITTI (Geiger et al. 2013) and NYUv2 (Silberman et al. 2012) datasets among the methods with public available training code. We obtain results of DenseDepth using the code at github[1], published by the authors. Except

[1]https://github.com/ialhashim/DenseDepth

	Method	absRel ↓	sqRel ↓	SILog ↓	log10 ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑
Foreground	DenseDepth	0.135	0.214	**0.204**	0.057	0.830	0.951	**0.984**
	ForeSeE	**0.118**	**0.193**	0.205	**0.053**	**0.851**	**0.952**	0.982
Global	DenseDepth	**0.138**	**0.208**	**0.209**	0.062	0.782	0.946	**0.987**
	ForeSeE	**0.138**	0.213	0.210	**0.061**	**0.793**	**0.949**	**0.987**

Table 3: Depth estimation performance comapred with DenseDepth (Alhashim and Wonka 2018).

the dataset used, all the settings and hyper-parameters are not modified. The results are shown in Table 3. Compared with DenseDepth, our ForeSeE shows significant advantage on foreground level. For instance, ForeSeE outperforms DenseDepth by absolute 2.7% absRel (from 0.135 to 0.118 absRel), which is a relative improvement of 12.6%.

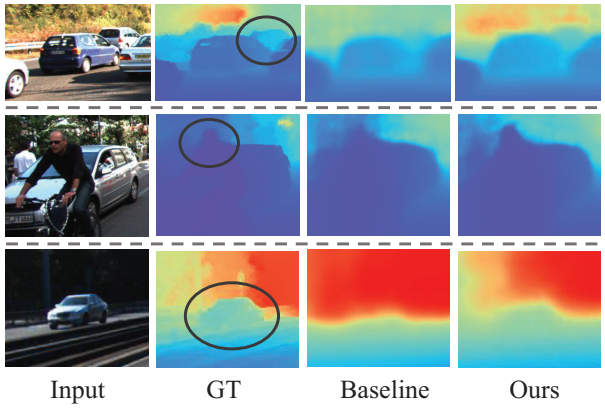


Figure 5: Quantitative comparison of the baseline and our ForeSeE on estimated depth maps.

Figure 6: Quantitative comparison of the baseline and our ForeSeE on converted pseudo-LiDAR signals. Signals in blue are converted from ground truth depth; Baseline pseudo-LiDAR are in red; Our ForeSeE pseudo-LiDAR are in yellow.

Qualitative Results. Besides the quantitative comparison, we show some visualization results. The predicted depth maps are visualized in Figure 5. As shown, our ForeSeE estimates more precise depth on foreground regions. The contour of foreground objects is more clear and accurate. Further, in Figure 6 we compare the estimated depth in the format of 3D point cloud. 3D point cloud is a more intuitive and reasonable representation for visually comparing and debugging depth maps. As shown in Figure 6, our method shows less estimation errors and more accurate bird's-eye-view (BEV) shapes.

5.3 3D Object Detection

To further validate the effectiveness, we conduct experiments on 3D object detection problem. We convert the estimated image-based depth map to LiDAR-like point cloud (pseudo-LiDAR). Then the LiDAR-based algorithms can be applied to recognizing and localizing 3D objects. Here we adopt Frustum-PointNet (F-PointNet) (Qi et al. 2018a) and AVOD (Ku et al. 2018), specifically the F-PointNet-v1 and AVOD-FPN, which are top-performing 3D object detection methods and both utilize the information from LiDAR and RGB images.

Brief Introduction of Detection Methods. Frustum-PointNet leverages 2D detector to generate 2D object region proposals in a RGB image. Each 2D region corresponds to a 3D frustum in 3D space. PointNet-based networks are used to estimate a 3D bounding box from the points within the frustum. AVOD uses multimodal feature fusion RPN which aggregates the front-view image features and BEV LiDAR features to generate 3D object proposals. Based on the proposals, the bounding box regression and category classification are performed in the second subnetwork. We apply the F-PointNet and AVOD on the pseudo-LiDAR generated by our depth estimation model during the training and inference. Hyper-parameters are not modified. More details about the 3D object detector can be referred to the original papers.

Comparisons with State-of-the-art Methods. It should be noted that some works (Wang et al. 2019; Weng and Kitani 2019) use the DORN (Fu et al. 2018) pre-trained on KITTI-raw dataset for depth estimation, which includes the images in training and validation subsets of KITTI-Object. Wang *et al*. (Wang et al. 2019) claim that their results serve as the upper bound. If we also pre-train our baseline depth estimation model on KITTI-raw and use it to generate pseudo-LiDAR, we achieve 20.1 AP which outperforms their reported 18.5 AP when both use the F-PointNet as detector and evaluate on the moderate level of car class. But, we want to clearly set a baseline and fairly compare to other state-of-the-art monocular 3D detection methods.

We compare our method with other methods in Table 4.

(a) Input Image (b) Baseline-PL (c) ForeSeE-PL

Figure 7: Qualitative results of 3D object detection. The ground truth 3D bounding boxes are in red; the predictions are in green.

The methods are evaluated by the average precision (AP) with IoU threshold at 0.7. All the methods are tested on ground-truth 3D bounding boxes. The compared 3D object detection methods include Mono3D (Chen et al. 2016), MLF-MONO (Xu and Chen 2018), ROI-10D (Manhardt, Kehl, and Gaidon 2019), MonoGRNet (Qin, Wang, and Lu 2019a), MonoPSR (Ku, Pon, and Waslander 2019), TLNet-Mono (Qin, Wang, and Lu 2019b) and DFDSNet (Liu et al. 2019). Our depth estimation models are trained on KOD training subset which does not contain validation subset of KITTI-Object. Either with F-PointNet or AVOD as the detection method, our ForeSeE-PL brings remarkable improvements on the basis of baseline-PL over all the metrics, *e.g.*, from 19.0 to 23.4 $\mathrm{AP}_{\mathrm{BEV}}$ with AVOD detector. Note that the 3D detection average precision ($\mathrm{AP}_{\mathrm{3D}}$) is the most widely used metric, on which our method achieves 7.5 AP gains (from 7.5 to 15.0 AP) and outperforms all the state-of-the-art methods. Another advantage of our method is that it is not limited to specific 3D object detection methods. With stronger 3D object detector, we achieve larger improvements, *e.g.*, 3.6 AP gains on F-PointNet and 7.5 AP gains on AVOD when evaluate on easy level of $\mathrm{AP}_{\mathrm{3D}}$.

Qualitative Results. The visualization of detection results are shown in Figure 7. The 3D bounding boxes are projected into image space for better visualization. There are two obvious advantages of using ForeSeE-PL: less missed detection and more accurate localization. Inaccurate depth predictions will result in shifted localization or rotated orientation, as in Figure 7(b). Even worse, the objects can not be detected if the depth estimation model treats foreground objects as background region, thus causing more missed detections. Our ForeSeE method largely alleviates the problems through predicting more accurate depth on foreground regions.

6 Conclusion

In this paper, we first analyse the data distribution of foreground and background depth and explicitly explore the interactions. Based on the observations, a simple and effective depth estimation pipeline, namely ForeSeE, is proposed to estimate foreground depth and background depth separately. We introduce a foreground depth estimation benchmark and set fair baselines to encourage the future studies. The experiments on monocular depth estimation and 3D object detection problems demonstrate the effectiveness of ForeSeE. We expect wide application of the proposed method in depth estimation and related downstream problems, *e.g.*, 3D object recognition and localization.

Method	Easy	Moderate	Hard
Mono3D	5.2 / 2.5	5.2 / 2.3	4.1 / 2.3
MLF-MONO	22.0 / 10.5	13.6 / 5.7	11.6 / 5.4
ROI-10D	14.5 / 9.6	9.9 / 6.6	8.7 / 6.3
MonoGRNet	- / 13.9	- / 10.2	- / 7.6
MonoPSR	20.6 / 12.8	**18.7** / 11.5	14.5 / 8.6
TLNet-Mono	21.9 / 13.8	15.7 / 9.7	14.3 / 9.3
DFDSNet	9.5 / 6.0	8.0 / 5.5	7.7 / 4.8
F-PN (baseline-PL)	17.3 / 9.6	11.8 / 5.4	10.4 / 5.0
F-PN (Our ForeSeE-PL)	20.2 / 13.2	12.6 / 9.4	12.0 / 8.2
AVOD (baseline-PL)	19.0 / 7.5	15.3 / 6.1	13.0 / 5.4
AVOD (Our ForeSeE-PL)	**23.4** / **15.0**	17.4 / **12.5**	**15.9** / **12.0**

Table 4: Monocular 3D object detection results on KITTI benchmark. We report $\mathrm{AP}_{\mathrm{BEV}}/\mathrm{AP}_{\mathrm{3D}}$ (in %) of the car category. F-PN refers to Frustum-PointNet. PL refers to pseudo-LiDAR. Here ForeSeE-PL stands for using pseudo LiDAR from ForeSeE.

References

Alhashim, I., and Wonka, P. 2018. High quality monocular depth estimation via transfer learning. *arXiv:1812.11941*.

Chen, X.; Kundu, K.; Zhu, Y.; Berneshawi, A. G.; Ma, H.; Fidler, S.; and Urtasun, R. 2015. 3d object proposals for accurate object class detection. In *NIPS*.

Chen, X.; Kundu, K.; Zhang, Z.; Ma, H.; Fidler, S.; and

Urtasun, R. 2016. Monocular 3d object detection for autonomous driving. In *CVPR*.

Eigen, D.; Puhrsch, C.; and Fergus, R. 2014. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*.

Fu, H.; Gong, M.; Wang, C.; Batmanghelich, K.; and Tao, D. 2018. Deep ordinal regression network for monocular depth estimation. In *CVPR*.

Geiger, A.; Lenz, P.; Stiller, C.; and Urtasun, R. 2013. Vision meets robotics: The KITTI dataset. *IJRR*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

Hoiem, D.; Efros, A. A.; and Hebert, M. 2007. Recovering surface layout from an image. *IJCV*.

Jiao, J.; Cao, Y.; Song, Y.; and Lau, R. 2018. Look deeper into depth: Monocular depth estimation with semantic booster and attention-driven loss. In *ECCV*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

Ku, J.; Mozifian, M.; Lee, J.; Harakeh, A.; and Waslander, S. L. 2018. Joint 3d proposal generation and object detection from view aggregation. In *IROS*.

Ku, J.; Pon, A. D.; and Waslander, S. L. 2019. Monocular 3d object detection leveraging accurate proposals and shape reconstruction. In *CVPR*.

Ladicky, L.; Shi, J.; and Pollefeys, M. 2014. Pulling things out of perspective. In *CVPR*.

Li, X.; Liu, Z.; Luo, P.; Loy, C. C.; and Tang, X. 2017. Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade. In *CVPR*.

Li, R.; Xian, K.; Shen, C.; Cao, Z.; Lu, H.; and Hang, L. 2018. Deep attention-based classification network for robust depth prediction. In *ACCV*.

Li, B.; Dai, Y.; and He, M. 2018. Monocular depth estimation with hierarchical fusion of dilated cnns and soft-weighted-sum inference. *Pattern Recognition*.

Li, J.; Klein, R.; and Yao, A. 2017. A two-streamed network for estimating fine-scaled depth maps from single RGB images. In *ICCV*.

Liu, F.; Shen, C.; Lin, G.; and Reid, I. D. 2016. Learning depth from single monocular images using deep convolutional neural fields. *TPAMI*.

Liu, L.; Lu, J.; Xu, C.; Tian, Q.; and Zhou, J. 2019. Deep fitting degree scoring network for monocular 3d object detection. In *CVPR*.

Manhardt, F.; Kehl, W.; and Gaidon, A. 2019. Roi-10d: Monocular lifting of 2d detection to 6d pose and metric shape. In *CVPR*.

Mousavian, A.; Anguelov, D.; Flynn, J.; and Kosecka, J. 2017. 3d bounding box estimation using deep learning and geometry. In *CVPR*.

Qi, C. R.; Liu, W.; Wu, C.; Su, H.; and Guibas, L. J. 2018a. Frustum pointnets for 3d object detection from RGB-D data. In *CVPR*.

Qi, X.; Liao, R.; Liu, Z.; Urtasun, R.; and Jia, J. 2018b. Geonet: Geometric neural network for joint depth and surface normal estimation. In *CVPR*.

Qin, Z.; Wang, J.; and Lu, Y. 2019a. Monogrnet: A geometric reasoning network for monocular 3d object localization. In *AAAI*.

Qin, Z.; Wang, J.; and Lu, Y. 2019b. Triangulation learning network: From monocular to stereo 3d object detection. In *CVPR*.

Saxena, A.; Chung, S. H.; and Ng, A. Y. 2005. Learning depth from single monocular images. In *NIPS*.

Saxena, A.; Sun, M.; and Ng, A. Y. 2009. Make3d: Learning 3d scene structure from a single still image. *TPAMI*.

Sevilla-Lara, L.; Sun, D.; Jampani, V.; and Black, M. J. 2016. Optical flow with semantic segmentation and localized layers. In *CVPR*.

Shen, Z.; Huang, M.; Shi, J.; Xue, X.; and Huang, T. S. 2019. Towards instance-level image-to-image translation. In *CVPR*.

Silberman, N.; Hoiem, D.; Kohli, P.; and Fergus, R. 2012. Indoor segmentation and support inference from RGBD images. In *ECCV*.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*.

Sun, R.; Zhu, X.; Wu, C.; Huang, C.; Shi, J.; and Ma, L. 2019. Not all areas are equal: Transfer learning for semantic segmentation via hierarchical region selection. In *CVPR*.

Wang, Y.; Chao, W.-L.; Garg, D.; Hariharan, B.; Campbell, M.; and Weinberger, K. Q. 2019. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *CVPR*.

Wei, Y.; Liu, Y.; Shen, C.; and Yan, Y. 2019. Enforcing geometric constraints of virtual normal for depth prediction. In *ICCV*.

Weng, X., and Kitani, K. 2019. Monocular 3d object detection with pseudo-lidar point cloud. *arXiv:1903.09847*.

Xie, S.; Girshick, R. B.; Dollár, P.; Tu, Z.; and He, K. 2017. Aggregated residual transformations for deep neural networks. In *CVPR*.

Xu, B., and Chen, Z. 2018. Multi-level fusion based 3d object detection from monocular images. In *CVPR*.

Xu, D.; Wang, W.; Tang, H.; Liu, H.; Sebe, N.; and Ricci, E. 2018. Structured attention guided convolutional neural fields for monocular depth estimation. In *CVPR*.

Xu, D.; Ricci, E.; Ouyang, W.; Wang, X.; and Sebe, N. 2019. Monocular depth estimation using multi-scale continuous crfs as sequential deep networks. *TPAMI*.

Yuan, L.; Chen, Y.; Liu, H.; Kong, T.; and Shi, J. 2019. Zoom-in-to-check: Boosting video interpolation via instance-level discrimination. In *CVPR*.

Zhao, S.; Fu, H.; Gong, M.; and Tao, D. 2019. Geometry-aware symmetric domain adaptation for monocular depth estimation. In *CVPR*.