Supplementary Material

Yan Lu[1,†,*] Xinzhu Ma[1,*] Lei Yang[2] Tianzhu Zhang[3]
Yating Liu[4] Qi Chu[3,✉] Junjie Yan[2] Wanli Ouyang[1,✉]
[1]The University of Sydney, SenseTime Computer Vision Group [2]Sensetime Group Limited
[3]School of Information Science and Technology, University of Science and Technology of China
[4]School of Data Science, University of Science and Technology of China
{yan.lu1, xinzhu.ma, wanli.ouyang}@sydney.edu.au {yanglei, yanjunjie}@sensetime.com
{tzzhang, qchu}@ustc.edu.cn liuyat@mail.ustc.edu.cn

Table 1: **AP_{11} Performance of the Car category on the KITTI *validation* set.** We highlight the best results in **bold**.

Method	3D@IoU=0.7			BEV@IoU=0.7			3D@IoU=0.5			BEV@IoU=0.5		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
Mono3D [2]	2.53	2.31	2.31	5.22	5.19	4.13	-	-		-	-	
OFTNet [12]	4.07	3.27	3.29	11.06	8.79	8.91	-	-		-	-	
Deep3DBox [10]	5.85	4.19	3.84	9.99	7.71	5.30	27.04	20.55	15.88	30.02	23.77	18.83
FQNet [8]	5.98	5.50	4.75	9.50	8.02	7.71	28.16	28.16	28.16	32.57	24.60	21.25
Mono3D++ [5]	10.60	7.90	5.70	16.70	11.50	10.10	42.00	29.80	24.20	46.70	34.30	28.10
GS3D [6]	13.46	10.97	10.38	-	-		32.15	29.89	29.89	-	-	
MonoGRNet [11]	13.88	10.19	7.62	50.51	36.97	30.82	-	-		-	-	
MonoDIS [13]	18.05	14.98	13.42	24.26	18.43	16.95	-	-		-	-	
M3D-RPN [1]	20.27	17.06	15.21	25.94	21.18	21.18	48.96	39.57	33.01	53.35	39.60	31.76
RTM3D [7]	20.77	20.77	16.63	25.56	22.12	20.91	54.36	41.90	35.84	57.47	44.16	42.31
RARNet [9]+GS3D [6]	11.63	10.51	10.51	14.34	12.52	11.36	30.60	26.40	22.89	38.24	32.01	28.71
RARNet [9]+MonoGRNet [11]	13.84	10.11	7.59	24.84	19.27	16.20	50.27	36.67	30.53	53.91	39.45	32.84
RARNet [9]+M3D-RPN [1]	23.12	19.82	16.19	29.16	22.14	18.78	51.20	44.12	32.12	57.12	44.41	37.12
GUP Net (Ours)	**25.76**	**20.48**	**17.24**	**34.00**	**24.81**	**22.96**	**59.36**	**45.03**	**38.14**	**62.58**	**46.82**	**44.81**

0. 3D detection visualization

We show some detection results of our method. We first estimate the 3D bounding boxes by our method, and then project the boxes on the image plane to draw them on the scene image. The results are shown in Figure 1. We also give additional visualization of scores, we draw the 2D score p_{2d}, the depth score (3D scores conditioned on the 2D information) $p_{3d|2d}$ and the final 3D scores p_{3d} in Figure 2, showing that both p_{2d} and $p_{3d|2d}$ contribute to the final score and leading to better detection reliability.

1. Further Comparison

There are some works that do not report the newly released AP_{40} metric on the KITTI validation dataset. So we do not compare them on the validation set in the original paper. Here, we show the further comparison under AP_{11} metric on the validation set with the state-of-the-art methods in Table 1.

Under the AP_{11} setting, our method gets the best competing performance whatever the IoU threshold is. Specifically, our method surpasses RAR-Net [9] by 2.64%/4.84% AP under the easy setting at 0.7 IoU threshold. And for the 0.5 IoU threshold, we also achieve 8.16% and 5.46% gains for 3D/BEV detection under the easy setting respectively. This shows the high-precision performance of our method.

2. Loss function details

In this section, we would describe more details about loss functions.

2.1. 2D Detection loss function details

Our 2D detection module is built on the CenterNet [4]. Details can be seen in the original paper. The heatmap loss

†This work was done when Yan Lu was an intern at SenseTime.
*Equal contribution.
✉Corresponding authors.

Figure 1: Projected 3D Detection results of our method. We use green, blue and red to indicate car,pedestrian and cyclist categories, respectively.

Figure 2: Visualization of different kinds of scores.

is defined as:

$$\mathcal{L}_{heatmap} = \text{Focal}(Y, Y^{gt}) = \frac{-1}{WHC} \sum_{uvc} \begin{cases} (1 - Y_{uvc})^{\alpha} \log(Y_{uvc}), & \text{if } Y^{gt}_{uvc} = 1 \\ (1 - Y^{gt}_{uvc})^{\beta} Y^{\alpha}_{uvc} \log(1 - Y_{uvc}), & \text{otherwise} \end{cases}, \tag{1}$$

where Y represents the predicted heatmaps and Y^{gt} is the ground-truth heatmaps.

The 2D size loss function is defined as:

$$\mathcal{L}_{size2d} = -\frac{1}{N} \sum_{k=1}^{N} \left(|w_{2d}(k) - w^{gt}_{2d}(k)| + |h_{2d}(k) - h^{gt}_{2d}(k)| \right), \tag{2}$$

where N is the total ground-truth object number on the current batch. The w^{gt}_{2d} and h^{gt}_{2d} are ground-truth size of the k-th object.

The 2D offset loss function is:

$$\mathcal{L}_{offset2d} = \frac{1}{N} \sum_{k=1}^{N} \left(|\delta^{u}_{2d}(k) - \delta^{u,gt}_{2d}(k)| + |\delta^{v}_{2d}(k) - \delta^{v,gt}_{2d}(k)| \right). \tag{3}$$

The $\delta^{u,gt}_{2d}(k)$ and $\delta^{v,gt}_{2d}(k)$ are the ground-truth offsets of the k-th object.

2.2. Basic 3D Detection loss function details

The basic 3D detection loss function of our method follows the MonoPair [3]. The angle prediction task aims to predict the alpha angle [10, 14], which is the relative rotation of the camera viewing angle. The original 360° angle range is split into 12 bins and each bin covers 30° range. Its loss function is the *MultiBin* loss [10]:

$$\mathcal{L}_{angle} = \mathcal{L}_{conf}(\theta, \theta^{gt}) + \mathcal{L}_{loc}(\theta, \theta^{gt}). \tag{4}$$

And for the the ROI 3D demension estimation, its loss function is:

$$\mathcal{L}_{size3d} = \mathcal{L}_{h3d} + |w_{3d} - w^{gt}_{3d}| + |l_{3d} - l^{gt}_{3d}|. \tag{5}$$

Finally, the ROI based 3D offset loss function is:

$$\mathcal{L}_{offset3d} = |\delta^{u}_{3d}(k) - \delta^{u,gt}_{3d}(k)| + |\delta^{v}_{3d}(k) - \delta^{v,gt}_{3d}(k)|. \tag{6}$$

2.3. Hierarchical Task Learning details

In this subsection, we would give some computation details of the proposed Hierarchical Task Learning (HTL). We

first give a brief review of the HTL. Our HTL strategy assigns loss weights by computing the learning situation for each task. And the learning situation indicator (Eq 7 in the supplementary material. Eq 10 in the original paper.) essentially compares the mean loss trends between the current K epochs and the first K epochs. It can be interpreted as comparing information in two sliding windows as shown in Figure 3. In the following, we would give the details about the algorithm initialization and the learning indicator.

Initialization. The learning situation indicator requests both the current and the first sliding windows information. So before getting the first K epochs information, the HTL cannot be started. So we set the first K epochs (K is set as 5, which is equal to the warm-up period length.) as the initialization period of the HTL. Also, the first 5 epochs are also the warm-up period of the total model. In this period, the loss weights of the 2nd and 3rd task stages are all set to zero. And the HTL gathers the loss values and stores them as the first sliding window information.

Learning situation indicator. We give the equation of the learning situation indicator again here:

$$\begin{aligned} ls_j(t) &= \frac{\mathcal{DF}_j(K) - \mathcal{DF}_j(t)}{\mathcal{DF}_j(K)}, \\ \mathcal{DF}_j(t) &= \frac{1}{K} \sum_{\hat{t}=t-K}^{t-1} |\mathcal{L}'_j(\hat{t})|, \end{aligned} \tag{7}$$

where $\mathcal{L}'_j(t)$ is the loss derivative of the j-th task at t-th epoch. We give an online computation way about that here:

$$\begin{aligned} \mathcal{L}'_j(t) &= \underbrace{\frac{\mathcal{L}_j(t+\Delta t) - \mathcal{L}_j(t)}{\Delta t}}_{\text{left derivative}} + \underbrace{\frac{\mathcal{L}_j(t) - \mathcal{L}_j(t-\Delta t)}{\Delta t}}_{\text{right derivative}} \\ &= \frac{\mathcal{L}_j(t+\Delta t) - \mathcal{L}_j(t-\Delta t)}{2\Delta t} \\ &\overset{\Delta t \to 1}{=} \frac{\mathcal{L}_j(t+1) - \mathcal{L}_j(t-1)}{2}, \end{aligned} \tag{8}$$

where the $\mathcal{L}_j(t)$ means the averaged loss function value of the t-th epoch. At the time border of the sliding window, we only compute one-sided derivative.

To better understand the learning situation indicator, we show an example in Figure 3. As the loss function decreases, the value of the learning situation gradually increases to 1. And the learning situation at t-th epoch is decided by both the first K epochs information and the current K epochs trends.

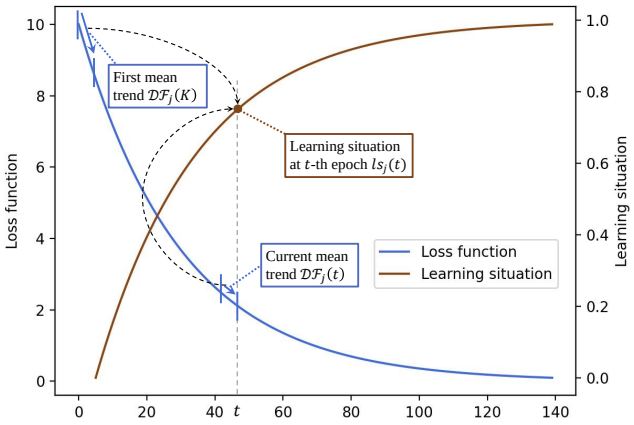

Figure 3: An example of learning situation computation pipeline. The blue line is the loss curve and the brown line means the learning situation value curve. The learning situation at t-th epoch is computed by comparing the current averaged trend $\mathcal{DF}_j(t)$ and the first averaged trend $\mathcal{DF}_j(K)$ of the loss funtcion.

References

[1] Garrick Brazil and Xiaoming Liu. M3d-rpn: Monocular 3d region proposal network for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9287–9296, 2019.

[2] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2156, 2016.

[3] Yongjian Chen, Lei Tai, Kai Sun, and Mingyang Li. Monopair: Monocular 3d object detection using pairwise spatial relationships. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12093–12102, 2020.

[4] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6569–6578, 2019.

[5] Tong He and Stefano Soatto. Mono3d++: Monocular 3d vehicle detection with two-scale 3d hypotheses and task priors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8409–8416, 2019.

[6] Buyu Li, Wanli Ouyang, Lu Sheng, Xingyu Zeng, and Xiaogang Wang. Gs3d: An efficient 3d object detection framework for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1019–1028, 2019.

[7] Peixuan Li, Huaici Zhao, Pengfei Liu, and Feidao Cao. Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving. *arXiv preprint arXiv:2001.03343*, 2, 2020.

[8] Lijie Liu, Jiwen Lu, Chunjing Xu, Qi Tian, and Jie Zhou. Deep fitting degree scoring network for monocular 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1057–1066, 2019.

[9] Lijie Liu, Chufan Wu, Jiwen Lu, Lingxi Xie, Jie Zhou, and Qi Tian. Reinforced axial refinement network for monocular 3d object detection. In *European Conference on Computer Vision*, pages 540–556. Springer, 2020.

[10] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7074–7082, 2017.

[11] Zengyi Qin, Jinglu Wang, and Yan Lu. Monogrnet: A geometric reasoning network for monocular 3d object localization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8851–8858, 2019.

[12] Thomas Roddick, Alex Kendall, and Roberto Cipolla. Orthographic feature transform for monocular 3d object detection. *arXiv preprint arXiv:1811.08188*, 2018.

[13] Andrea Simonelli, Samuel Rota Bulo, Lorenzo Porzi, Manuel López-Antequera, and Peter Kontschieder. Disentangling monocular 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1991–1999, 2019.

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