Supplementary Material:
Sparse Fuse Dense: Towards High Quality 3D Detection with Depth Completion

Here we provide more details about our method. An overview of this supplementary material is as follow:

- Section A: Additional Ablation Studies
  - A.1 Auxiliary Heads
  - A.2 Grid Size for Pseudo Clouds Voxelization
  - A.3 Comparison of PointNet++ and CPConv
  - A.4 Number of CPConv
  - A.5 Robustness Analysis on Depth Completion
- Section B: Additional Discussions
  - B.1 Analysis of Raw Clouds and Pseudo Clouds
  - B.2 How About Using Pseudo Clouds in RPN?
- Section C: Additional Details
  - C.1 Dilated Neighbor
  - C.2 Experiment Settings
- Section D: KITTI Detection Leaderboard Screenshot

A. Additional Ablation Studies

A.1. Auxiliary Heads

To prevent gradients from being dominated by a particular *Stream*, we employ two auxiliary heads on *LiDAR Stream* and *Pseudo Stream*, respectively. As seen in Table 1, each auxiliary head can improve our SFD. With two auxiliary heads employed, we can achieve a better result.

P.A.H.	R.A.H.	AP_{3D}		
		Easy	Mod.	Hard
		94.94	88.12	85.30
✓		95.36	88.26	85.62
	✓	95.40	88.38	85.72
✓	✓	**95.47**	**88.56**	**85.74**

Table 1. Ablation study on auxiliary heads. "P.A.H." and "R.A.H." stand for Pseudo Auxiliary Head and Raw Auxiliary Head.

A.2. Grid Size for Pseudo Clouds Voxelization

At the end of *Pseudo Stream*, we voxelize pseudo clouds in a 3D RoI to a $G \times G \times G$ RoI feature. Then we use 3D sparse convolutions to extract grid-wise features in the RoI. Finally, the grid size of RoI feature is converted to $6 \times 6 \times 6$, which is consistent with the grid size of the raw RoI feature. The grid size of raw RoI features has been fine-tuned by Voxel-RCNN [1], so we do not change it. Here G is a multiple of 6. We conduct experiments to choose a proper G, as shown in Table 2. With G = 12, our SFD can achieve a good performance, so we let G as 12.

Grid Size	AP_{3D}		
	Easy	Moderate	Hard
6	95.50	88.47	85.64
12	95.47	88.56	85.74
24	95.38	88.39	85.91

Table 2. Ablation study on different grid sizes.

A.3. Comparison of PointNet++ and CPConv

Because of the massive pseudo points, it is impossible to perform PointNet++ [6] on all pseudo points. Thus, we need to downsample pseudo clouds to utilize PointNet++. In our experiment, we sample 1024 pseudo points in each 3D RoI. Table 3 provides a comparison of PointNet++ and CPConv. With more time, PointNet++ performs much worse than CPConv. We summarize the reasons as the following two points. *Firstly*, PointNet++ cannot make use of 2D semantic features in pseudo clouds due to the ball query. *Secondly*, downsampling used by PointNet++ causes a lot of information loss, while in CPConv, we can keep most pseudo points in 3D RoIs thanks to the fast neighbor search.

Method	Inference Time	AP_{3D}		
		Easy	Moderate	Hard
PointNet++	95ms	92.25	85.61	83.36
CPConv	**12ms**	**95.47**	**88.56**	**85.74**

Table 3. Comparison of PointNet++ and CPConv. The results are evaluated with AP calculated by 40 recall positions for car class.

A.4. Number of CPConv

In our SFD, we stack several CPConvs to extract high-level features of pseudo clouds. Table 4 shows the results of SFD with different numbers of CPConvs. As the number of CPConv increases, the performance gradually improves. When we use four CPConvs, the performance is not improved. Thus, we only use three CPConvs for a balance of accuracy and efficiency.

Number	AP_{3D}		
	Easy	Moderate	Hard
1	95.40	88.24	85.73
2	95.52	88.29	85.71
3	95.47	88.56	85.74
4	93.53	88.50	85.83

Table 4. Ablation study on different numbers of CPConvs.

A.5. Robustness Analysis on Depth Completion

To validate the robustness of our method to depth completion networks, we utilize four different depth completion networks on our SFD, as seen in Table 5. The RMSE (root mean squared error) is the most important metric for evaluating the performance of depth completion methods. The 3D detection results in Table 5 show that with different depth completion networks, our method can still maintain a high performance. Interestingly, although TWISE [3] performs worst in the four depth completion networks on the RMSE metric, it works very well in our 3D detection framework.

Depth Completion Network	RMSE ↓	AP_{3D}		
		Easy	Moderate	Hard
PENet [2]	730.08	95.47	88.56	85.74
MSG-CHN [4]	762.19	95.10	87.84	85.24
Sparse-to-Dense [5]	814.73	95.20	88.19	85.59
TWISE [3]	840.20	95.41	88.58	86.01

Table 5. Robustness experiment with different depth completion networks. The depth completion results are from the KITTI depth completion leaderboard. ↓ is for lower better. The detection results are evaluated on the KITTI *val* set.

Statistics	0m-10m	10m-20m	20m-30m	30m-40m	40-50m	50m-Inf
Pseudo point number	29023	8739	2101	896	504	218
Raw point number	2522	594	155	59	29	11
Ratio (Pseudo / Raw)	11.5	14.7	13.6	15.2	17.4	19.8

Table 6. The statistics of pseudo points and raw points on objects in different distance ranges.

B. Additional Discussions

B.1. Analysis of Raw Clouds and Pseudo Clouds

To quantitatively compare the raw clouds and pseudo clouds, we count the average number of raw points and pseudo points on objects within different distances, as shown in Table 6 and Figure 1. In different distances, pseudo points are always more than ten times the raw points, demonstrating

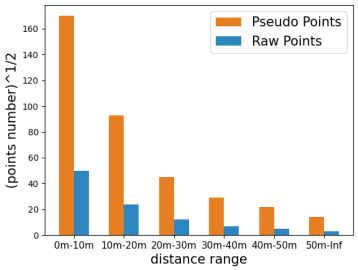

Figure 1. The average number of pseudo points and raw points on objects in different distances. For the convenience of visualization, we use the square of the number of points as the vertical axis.

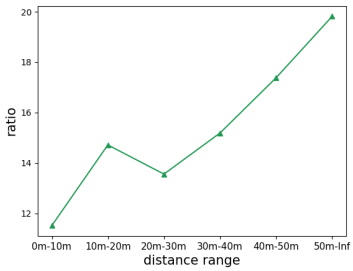

Figure 2. The ratio of pseudo point number and raw point number on objects in different distance ranges.

Figure 3. Comparison of raw clouds and pseudo clouds.

the rationality of enhancing raw clouds with pseudo clouds. In addition, we observe that the number of pseudo clouds decays slower than raw clouds with depth increasing, as shown in Figure 2, which indicates that pseudo clouds are more advantageous than raw clouds for long-distance object detection. We also provide a qualitative comparison between different types of clouds, as shown in Figure 3. Pseudo point clouds can provide sufficient information, especially for distant and occluded objects.

B.2. How About Using Pseudo Clouds in RPN?

In our SFD, pseudo clouds and raw clouds are fused in RCNN with our 3D-GAF. The pseudo clouds are not used in RPN. The reasons are three-folds.

Firstly, using pseudo clouds in RPN will increase much computational overhead due to the huge number of pseudo points. In our SFD, we only need process pseudo points in 3D RoIs, which can save much time.

Secondly, the compatibility of SFD with LiDAR-only methods will be weakened. With only RoI feature fusion, we don't need to modify anything of LiDAR-only detectors, which enables our SFD to be compatible with almost all LiDAR-only methods, including one-stage methods, two-stage methods, point-based methods, voxel-based methods and point-voxel-based methods.

Thirdly, most objects missed by the model are those that RPN can roughly detect but cannot accurately detect $(IoU(dt, gt) \in [0.3, 0.7))$, rather than those that RPN can barely detect $(IoU(dt, gt) \in [0, 0.3))$. Table 7 shows that with IoU threshold 0.3, the RPN recall of SFD can reach 98%, while with IoU threshold 0.7, the recall of RPN and RCNN is only about 80%. It indicates that RPN can roughly detect most objects with only raw clouds, but it fails to give accurate predictions. Therefore, we argue that the top priority of current 3D detection methods is recalling those objects (about $98\% - 80\% = 18\%$) that can be roughly detected instead of recalling those barely detectable objects (only $100\% - 98\% = 2\%$). So in our method, we concentrate on refining the proposals generated from RPN. As shown in Table 7, our method improves the RCNN recall by 4.4%. In addition, we find that the RPN recall of SFD is higher than that of Voxel-RCNN, suggesting that our method can optimize RPN implicitly.

Method	$\text{RPN}_{0.3}$	$\text{RPN}_{0.7}$	$\text{RCNN}_{0.7}$
Voxel-RCNN	95.5%	76.1%	78.2%
SFD(ours)	**98.1%**	**77.2%**	**82.6%**

Table 7. Recall of different stages with different IoU thresholds.

C. Additional Details

C.1. Dilated Neighbor

When performing CPConv, we search nine neighbors (including self) for each pseudo point in 3D RoIs on the image domain. In practice, we search *dilated neighbors* instead of *nearest neighbors*, as shown in the Figure 4. The reasons are two-folds.

Firstly, dilated neighbor search provides a large receptive field [7]. Stacking more CPConvs can provide a larger receptive field while calculations are also increased. Dilated neighbor search can alleviate this issue. The receptive field of the dilated neighbor search is twice that of the nearest neighbor search, allowing our SFD to get a larger receptive field without increasing computation overhead.

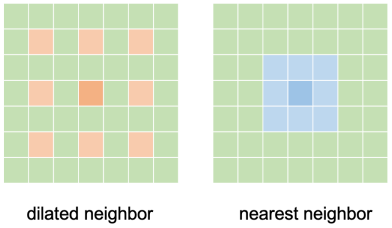

Figure 4. Illustration of dilated neighbor and nearest neighbor.

Secondly, dilated neighbor search enables us to downsample nearby pseudo clouds. We count the number of pseudo points within 15m and find that these points account for about 70% of the total, while current 3D detection methods perform very well in this range. It is unnecessary to spend too many calculations on the close range. Therefore, before feeding pseudo clouds to our CPConv, we perform downsampling on the nearby pseudo clouds. Concretely, for pseudo points within 15m, we remove those whose u or v is even. In this way, only 25% of nearby pseudo clouds are left, saving about $70\% \times (1 - 25\%) \approx 53\%$ calculations. However, the downsampling makes nearest neighbors of nearby pseudo points are all discarded. Fortunately, their dilated neighbors are left. So dilated neighbor search is more suitable than nearest neighbor search in this situation.

C.2. Experiment Settings

Voxelization The raw clouds are divided into regular voxels before being fed into *LiDAR Stream*. Because the KITTI dataset only provides annotations in the FOV, we clip the range of raw clouds into $[0, 70.4]m$ for the X axis, $[-40, 40]m$ for the Y axis and $[-3, 1]m$ for the Z axis. The input voxel size is set as $(0.05m, 0.05m, 0.1m)$. In addition, we clip the range of pseudo clouds into $[-3, 1]m$ for Z axis.

Training Our SFD is optimized with the Adam optimizer and trained for 40 epochs with a batch size of 8. The learning rate is initialized as 0.01 and updated by cosine annealing strategy. In the RoI head, binary cross entropy loss and 3D GIoU Loss [8] are employed for classification and regression loss. The foreground IoU threshold θ_H, background IoU threshold θ_L, box regression IoU threshold θ_{reg} is set as 0.75, 0.25 and 0.55. We randomly sample 128 proposals as the training samples of the RoI head.

Inference At the inference stage, the two auxiliary heads are detached. We perform NMS on the proposals generated from the RPN with IoU threshold 0.7 and keep the top-100 proposals as the input of the RoI head. After the refinement stage, we use an NMS threshold of 0.1 to remove the redundant detections.

D. KITTI Detection Leaderboard Screenshot

We validate our method by submitting our results to KITTI online test server. As shown in Figure 5 and Figure 6, our SFD ranks 1^{st} and 3^{rd} on the KITTI car 3D and BEV detection leaderboard.

	Method	Setting	Code	Moderate	Easy	Hard	Runtime
1	SFD			**84.76 %**	**91.73 %**	77.92 %	0.1 s
2	Anonymous			83.96 %	90.83 %	77.47 %	n/a s
3	DGDNH			83.88 %	90.69 %	**79.50 %**	0.04 s
4	VPFNet			83.21 %	91.02 %	78.20 %	0.06 s
5	Anonymous			82.99 %	91.64 %	78.02 %	0.1 s
6	NFAF3D			82.97 %	91.57 %	77.72 %	0.06 s
7	BtcDet			82.86 %	90.64 %	78.09 %	0.09 s
8	Anonymous			82.79 %	91.30 %	78.07 %	n/a s
9	PE-RCVN			82.69 %	91.51 %	77.75 %	0.03 s
10	SPG_mini			82.66 %	90.64 %	77.91 %	0.09 s
11	DSASNet			82.63 %	89.48 %	77.94 %	0.08 s
12	SE-SSD		code	82.54 %	91.49 %	77.15 %	0.03 s
W. Zheng, W. Tang, L. Jiang and C. Fu: SE-SSD: Self-Ensembling Single-Stage Object Detector From P							
13	TF3D			82.46 %	89.10 %	77.78 %	0.1 s
14	DVF-V			82.45 %	89.40 %	77.56 %	0.1 s
15	DVF-PV			82.40 %	90.99 %	77.37 %	0.1 s

Figure 5. Screenshot of the KITTI car 3D detection leaderboard on the date of CVPR deadline, *i.e.*, Nov 16, 2021.

	Method	Setting	Code	Moderate	Easy	Hard	Runtime
1	NFAF3D			**91.97 %**	**95.72 %**	86.97 %	0.06 s
2	VPFNet			91.86 %	93.02 %	86.94 %	0.06 s
3	SFD			91.85 %	95.64 %	86.83 %	0.1 s
4	SE-SSD		code	91.84 %	95.68 %	86.72 %	0.03 s
W. Zheng, W. Tang, L. Jiang and C. Fu: SE-SSD: Self-Ensembling Single-Stage Object Detector From P							
5	CityBrainLab			91.60 %	94.75 %	86.67 %	0.04 s
6	SPANet			91.59 %	95.59 %	86.53 %	0.06 s
Y. Ye: SPANet: Spatial and Part-Aware Aggregation Network for 3D Object Detection. Pacific Rim Inter							
7	Anonymous			91.53 %	95.04 %	86.69 %	0.1 s
8	Anonymous			91.38 %	95.23 %	86.71 %	n/a s
9	DGDNH			91.36 %	95.03 %	**88.79 %**	0.04 s
10	BANet		code	91.32 %	95.23 %	86.48 %	0.11 s
R. Qian, X. Lai and X. Li: Boundary-Aware 3D Object Detection from Point Clouds. 2021.							
11	Anonymous			91.04 %	94.31 %	86.31 %	0.1s
12	Anonymous			91.04 %	94.76 %	86.31 %	n/a s
13	SA-SSD		code	91.03 %	95.03 %	85.96 %	0.04 s
C. He, H. Zeng, J. Huang, X. Hua and L. Zhang: Structure Aware Single-stage 3D Object Detection fror							
14	MMLab PV-RCNN		code	90.65 %	94.98 %	86.14 %	0.08 s
S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang and H. Li: PV-RCNN: Point-Voxel Feature Set Abstract							
15	VueronNet		code	90.56 %	94.67 %	85.31 %	0.06 s

Figure 6. Screenshot of the KITTI car BEV detection leaderboard on the date of CVPR deadline, *i.e.*, Nov 16, 2021.

References

[1] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. *arXiv preprint arXiv:2012.15712*, 2020. 1

[2] Mu Hu, Shuling Wang, Bin Li, Shiyu Ning, Li Fan, and Xiaojin Gong. Penet: Towards precise and efficient image guided depth completion. *arXiv preprint arXiv:2103.00783*, 2021. 2

[3] Saif Imran, Xiaoming Liu, and Daniel Morris. Depth completion with twin surface extrapolation at occlusion boundaries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2583–2592, 2021. 2

[4] Ang Li, Zejian Yuan, Yonggen Ling, Wanchao Chi, Chong Zhang, et al. A multi-scale guided cascade hourglass network for depth completion. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 32–40, 2020. 2

[5] Fangchang Ma, Guilherme Venturelli Cavalheiro, and Sertac Karaman. Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3288–3295. IEEE, 2019. 2

[6] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, pages 5099–5108, 2017. 1

[7] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 3

[8] Dingfu Zhou, Jin Fang, Xibin Song, Chenye Guan, Junbo Yin, Yuchao Dai, and Ruigang Yang. Iou loss for 2d/3d object detection. In *2019 International Conference on 3D Vision (3DV)*, pages 85–94. IEEE, 2019. 3