The supplementary material to: "ZAHA: Introducing the Level of Facade Generalization and the Large-Scale Point Cloud Facade Semantic Segmentation Benchmark Dataset"

A. Experiments

A.1. Evaluation Metrics

To evaluate the performance of the 3D facade segmentation, we used the established semantic segmentation network metrics, such as Overall Accuraccy, Precision, Recall, and Jaccard Index also known as Intersection over Union (IoU) [2]. They were defined as follows:

$$\text{Overall Accuracy} = \frac{\text{True Positives + True Negatives}}{\text{Total Instances}}$$

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision + Recall}}$$

$$\text{IoU} = \frac{\text{Intersection Area}}{\text{Union Area}} = \\ = \frac{\text{True Positives}}{\text{True Positives + False Positives + False Negatives}}$$

A.2. Parameter Settings

We conducted all the experiments using an NVIDIA GeForce RTX 4090 GPU with 16 GB VRAM with a fixed number of 100 epochs per training. The implementation will be released under our repository web page.[1]

To train the PointNet and PointNet++, we utilized the implementation sourced from [3, 4, 9]. We used the point cloud coordinates as input layers to adapt the model and modify the corresponding classes. We employed a batch size of 32 for training and testing, and each batch contains 1024 points per sample point cloud. Stochastic gradient descent with a momentum of 0.1 and a learning rate 0.001 was employed.

[1]Project page: https://github.com/OloOcki/zaha

To train the Point Transformer, we used the implementation of the first author of Point Transformer [11], in [10]. Diverging from the voxelization method employed in the source code to generate batches, we have opted for a batch design strategy based on index segmentation. Consistent with previous experiments, we also utilized a batch size of 32 as input and 1024 points per sample point cloud. Stochastic gradient descent with a momentum of 0.9 and an initial learning rate of 0.1 was employed. After 60 epochs, the learning rate would reduced to 0.01, and after 80 epochs, 0.001.

To train the DGCNN, we utilized the implementation sourced from [5, 8]. We customized their implementation by adjusting the dimensions of the input and output layers to suit the dimensionality and the number of classes. During training, we employed a batch size of 32, while for testing, we used a batch size of 16, each containing 1024 points per sample point cloud. Stochastic gradient descent with a momentum of 0.9 and a learning rate of 0.1 was employed. We set the dropout rate to 0.5 and the number of nearest neighbors that we considered to 20.

A.3. Extra Baseline Experiment on a Large-Scale-Oriented Network

Owing to the space limitation and similar performance scores to the other networks, we have moved the extra experiments on the large-scale-oriented KPConv [6] network to the supplemental material. Here, we show the extra set of experiments that we conducted on the KPConv network, whereby we also fine-tuned the hyper-parameters. KPConv introduces a deformable convolution operation, allowing the neural network to learn flexible and adaptive convolutional filters. The use of kernel points in KPConv allows for more efficient processing of point clouds, and as such, it has often been used in the context of large-scale, outdoor point clouds [1]. However, corroborating our experiment results in the main paper, there were no significant performance differences observed, as we visualize in Figure 1 and Figure 2, and list in Table 1 and Table 2.

To train the KPConv, we employed the implementation sourced from [6,7]. The input radius of the input sphere was

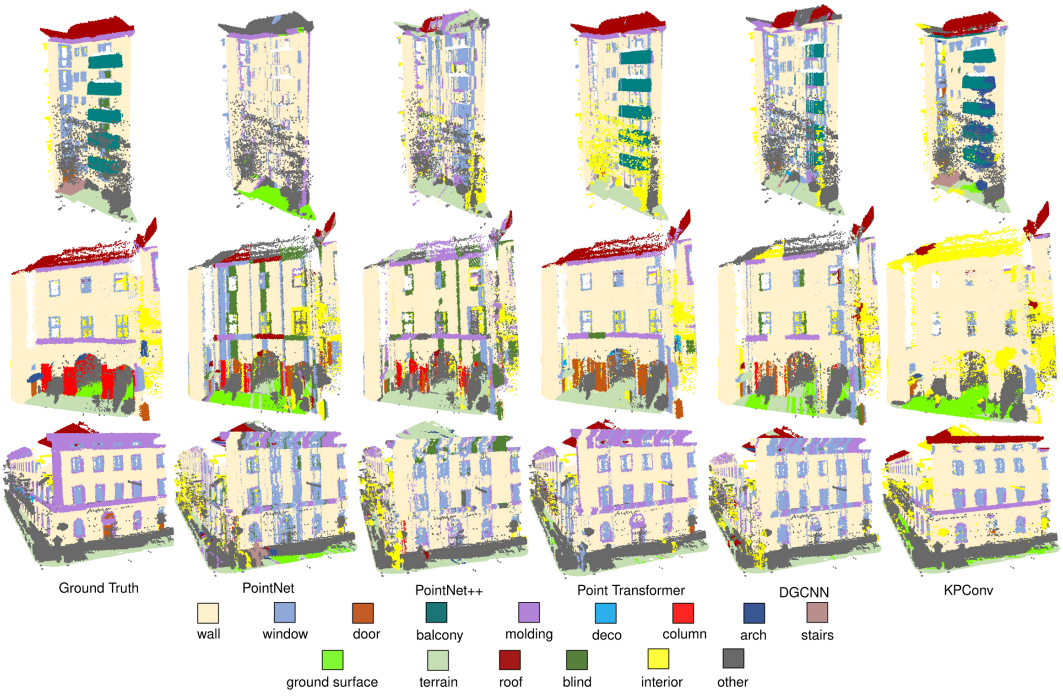

Figure 1. LoFG3 inference on the test set comprising residential (top), underpass and cultural heritage (middle), and university buildings (bottom) with the extra fine-tuned KPConv.

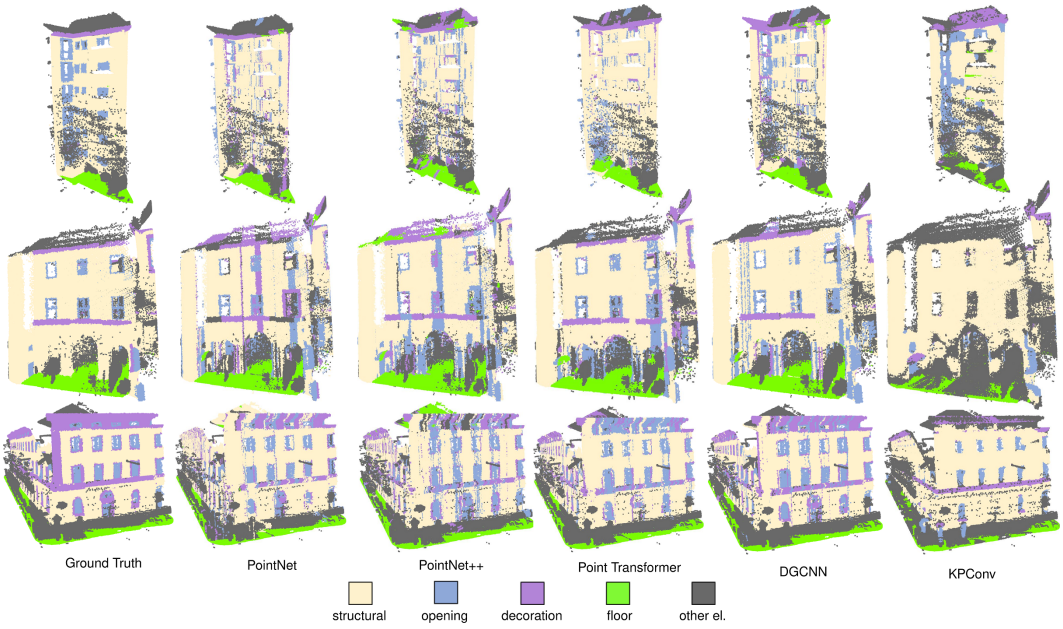

Figure 2. LoFG2 inference on the test set comprising residential (top), underpass and cultural heritage (middle), and university buildings (bottom); color-coding according to the most prominent merged sub-class with the extra fine-tuned KPConv.

set as 1.5m. We also generated the radius of deformable convolution in the "number grid cell" as 1.5m, to minimize noisy clusters. For both baselines, we set the epoch steps as 2000 and the validation size as 100. The batch size was set to 6 for training and 1 for testing. For the training of LoFG2, stochastic gradient descent with a momentum of 0.98 and a learning rate of 0.001 was employed, and there were 150 training epochs. For the training of LoFG3, stochastic gra-

Table 1. LoFG3 test: OA, μP, μR, μF1, μIoU, and F1 scores per class, in percentages; with the extra fine-tuned KPConv

Method	PointNet	PointNet++	Point Transformer	DGCNN	KPConv
OA	59.9	66.4	75.0	71.1	65.2
μP	46.1	37.8	52.7	53.6	46.4
μR	42.2	35.9	54.7	45.8	44.6
μF1	38.7	34.8	52.1	44.5	39.3
μIoU	26.4	25.6	41.6	33.4	28.6
wall	61.1	68.5	76.8	83.8	66.6
window	25.6	26.3	43.1	64.1	41.0
door	13.5	7.8	19.8	21.6	9.6
balcony	25.1	0.0	77.5	66.7	61.7
molding	22.5	43.4	58.0	57.5	23.3
deco	0.0	0.0	5.0	0.0	0.0
column	22.4	33.4	0.0	37.2	0.0
arch	19.2	25.4	50.2	2.6	11.5
stairs	16.0	0.0	7.5	5.6	6.5
ground surface	12.0	0.0	24.4	21.3	26.3
terrain	53.5	53.5	57.6	68.0	31.4
roof	18.7	6.8	66.3	57.4	15.4
blinds	4.6	2.3	18.5	20.0	10.0
interior	59.7	69.1	72.8	88.0	75.1
other	42.7	47.1	70.6	74.1	50.0

Table 2. LoFG2 test: OA, μP, μR, μF1, μIoU, and F1 scores per class, in percentages; with the extra fine-tuned KPConv

Method	PointNet	PointNet++	Point Transformer	DGCNN	KPConv
OA	71.9	75.5	78.2	82.6	71.6
μP	69.6	73.0	75.8	80.0	71.2
μR	68.1	73.0	76.6	81.8	64.3
μF1	68.1	72.6	76.1	80.4	66.4
μIoU	55.8	59.8	63.9	68.5	52.3
floor	92.3	87.6	90.7	92.1	80.1
decoration	26.2	47.1	47.0	70.0	28.2
structural	60.9	65.5	67.0	85.2	62.4
opening	28.2	27.2	36.0	66.2	31.7
other el.	71.2	71.6	78.9	88.8	58.5

dient descent with a momentum of 0.98 and a learning rate of 0.01 was employed, and the training epochs were set as the default number 500.

A.4. Benchmark and Leaderboard

We introduce the ZAHA as a benchmark to foster the research on facade semantic segmentation. It is a common practice to publish a leaderboard, which encourages researchers to delve into a challenge. We initialize the leaderboard at the webpage [2] and invite researchers to develop novel and more efficient facade segmentation methods.

A.5. Extra visuals

Additional visuals are included at the project page [3] showing an animated gif file, and the full dataset and annotations according to the 15 introduced classes.

References

[1] Martin Kada and Dmitry Kuramin. Als point cloud classification using pointnet++ and kpconv with prior knowledge. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 46:91–96, 2021. 1

[2] Ying Li, Lingfei Ma, Zilong Zhong, Fei Liu, Michael A Chapman, Dongpu Cao, and Jonathan Li. Deep learning for lidar point clouds in autonomous driving: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8):3412–3432, 2020. 1

[3] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3d classification and segmentation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017. 1

[4] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing System (NeurIPS)*, 30, 2017. 1

[5] An Tao. dgcnn.pytorch. https://github.com/antao97/dgcnn.pytorch, 2024. Accessed: 2024-03-06. 1

[6] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6411–6420, 2019. 1

[7] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. KPConv pytorch. https://github.com/HuguesTHOMAS/KPConv-PyTorch, 2019. Accessed: 2024-03-06. 1

[8] Yue Wang, Yongbin Sung, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics*, 38(5):1–12, 2019. 1

[9] Xu Yan. Pointnet/pointnet++ pytorch. https://github.com/yanx27/Pointnet_Pointnet2_pytorch, 2019. Accessed: 2024-03-06. 1

[10] Hengshuang Zhao. point-transformer. https://github.com/POSTECH-CVLab/point-transformer, 2021. Accessed: 2024-03-06. 1

[11] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16259–16268, 2021. 1

[2] Leaderboard: https://tum2t.win/benchmarks/pc-fac

[3] Project page: https://github.com/OloOcki/zaha