Supplementary Material for RoofDiffusion

Kyle Shih-Huang Lo[1], Jörg Peters[1], and Eric Spellman[2]

[1] University of Florida, Gainesville FL 32611, USA
[2] Meta Platforms, Inc.
kyleshihhuanglo@ufl.edu, jorg@cise.ufl.edu, espellman@meta.com

This supplement contains:

1 Selection of Hyper-parameters

1.1 Roof Height Normalization

Our choice of 10-meters as the upper bound on the difference between maximum minus minimum roof height, for mapping into the range of $[-1, 1]$, is justified by the distribution of roof height differences in the PoznanRD, see Fig. 1: 99% of the height differences are less or equal to 10 meters, marked as the black dash line. For the buildings with height differences exceeding 10 meters, we scale them to 10 meters and record the scaling factor for recovering the true height.

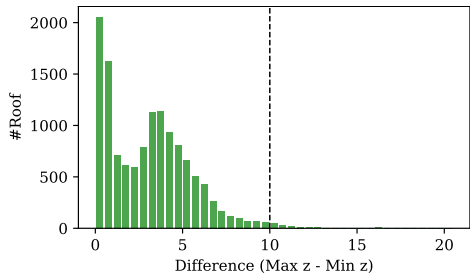


Fig. 1: Distribution of height differences and roof counts.

1.2 Notations

As in the main document, in this supplement, "s" represents Sparsity (%), the ratio of randomly removed to total pixels in the footprint. Furthermore, "i" denotes the Incompleteness ratio (%), the proportion of pixels removed due to incompleteness in the footprint. Lastly, "t" signifies the Tree count used to synthesize tree noise.

2 PoznanRD Dataset

2.1 Data Balancing

We started with collecting 16k compact and high-detail LoD 2.2 [2] roof meshes from the city of Poznan in Poland [6]. To match our focus on complex roof structures, we rebalanced the dataset by excluding 3k flat roofs. This resulted in our PoznanRD comprising 13k buildings. The original longitude, latitude, height, and scaling for each roof are recorded in a Comma-Separated Values (CSV) file for easy access.

Algorithm 1 Incompleteness Mask (Training)

set $\boldsymbol{m}_{\text{incomplete}}$ **to** ZEROS(H, W)
for $g = 1 \dots G$ **do** ▷ Index of Gauss distribution.
  for $i = 1 \dots H$ **do** ▷ Image height.
    for $j = 1 \dots W$ **do** ▷ Image width.
      $p \sim \mathcal{U}(0, 1)$
      if $p_{\text{Gauss},g}(i, j) > p$ **then**
        $\boldsymbol{m}_{\text{incomplete}}(i, j) = 1$
      end if
    end for
  end for
end for
$\boldsymbol{m}_{\text{incomplete}} = \boldsymbol{m} \odot \boldsymbol{m}_{\text{incomplete}}$ ▷ Set outside footprint to zero.

Algorithm 2 Incompleteness Mask (Benchmark)

set $\boldsymbol{m}_{\text{incomplete}}$ **to** ZEROS(H, W)
while SUM(M) $< N_{\text{thres}}$ **do**
  $(i, j) \sim p_{\text{GMM}}(i, j)$
  if $\boldsymbol{m}(i, j) == 1$ **then** ▷ Should be within footprint.
    $\boldsymbol{m}_{\text{incomplete}}(i, j) = 1$
  end if
end while

2.2 Synthesizing Incompleteness

We show the detail of generating a binary mask for synthesizing incompleteness, denoted as $\boldsymbol{m}_{\text{incomplete}}$. In this mask, a value of 1 indicates the pixels that will be removed, thereby yielding incompleteness, otherwise, 0. First, we employ a Gaussian Mixture Model (GMM) [23] to establish a probability density function,

$$p_{\text{GMM}}(i, j) := \sum_{g=1}^{G} \frac{1}{G} \underbrace{e^{-\frac{\|(i,j)-\boldsymbol{\mu}_g\|^2}{2\boldsymbol{\sigma}_g^2}}}_{p_{\text{Gauss},g}}, \tag{1}$$

where i and j are the pixel coordinate in x and y direction of right-handed coordinate system, respectively, and

$$\mu_{g,x} \sim \mathcal{U}(1, H) \quad \mu_{g,y} \sim \mathcal{U}(1, W), \tag{2}$$

$$\sigma_{g,x} \sim \mathcal{U}(\sigma_{\min}, \sigma_{\max}), \quad \sigma_{g,y} \sim \mathcal{U}(\sigma_{\min}, \sigma_{\max}), \tag{3}$$

$$\boldsymbol{\mu}_g = (\mu_{g,x}, \mu_{g,y}), \tag{4}$$

$$\boldsymbol{\sigma}_g = (\sigma_{g,x}, \sigma_{g,y}), \tag{5}$$

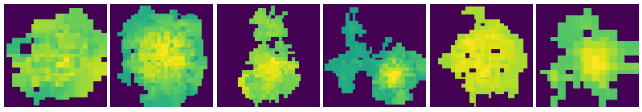


Fig. 2: Examples of tree height map in PoznanRDdataset. (The examples are color mapped and resized for clear visualization. The height increases as the color becomes more yellow.)

Algorithm 3 Tree Noise Injection

Given $\boldsymbol{x}_{\text{gt}} \sim q(\boldsymbol{x}_{\text{gt}})$ **and** $\boldsymbol{x}_{\text{tree}} \sim q(\boldsymbol{x}_{\text{tree}})$
$N_{\text{tree}} \sim \mathcal{U}_{\mathbb{Z}}(N_{\text{tree}}^{\min}, N_{\text{tree}}^{\max})$
for $k = 1 \dots N_{\text{tree}}$ **do**
  $d \sim \mathcal{U}(1, 360)$ ▷ Tree rotation degree.
  $s_{\text{xy}} \sim \mathcal{U}(s_{\text{xy}}^{\min}, s_{\text{xy}}^{\max})$ ▷ Tree coverage area scaler.
  $s_{\text{z}} \sim \mathcal{U}(s_{\text{z}}^{\min}, s_{\text{z}}^{\max})$ ▷ Tree height scaler.
  $\boldsymbol{x}_{\text{tree}} = \text{ROTATE}(\boldsymbol{x}_{\text{tree}}, d)$
  $\boldsymbol{x}_{\text{tree}} = \text{RESIZE}(\boldsymbol{x}_{\text{tree}}, s_{\text{xy}})$
  $\boldsymbol{x}_{\text{tree}} = s_{\text{z}} \boldsymbol{x}_{\text{tree}}$
  repeat
    $(i, j) \sim \{(i, j) \mid \boldsymbol{m}(i, j) = 0\}$ ▷ Sample coordinates outside of footprint.
    set $\boldsymbol{x}'_{\text{tree}}$ **to** $\text{ZEROS}(H, W)$
    merge $\boldsymbol{x}_{\text{tree}}$ **into** $\boldsymbol{x}'_{\text{tree}}$ **at center** (i, j)
    $\boldsymbol{x}, c_{\text{replaced}} = \text{MAX}(\boldsymbol{x}_{\text{gt}}, \boldsymbol{x}'_{\text{tree}})$ ▷ Get occluded height map and pixel count.
  until $c_{\text{replaced}} > 0$ ▷ Enforce tree occlusion.
end for

and G denotes the number of Gaussian distributions utilized in the construction of the GMM. Furthermore, $\mu_{g,x}$ and $\mu_{g,y}$ represent the mean in x and y direction, respectively, for g-th Gaussian distribution in the model. Similarly, $\sigma_{g,x}$ and $\sigma_{g,y}$ represent the standard deviation in x and y direction, respectively, for g-th Gaussian distribution. Also, $\sigma_{\min}$, $\sigma_{\max}$ are hyper-parameters. H and W are the height and width of the image. Note that a greater value of G can yield a more intricate GMM, which, in turn, can synthesize a more complex shape of incompleteness.

During training, there is no requirement to generate an incomplete mask with a precise number of missing points. Therefore, for more efficient mask generation within the dataloader, we propose a method outlined in Algorithm 1.

For benchmarking, we can sample an incompleteness mask, $\boldsymbol{m}_{\text{incomplete}}$, with a specific number of pixels to be removed, N_{thres}, using Algorithm 2.

2.3 Synthesizing Tree Noise

Figure 2 shows examples of tree height maps in our dataset. Algorithm 3 outlines injecting tree noise into ground truth height maps, $\boldsymbol{x}_{\text{gt}}$, to generate a corrupted height map, $\boldsymbol{x}$.

Dataset	Roof Variety	Point Cloud	Mesh Property	Roof Only	#Roof (k)
RoofGAN [21]	Limit	-	Compact (Noise Free)	✓	0.5
Ren *et al.* [22]	Limit	-	Compact (Noise Free)	✓	3
RoofN3D [35]	Limit	Real-world Scan	Compact (Noise Free)	✓	118
UrbanScene3D [17]	High	Real-world Scan	Dense (Reconstructed)		-
STPLS3D [3]	High	Real-world Scan	Dense (Reconstructed)		-
City3D [13]	High	Real-world Scan	Coarse (Reconstructed)		20
Building3D [33]	High	Real-world Scan	Coarse (Reconstructed)	✓	160
BuildingNet [26]	High	Sample from Mesh	Compact (Noise Free)		2
PoznanRD (Ours)	High	Sample from Mesh	Compact (Noise Free)	✓	13

Table 1: Comparison of Building Datasets.

2.4 Comparison to Existing Datasets

Table 1 provides the comparison between our PoznanRD and the existing building or roof datasets. RoofGAN [21], RoofN3D [35], and the dataset [22] provide compact, noise-free roof meshes but offer only a limited range of roof types. Specifically, RoofGAN [21] encompasses 0.5k roofs, constructed with 2 to 5 hip roof primitives. The dataset [22] presents 3k more complex meshes compared to RoofGAN [21], yet it primarily features buildings constructed with hip-based primitives. Moreover, most roofs in [22] lack detailed structures like dormers, which are central to our focus. RoofN3D [35] is confined to pyramid, saddleback, and two-sided hip roofs.

UrbanScene3D [17] and STPLS3D [3] offer a diverse range of roof types but provide ground truth meshes with scan noise and dense triangles, reconstructed from large-coverage real-world scanning. City3D [13] and Building3D [33] provide datasets featuring coarser ground truths, utilizing plane partition-based reconstruction algorithms and artist-assisted refinement methods, respectively. These datasets can contribute to research in compact mesh reconstruction. However, the meshes, are reconstructed from real-world scans and may contain scan noise and algorithmic errors, which do not align with our requirement for clean, error-free data.

BuildingNet [26] provides a wide variety of compact meshes but of only 2k buildings. Furthermore, direct usage is often impractical due to the inclusion of non-relevant elements like humans, virtual ground, trees, cars, and landscaping. Although BuildingNet [26] provides the classifying labels, we have encountered several misclassifications that can adversely affect the accuracy of roof-only ground truth extraction.

3 Experiments

3.1 Implementation Details

Network Architecture. We followed Palette [25] by employing a U-Net [24] architecture with an attention mechanism [32] in its deeper layers to construct our conditional diffusion models. Specifically, the input size is $2 \times 128 \times 128$, one channel for the corrupted height map, $\boldsymbol{x}$, and another for the estimated height map at step t, $\hat{\boldsymbol{x}}_t$. The output is the predicted noise in $\hat{\boldsymbol{x}}_t$ and is of size $1 \times 128 \times 128$. Our network contains four down-sampling modules and four up-sampling modules. These modules operate at resolutions of 128×128, 64×64, 32×32, and 16×16, with channel dimensions set to 64, 128, 256, and 512, respectively. Each module includes two residual blocks [7, 11] and the attention mechanism [32] is integrated into the modules when the resolution reaches 32×32 and 16×16.

Data Augmentation. We augment each height map by rotating 90, 180, and 270 degrees. Outlier noise occurs with a probability of 0.01%. Global noise is synthesized by sampling σ_{global} from a uniform distribution $\mathcal{U}(0, 0.05)$. For each pixel, a Gaussian distribution is constructed with the height as the mean and σ_{global} as the variance; subsequently, the height value is re-sampled using this distribution. We also augment the data with varying sparsity including 99, 98, 90, 80, 50, and 25%.

For synthesizing incompleteness, we follow Sec. 2.2 and use Algorithm 1 with five Gaussian distributions, $G = 5$, to synthesize various types of incompleteness. For each Gauss distribution, the minimum variance, σ_{min}, and maximum variance, σ_{max}, are set to 0 and 0.3, respectively.

The tree noise is injected according to Algorithm 3 using a 30% probability. We set the minimum and maximum tree count, $N_{\text{tree}}^{\text{min}}$ and $N_{\text{tree}}^{\text{max}}$, to 1 and 3, respectively. Furthermore, the tree coverage scaling parameters, $s_{\text{xy}}^{\text{min}}$ and $s_{\text{xy}}^{\text{max}}$, are set to 0.5 and 2.0. Lastly, $s_{\text{z}}^{\text{min}}$ and $s_{\text{z}}^{\text{max}}$ are set to 2.0 and 4.0 for height scaling.

Training is conducted on 8 NVIDIA Ampere A100 GPUs with a batch size of 512 for 260 epochs. This took approximately 2.5 days. The settings for variance scheduling and the use of exponential moving averages are adopted from [25]. The learning rate is set at 7×10^{-5}, and a warm-up learning rate is employed for the first 10k steps, starting with a factor of 0.2.

Training No-FP RoofDiffusion. During data augmentation, we retain the entire tree shape and do not remove tree noise outside the footprint. While RoofDiffusion uses a footprint image, $\boldsymbol{m}$, that is 0 for inside pixels and 1 outside, $\boldsymbol{m}$ is an array of 1s when training No-FP RoofDiffusion. This guarantees that the model possesses no prior footprint information.

Methods	Tree Counts			
	0	1	3	5
w/o tree aug.	0.203	0.363	0.523	0.768
w/ tree aug.	0.208	**0.278**	**0.356**	**0.504**

Table 2: Evaluation of tree noise resilience in training with tree augmentation (RMSE in meters).

Test Set	Steps					
	60	125	250	500	1000	2000
Easy (s95 i30)	0.444	**0.343**	0.339	0.337	0.338	0.344
Hard (s99 i80)	1.123	0.940	0.919	**0.893**	0.899	0.922

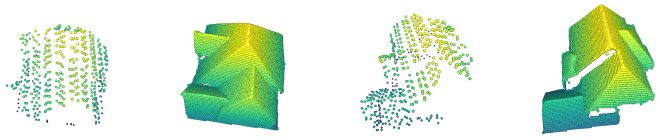
Table 3: Impact of inference steps on height completion quality (RMSE in meters). **Bold** denotes the most efficient in terms of achieving acceptable quality with the fewest steps.

3.2 Ablation of Tree Noise Augmentation

We investigated the impact of tree augmentation during model training. We constructed multiple test sets featuring varying numbers of tree intrusions per building. Specifically, tree count specifies the number of trees that will 100% appear in each building. Table 2 compares height map restoration quality for models trained with and without tree augmentation. The results clearly indicate that tree augmentation during training effectively enhances the resilience of the model to tree noise.

3.3 Sampling Step Analysis

We analyze the relationship between height completion quality and the number of inference steps. The model was trained using a fixed 2k steps, and we tested it by changing the number of inference steps, as detailed in Tab. 3. For easier datasets with 95% sparsity and 30% incompleteness, reducing the inference steps to 250 resulted in only a minor loss of quality. However, for harder datasets, 500 inference steps were needed for satisfactory results. We observed that for more difficult tasks more steps are needed. Also, we attempted to train the model directly with fewer steps such as 1k. This led to failure to converge.



(a) Input 1 **(b)** Predicted 1 **(c)** Input 2 **(d)** Predicted 2

Fig. 3: RoofDiffusion reconstruction of height maps corrupted by scan line strip patterns.

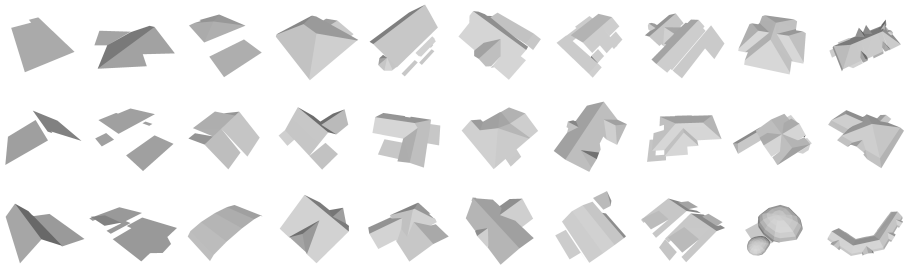


Fig. 4: Samples in the PoznanRD (Poznan Roof Dataset).

3.4 LiDAR Scan Patterns

Our observations indicate that simulating sparse LiDAR points through random point removal yields results without noticeable gaps, closely resembling real-world scans. We found that most real-world point clouds [13, 28, 29, 31] appear similar to those obtained via random sampling. While a few examples exhibit line strip patterns, these do not hinder the ability of RoofDiffusion to restore roofs. Figure 3 showcases the corrupted height maps featuring line strip patterns, and RoofDiffusion effectively reconstructs complete, noise-free height maps from these.

3.5 Roof Variety

Figure 4 illustrates the variety of roof types in the PoznanRD dataset. Constructions of valid roofs of different types and with a variety of features have been shown for RoofDiffusion and No-FP RoofDiffusion: gables (Fig. 10e), hips (Figs. 8a and 8h), shads (Figs. 8e and 10d), leans (Figs. 7b and 7i), and flats (Fig. 10a).

3.6 How Prone is the Model to Hallucination based on Footprints?

With only the footprint available, and no height data provided, we conducted tests using a height map set to all zeros, and varied the footprints. The output was predominantly zeros or values less than one meter. This suggests that the model uses footprints to enhance height prediction but does not by itself cause the diffusion model to hallucinate roof shapes.

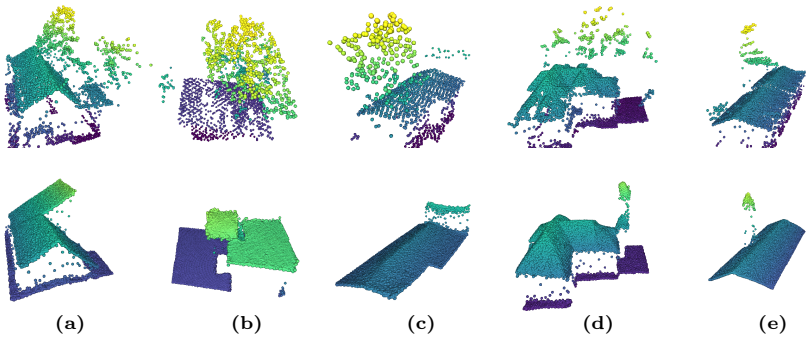


Fig. 5: Cases of tree noise misinterpreted as roof structures.

Methods	iRMSE	iMAE	RMSE	MAE
SparseConvs [30]	4.94	1.78	1601.33	481.27
IP_Basic [16]	3.78	1.29	1288.46	302.60
ADNN [4]	59.39	3.19	1325.37	439.48
Nconv_CNN [9]	4.67	1.52	1268.22	360.28
S2D (depth-only) [18]	3.21	1.35	954.36	288.64
HMS-Net [14]	2.93	1.14	937.27	258.48
pNCNN [8]	3.37	1.05	960.05	251.77
Physical_Surface [36]	3.76	1.21	1239.84	298.30
DTP [37]	2.94	1.07	937.27	247.81
CU-Net [34]	**2.69**	**1.04**	**917.76**	**244.36**
Ours (KITTI)	5.69	2.66	1641.41	491.61
Ours + M.O. (KITTI)	4.84	1.84	1631.98	448.97

Table 4: Evaluation of depth completion methods on KITTI dataset [30]. M.O. stands for applying mean shift offset.

3.7 Failure Cases

Figure 5 provides some examples of failure cases where RoofDiffusion misclassifies tree noise as roof structures. Figures 5a and 5b illustrate erroneous reconstructions of unlikely slope and flat planes, respectively, caused by extensive occlusion from tree points. Figure 5c incorrectly identifies tree points as thin wall structures. Figures 5d and 5e mistakenly interpret tree noise as non-existent chimney structures.

3.8 Depth Completion on the KITTI Dataset

We extend RoofDiffusion to address the unguided depth completion task on the KITTI dataset [30]. Given the original KITTI depth map resolution of 352×1216, training with such size is impractical due to the excessive processing time required by the diffusion model. Therefore, we downsampled the depth maps to

a resolution of 176×608 and trained our model using randomly cropped sections of size 176×304. We trained on the official training split with 86k samples. We conducted tests on the 1k validation depth maps [30], where the predicted results with the size of 176×608 are upsampled back to 352×1216 for evaluation.

We compared to the results of state-of-the-art and representative unguided depth completion methods, reported in [34]. Table 4 combines these results with those newly reported by our method. We observed a mean shift phenomenon in our method. To address this, we aligned the predicted depth map with the input sparse depth map by offsetting it with the difference between their respective means. This adjustment resulted in a significant performance boost.

Nevertheless, our results do not improve on prior methods, likely due to the differences in KITTI depth completion v.s. roof completion. Available pixels in the KITTI dataset [30] are more uniformly distributed, and regional incompleteness size is smaller compared to roof completion tasks. Consequently, KITTI depth completion resembles an interpolation task more than inpainting for roof completion.

From this we conclude: diffusion models are more adept at tasks requiring the inpainting of large missing regions, while for smaller interpolation tasks, their performance is on par with other methods. Our conclusion is supported by two observations. First, in our PoznanRD benchmark (see Table 2 in the main paper), the performance margin of RoofDiffusion over existing methods [1, 15, 27] is greater in tasks of restoring incompleteness compared to sparsity. Secondly, SpAgNet [5], a diffusion model-based depth completion method, does not outperform existing methods [8,10,12,19] when completing sparsely scanned data with 64 lines. However, as the scan lines become sparser, 32, 16, 8, and 4 lines, SpAgNet [5] outperforms others.

3.9 Additional Qualitative Experimental Results

RoofDiffusion.

- Figure 6 showcases examples of applying various height map pre-processors for 3D reconstruction algorithm, City3D [13], on our PoznanRD.
- Figure 7 illustrates the evaluation on AHN3 dataset [13].
- Figure 8 shows the evaluation on Dales3D dataset [31].
- Figure 9 provides the results on USGS 3DEP LiDAR sampled over Wayne County, MI [28].
- Figure 10 provides the results on USGS 3DEP LiDAR sampled over Cambridge, MA [29].

No-FP RoofDiffusion.[3]

- Figure 11 provides the results on USGS 3DEP LiDAR sampled over Wayne County, MI [28].
- Figure 12 provides the results on USGS 3DEP LiDAR sampled over Cambridge, MA [29].

[3] Since the point clouds in AHN3 [13] and Dales3D [31] are already cropped by footprints, we do not include these two datasets in our testing.

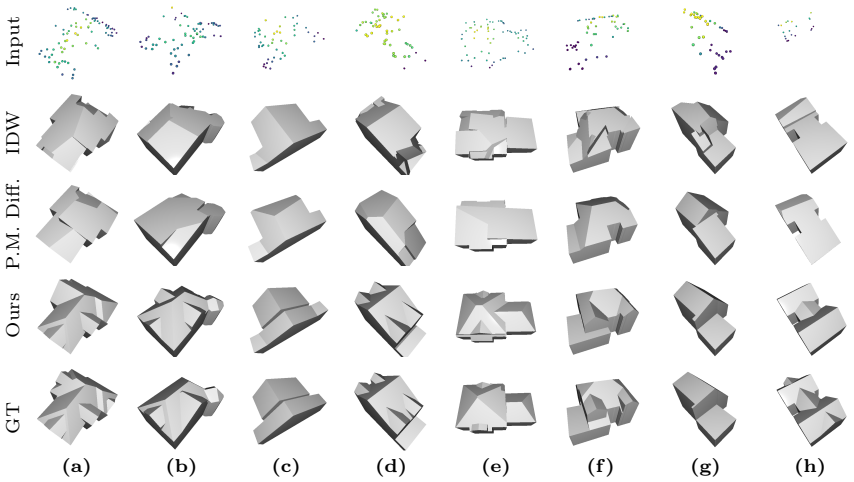


Fig. 6: 3D reconstruction using different height map pre-processors. RoofDiffusion closely matches the ground truth (GT).

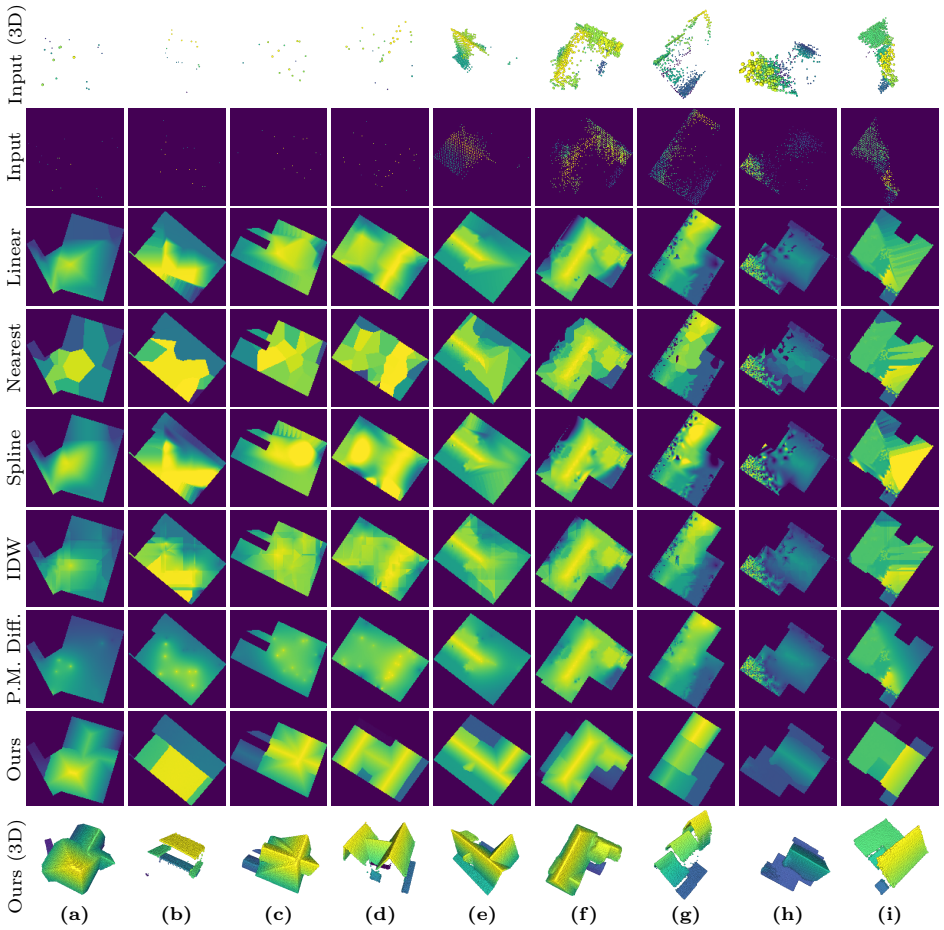


Fig. 7: Evaluation of completion and denoising on the AHN3 dataset [13]. Compared with Linear, Nearest, Spline [15], Inverse Distance Weighting (IDW) [27], and Perona-Malik Diffusion (P.M. Diff.) [1, 20] interpolation methods.

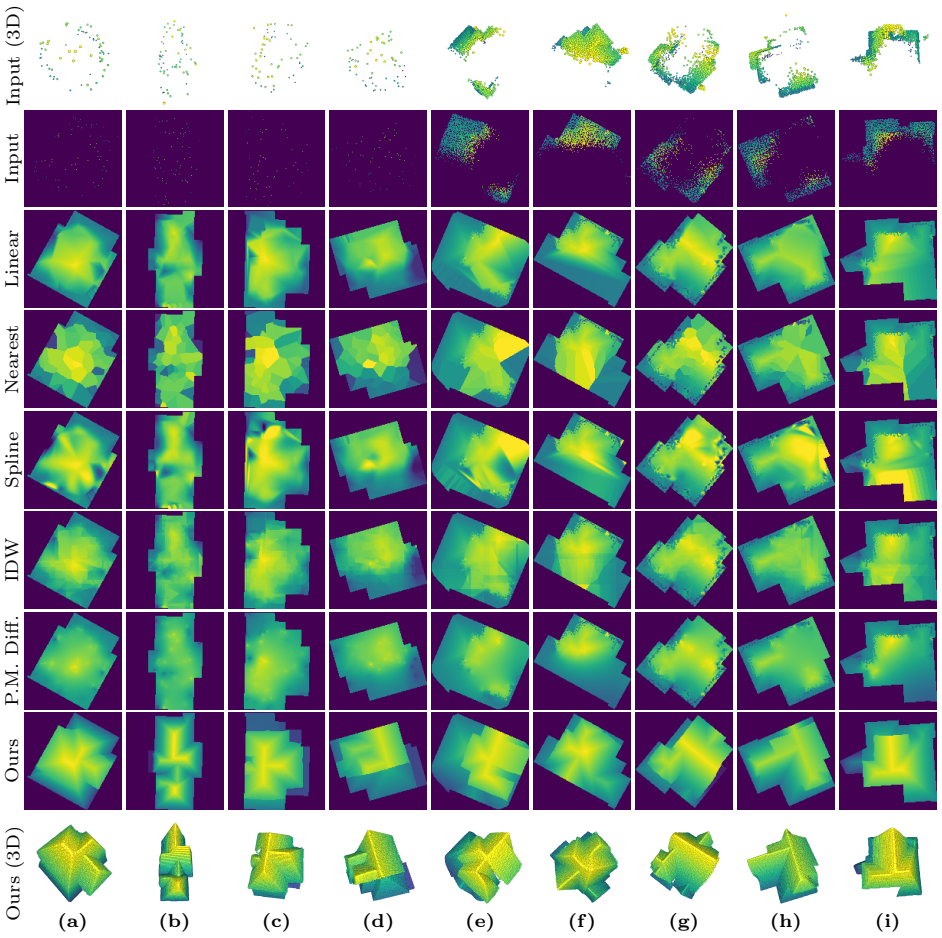


Fig. 8: Evaluation of the completion and denoising on the Dales3D dataset [31].

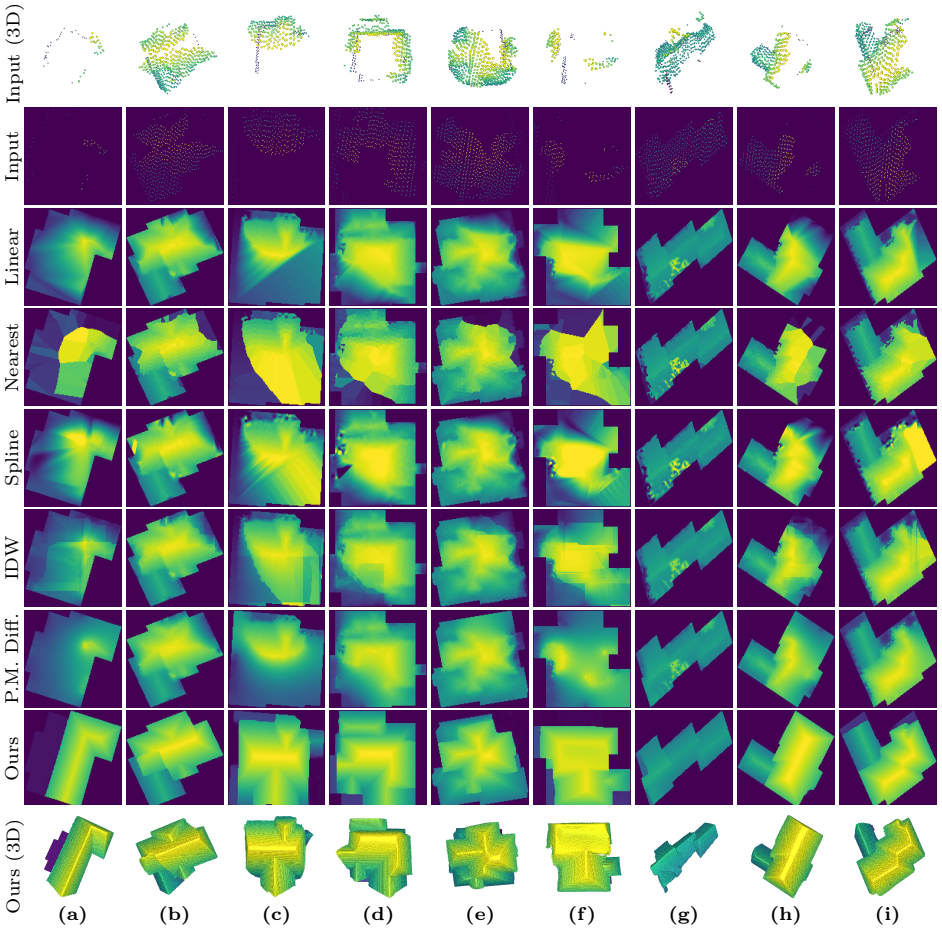


Fig. 9: Evaluation of the completion and denoising on USGS 3DEP LiDAR data sampled over Wayne County, MI [28].

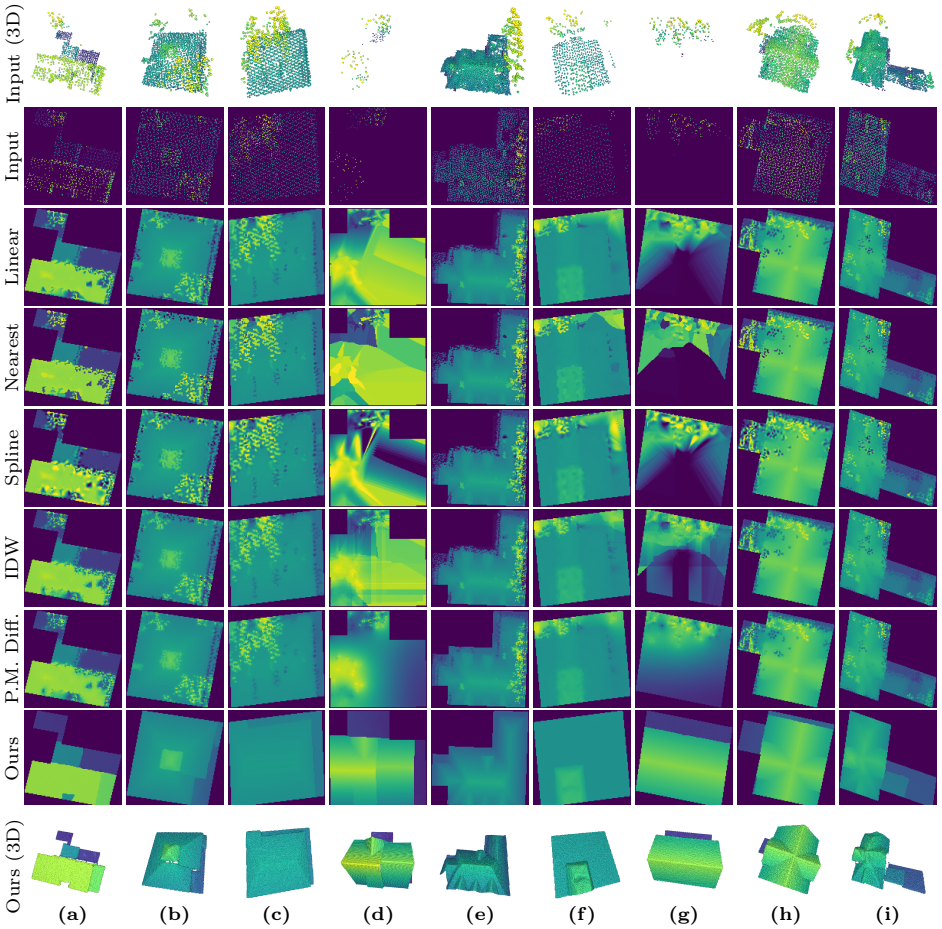


Fig. 10: Evaluation of the completion and denoising on USGS 3DEP LiDAR data sampled over Cambridge, MA [29].

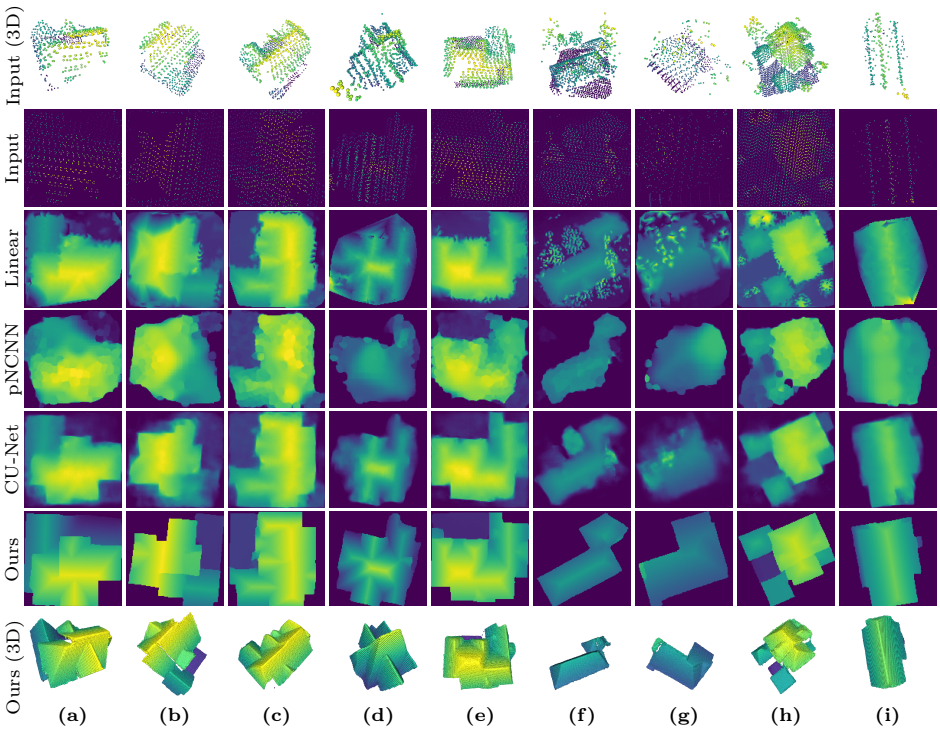


Fig. 11: Evaluation of the completion and denoising for No-FP RoofDiffusion on USGS 3DEP LiDAR data sampled over Wayne County, MI [28]. Comparison with linear, pNCNN [8], and CU-Net [34].

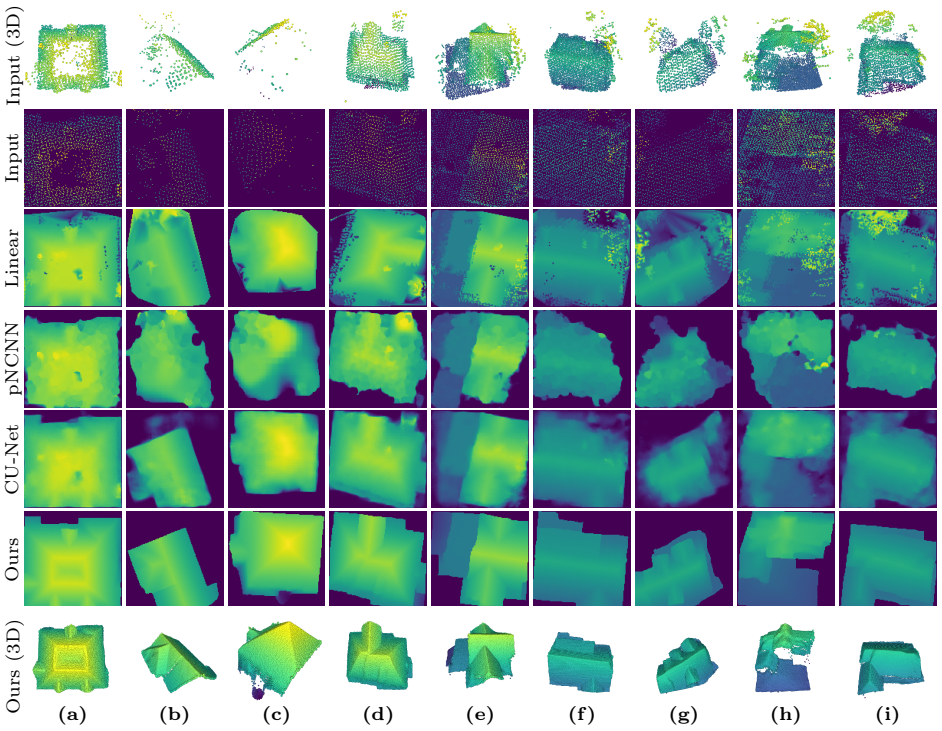


Fig. 12: Evaluation of the completion and denoising for No-FP RoofDiffusionon USGS 3DEP LiDAR data sampled over Cambridge, MA [29].

References

1. Biasutti, P., Aujol, J.F., Brédif, M., Bugeau, A.: Diffusion and inpainting of reflectance and height LiDAR orthoimages. Computer Vision and Image Understanding **179**, 31–40 (2019) 10, 12
2. Biljecki, F., Ledoux, H., Stoter, J.: An improved LOD specification for 3D building models. Computers, Environment and Urban Systems **59**, 25–37 (2016) 2
3. Chen, M., Hu, Q., Yu, Z., Thomas, H., Feng, A., Hou, Y., McCullough, K., Ren, F., Soibelman, L.: STPLS3D: A large-scale synthetic and real aerial photogrammetry 3d point cloud dataset. In: Proceedings of the British Machine Vision Conference (2022) 5
4. Chodosh, N., Wang, C., Lucey, S.: Deep convolutional compressed sensing for LiDAR depth completion. In: Proceedings of the Asian Conference on Computer Vision. pp. 499–513. Springer (2019) 9
5. Conti, A., Poggi, M., Mattoccia, S.: Sparsity agnostic depth completion. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 5871–5880 (2023) 10
6. contributors, S.C.P.: Three-dimensional model of Poznan. http://sip.poznan.pl/model3d/ (2023), accessed: July 1, 2023 2
7. Dhariwal, P., Nichol, A.: Diffusion models beat GANs on image synthesis. Advances in neural information processing systems **34**, 8780–8794 (2021) 6
8. Eldesokey, A., Felsberg, M., Holmquist, K., Persson, M.: Uncertainty-aware CNNs for depth completion: Uncertainty from beginning to end. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12014–12023 (2020) 9, 10, 16
9. Eldesokey, A., Felsberg, M., Khan, F.S.: Propagating confidences through CNNs for sparse data regression. In: Proceedings of the British Machine Vision Conference (2018) 9
10. Guizilini, V., Ambrus, R., Burgard, W., Gaidon, A.: Sparse auxiliary networks for unified monocular depth prediction and completion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11078–11088 (2021) 10
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016) 6
12. Hu, M., Wang, S., Li, B., Ning, S., Fan, L., Gong, X.: PENet: Towards precise and efficient image guided depth completion. In: Proceedings of the IEEE International Conference on Robotics and Automation. pp. 13656–13662. IEEE (2021) 10
13. Huang, J., Stoter, J., Peters, R., Nan, L.: City3D: Large-scale building reconstruction from airborne LiDAR point clouds. Remote Sensing **14**(9), 2254 (2022) 5, 8, 10, 12
14. Huang, Z., Fan, J., Cheng, S., Yi, S., Wang, X., Li, H.: HMS-Net: Hierarchical multi-scale sparsity-invariant network for sparse depth completion. IEEE Transactions on Image Processing **29**, 3429–3441 (2019) 9
15. Keys, R.: Cubic convolution interpolation for digital image processing. IEEE transactions on acoustics, speech, and signal processing **29**(6), 1153–1160 (1981) 10, 12
16. Ku, J., Harakeh, A., Waslander, S.L.: In defense of classical image processing: Fast depth completion on the cpu. In: 2018 15th Conference on Computer and Robot Vision. pp. 16–22. IEEE (2018) 9

17. Lin, L., Liu, Y., Hu, Y., Yan, X., Xie, K., Huang, H.: Capturing, reconstructing, and simulating: the UrbanScene3D dataset. In: European Conference on Computer Vision. pp. 93–109. Springer (2022) 5
18. Ma, F., Cavalheiro, G.V., Karaman, S.: Self-supervised sparse-to-dense: Self-supervised depth completion from LiDAR and monocular camera. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 3288–3295. IEEE (2019) 9
19. Park, J., Joo, K., Hu, Z., Liu, C.K., So Kweon, I.: Non-local spatial propagation network for depth completion. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16. pp. 120–136. Springer (2020) 10
20. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. IEEE Transactions on pattern analysis and machine intelligence **12**(7), 629–639 (1990) 12
21. Qian, Y., Zhang, H., Furukawa, Y.: Roof-GAN: Learning to generate roof geometry and relations for residential houses. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2796–2805 (2021) 5
22. Ren, J., Zhang, B., Wu, B., Huang, J., Fan, L., Ovsjanikov, M., Wonka, P.: Intuitive and efficient roof modeling for reconstruction and synthesis. ACM Transactions on Graphics **40** (2021) 5
23. Reynolds, D.A., et al.: Gaussian mixture models. Encyclopedia of biometrics **741**(659-663) (2009) 3
24. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. pp. 234–241. Springer (2015) 6
25. Saharia, C., Chan, W., Chang, H., Lee, C., Ho, J., Salimans, T., Fleet, D., Norouzi, M.: Palette: Image-to-image diffusion models. In: ACM SIGGRAPH 2022 Conference Proceedings. pp. 1–10 (2022) 6
26. Selvaraju, P., Nabail, M., Loizou, M., Maslioukova, M., Averkiou, M., Andreou, A., Chaudhuri, S., Kalogerakis, E.: BuildingNet: Learning to label 3D buildings. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10397–10407 (2021) 5
27. Shepard, D.: A two-dimensional interpolation function for irregularly-spaced data. In: Proceedings of the 1968 23rd ACM national conference. pp. 517–524 (1968) 10, 12
28. Survey, U.G.: MI Wayne County 2017 A17 345307 LAS. https://rockyweb.usgs.gov/vdelivery/Datasets/Staged/Elevation/LPC/Projects/MI_WayneCounty_2017_A17/ (2017), accessed: July. 1, 2023 8, 10, 14, 16
29. Survey, U.G.: MA CentralEastern 2021 B21 19TCG324693 LAS. https://rockyweb.usgs.gov/vdelivery/Datasets/Staged/Elevation/LPC/Projects/MA_CentralEastern_2021_B21/ (2019), accessed: July. 1, 2023 8, 10, 15, 17
30. Uhrig, J., Schneider, N., Schneider, L., Franke, U., Brox, T., Geiger, A.: Sparsity invariant CNNs. In: 2017 international conference on 3D Vision (3DV). pp. 11–20. IEEE (2017) 9, 10
31. Varney, N., Asari, V.K., Graehling, Q.: DALES: A large-scale aerial LiDAR data set for semantic segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 186–187 (2020) 8, 10, 13
32. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017) 6

33. Wang, R., Huang, S., Yang, H.: Building3D: A urban-scale dataset and benchmarks for learning roof structures from point clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 20076–20086 (2023) 5
34. Wang, Y., Dai, Y., Liu, Q., Yang, P., Sun, J., Li, B.: CU-Net: LiDAR depth-only completion with coupled U-Net. IEEE Robotics and Automation Letters **7**(4), 11476–11483 (2022) 9, 10, 16
35. Wichmann, A., Agoub, A., Kada, M.: ROOFN3D: Deep learning training data for 3D building reconstruction. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences **42**, 1191–1198 (2018) 5
36. Zhao, Y., Bai, L., Zhang, Z., Huang, X.: A surface geometry model for lidar depth completion. IEEE Robotics and Automation Letters **6**(3), 4457–4464 (2021) 9
37. Zhao, Y., Elhousni, M., Zhang, Z., Huang, X.: Distance transform pooling neural network for lidar depth completion. IEEE Transactions on Neural Networks and Learning Systems (2021) 9