

P2P-Bridge: Supplementary Materials

Mathias Vogel¹, Keisuke Tateno², Marc Pollefeys^{1,3}, Federico Tombari^{2,4},
Marie-Julie Rakotosaona^{*,2}, and Francis Engelmann^{*,1,2}

¹ ETH Zurich ² Google ³ Microsoft ⁴ TU Munich

1 Experiment Details

We present additional results, including experiment details and information about our backbone network. Additionally, we include extra experiments, such as object-level denoising using non-Gaussian noise types. In this section, we will first provide additional details about the data processing and training of our method and baseline methods.

1.1 Frameworks and Computational Resources

We implement our method using Python 3.10, PyTorch 2.0, and CUDA 11.8. All experiments are conducted using a single RTX 3090 or RTX 4090, except when training the large models on ARKitScenes and ScanNet++, for which 4 RTX 3090 are used.

1.2 Dataset Processing and Splitting

In our main paper, we only utilized a subset of the testing rooms due to computational limitations. The subsets were filtered to ensure representation of different room types (e.g., kitchen, bathroom, living room) with minimal errors or outliers in the point clouds. Subsequently, a random subset from the filtered rooms was chosen for testing all algorithms. You can access the training, validation, and testing splits for indoor scenes in our [code repository](#).

ARKitScenes. We filter out rooms that do not have high-resolution RGB frames or have incorrect laser scanner positions. Since the Faro scan is not in the same coordinate system as the low-resolution scans, we first use PREDATOR [3] to align the Faro scan with the low-resolution scans. Then, we use the multi-scale iterative closest point (ICP) algorithm with the transformation matrices from PREDATOR as initialization to refine the transformation matrix. After aligning all rooms with the refined transformation, we crop outliers caused by reflection errors by calculating a minimal bounding box around each room. For the training phase, we incorporate 96 visits (rooms) with 266 noisy reconstructions, as there are often multiple reconstructions per room created at different times and lighting conditions. To evaluate against the baselines, we use 10 rooms with a single reconstruction each.

ScanNet++. On ScanNet++, we calculate the noisy point cloud reconstructions using scripts provided by the authors. These reconstructions were obtained by first filtering the depth maps according to their agreement with the Faro laser depths. This was followed by 3D projection using the camera intrinsic matrices and globally optimized poses from COLMAP [11, 12] without applying further fusion methods. We also generate 3D reconstructions using 3DMatch [16] on the pre-filtered depth maps with globally optimized poses. This process provides additional evaluation data to assess the methods in a more advanced pipeline, which includes integrating deep-learning methods in the 3D reconstruction step. We have utilized the dataset version from November 2023. For training, we are using the `nvs_sem_train` dataset, while the `nvs_sem_val` dataset is split into two parts - one for ablation studies and another for comparing methods and generating tables in the main paper. We have opted to use `nvs_sem_val` for evaluation/testing due to its minimal errors in ground truth room scans. The training set comprises 230 rooms, and we are conducting evaluations against baseline methods on 8 rooms.

Object-Datasets. We use the data provided by ScoreDenoise in the form of a [Google Drive link](#) to obtain the PU-Net training data, as well as the test data for Gaussian noise (`examples` folder) and the PC-Net test data.

1.3 Patch-Wise Sampling

On the PU-Net dataset, we train our model on only coordinate features and KNN patches of size 2048. When it comes to indoor scene data, we sample training batches of size 4096 from spherical patches with a radius of 0.3 m on ScanNet++ and 0.5 m on ARKitScenes. The radius size was empirically tuned to approximately fit the target amount of points. When there are less than 4096 noisy points, we up-sample the size by duplicating points and adding 2% of bounding diagonal Gaussian noise to both coordinates and features of noisy points. If there are more than 4096 noisy points inside the sphere, we use farthest point sampling to down-sample the input. When there are less than 4096 clean points in the sampling sphere, we don't use it to avoid the model being trained on assigning multiple noisy points to the same clean points. We center each sphere and scale it to the unit radius. We generate overlapping patches during inference and average all point predictions for points in the overlapping regions. When a patch is artificially up-sampled, we only take predictions for the original points and ignore the predictions for the supplementary noisy points because including those leads to irregular point density.

1.4 Feature Extraction

We require a point cloud of a whole scene and additional RGB images, camera intrinsics, and poses to extract additional features. We use DINOv2-VITS14 to extract features from RGB frames of the scene of shape $\lfloor \frac{H}{14} \rfloor \times \lfloor \frac{W}{14} \rfloor, 384$ which we then up-sample to the original image size $[H, W, 384]$ using bilinear

interpolation. We use each feature map’s corresponding pose and camera intrinsics to project the full point cloud back onto the image grid, respecting occlusion and extracting the feature vector at the grid coordinate. Finally, we perform KNN interpolation on the full cloud to fill in missing features due to occlusion.

1.5 Data-Alignment

We use two approaches depending on the datasets used to align the clean data with the noisy data for nearest-neighbor interpolation. On the PU-Net dataset, we use an implementation based on the auction algorithm to align the noisy and clean data because the noisy and clean point clouds have the same amount of points. We use the CUDA-based implementation from PointMixup [1]. On ScanNet++ and ARKitScenes, we use the fact that the clean point clouds have a much higher amount of points than the noisy clouds. We find that directly assigning the nearest neighbor in the clean point cloud for every noisy point leads to very good results and assigns unique neighbors for nearly 100% of noisy points. This mapping describes the shortest path between the noisy points and the surface of the clean point cloud.

1.6 Baseline Methods

Our experiments on indoor scenes use the parameters and pre-trained weights provided in the publicly available code repositories. These are, [ScoreDenoise](#), [MAG](#), [DMRDenoise](#), [Itertive-PFN](#) and [PD-Flow](#). We utilized the best-performing hyperparameters as per the respective papers for each model. Pre-trained weights were used for initialization when training on ScanNet++ and ARKitScenes. The code for the bilateral point cloud filter can be found on their [project page](#).

1.7 Network Architecture

Figure 1 and Fig. 2 describe the individual components of our diffusion model backbone. It mainly consists of blocks introduced in PVCNN [5], except for the adaptive group normalization block we borrow from LION [17]. The grouping operation in the set-abstraction block first takes a k-nearest-neighborhood for each point in the input and then uses max-pooling on each neighborhood to extract the feature with the highest value.

1.8 Hyperparameters

We find that a network of smaller size is sufficient for datasets with a relatively small amount of noise or simpler underlying clean data, such as the PU-Net dataset or ScanNet++ scene reconstructions obtained using 3DMatch. We list the parameters for the building blocks and the diffusion parametrization in Tab. 3 and Tab. 4. The diffusion timesteps are quadratic discretized between t_0 and t_1 using T discretization steps. The linear-symmetric schedule is adapted from

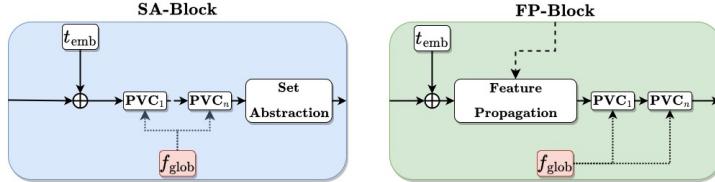


Fig. 1: Set abstraction and feature propagation block with additional point voxel convolution blocks.

I2SB [4] On ARKitScenes and ScanNet++ without 3DMatch, we use a larger architecture in terms of parameters, which is described in Tab. 2 and Tab. 1.

Table 1: Parameters of individual network blocks for the large architecture used on ARKitScenes and ScanNet++.

| Parameter | Setting | | | |
|---------------------------------|-----------|---------|---------|----------|
| | SA1 | SA2 | SA3 | SA4 |
| PVC Layers | 2 | 3 | 2 | - |
| PVC Dimension | 64 | 128 | 256 | - |
| Shared-MLP Output Dimension | 64-128 | 128-256 | 256-512 | 512-1024 |
| Grouper Points | 1024 | 256 | 64 | 16 |
| Grouper Radius | 0.1 | 0.2 | 0.4 | 0.8 |
| Voxel Resolution | 32 | 16 | 8 | 8 |
| Global Attention Head Dimension | 32 | | | |
| Global Attention Heads | 12 | | | |
| Parameter | FP1 | FP2 | FP3 | FP4 |
| | 2 | 2 | 3 | 2 |
| PVC Layers | 512 | 512 | 256 | 128 |
| PVC Dimension | 8 | 8 | 16 | 32 |
| Shared-MLP Output Dimension | 1024-512 | 512-512 | 512-256 | 256-128 |
| Shared-MLP Output Dimension | 128-128-3 | | | |

Table 2: Global network parameters for the large architecture used on ARKitScenes and ScanNet++.

| Parameter | Setting |
|-----------------------------------|------------------|
| PVC Dropout | 0.1 |
| Time Embedding Dimension | 64 |
| Global Feature Dimension | 1024 |
| Feature Embedding Dimension | 64 |
| Grouper Max Neighbors | 32 |
| Diffusion Timesteps T | 1000 |
| Diffusion t_0 | 0.0001 |
| Diffusion t_1 | 1 |
| Diffusion Schedule | linear-symmetric |
| Diffusion Schedule β_{\min} | 0.0001 |
| Diffusion Schedule β_{\max} | 0.0003 |

Table 3: Parameters of individual network blocks for the small architecture used on PU-Net and ScanNet++ & 3DMatch.

| Parameter | Setting | | | |
|---------------------------------|----------|---------|---------|---------|
| | SA1 | SA2 | SA3 | SA4 |
| PVC Layers | 1 | 2 | 1 | - |
| PVC Dimension | 32 | 64 | 128 | - |
| Shared-MLP Output Dimension | 32-64 | 64-128 | 128-256 | 256-512 |
| Grouper Points | 512 | 128 | 32 | 8 |
| Grouper Radius | 0.1 | 0.2 | 0.4 | 0.8 |
| Voxel Resolution | 32 | 16 | 8 | 8 |
| Global Attention Head Dimension | 32 | | | |
| Global Attention Heads | 4 | | | |
| Parameter | FP1 | FP2 | FP3 | FP4 |
| | 1 | 1 | 2 | 1 |
| PVC Layers | 256 | 256 | 128 | 64 |
| PVC Dimension | 8 | 8 | 16 | 32 |
| Shared-MLP Output Dimension | 512-256 | 256-256 | 256-128 | 128-64 |
| Shared-MLP Output Dimension | 64-128-3 | | | |

Table 4: Global network parameters for small the architecture used on PU-Net and ScanNet++ & 3DMatch.

| Parameter | Setting |
|-----------------------------------|------------------|
| PVC Dropout | 0.15 |
| Time Embedding Dimension | 64 |
| Global Feature Dimension | 1024 |
| Feature Embedding Dimension | 32 |
| Grouper Max Neighbors | 32 |
| Diffusion Timesteps T | 1000 |
| Diffusion t_0 | 0.0001 |
| Diffusion t_1 | 1 |
| Diffusion Schedule | linear-symmetric |
| Diffusion Schedule β_{\min} | 0.0001 |
| Diffusion Schedule β_{\max} | 0.02 |

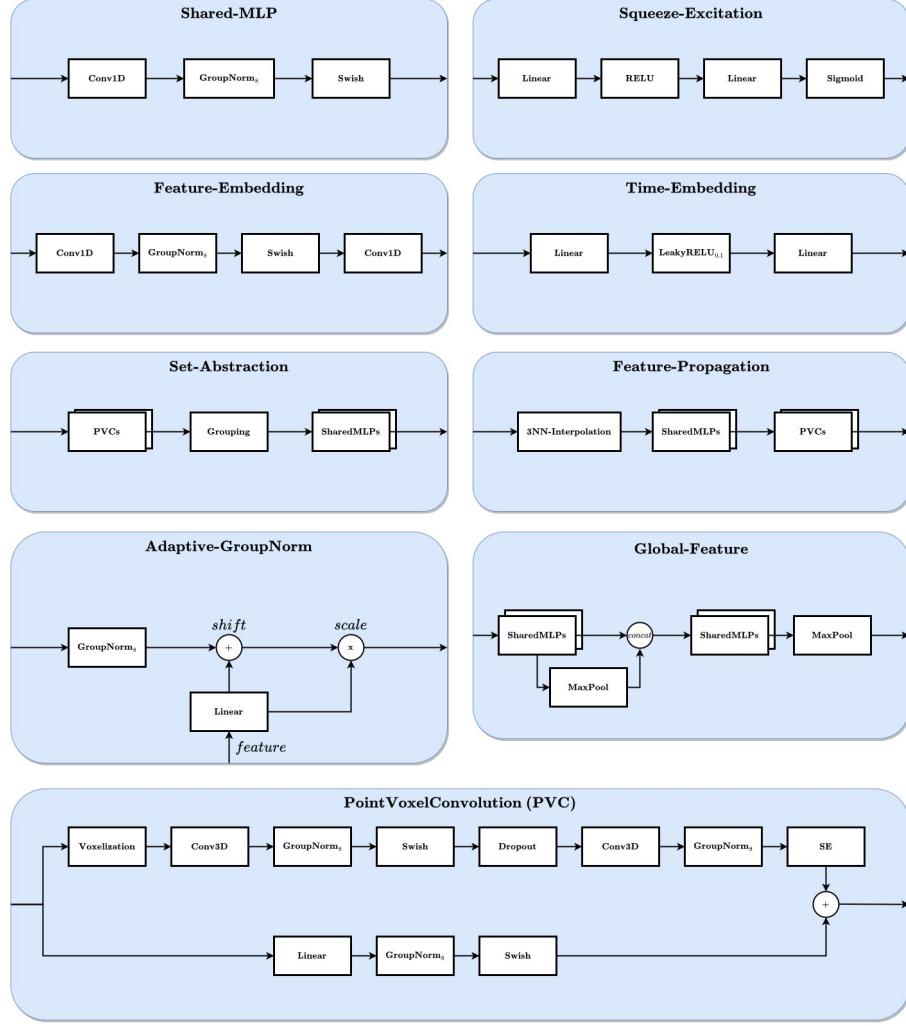


Fig. 2: Building blocks of our network architecture based on PVCNN [5].

2 Additional Experiments

This section provides additional quantitative and qualitative results on the denoising task but also discusses runtime as well as the effectiveness of using EMD assignment for shortest-path-interpolation.

2.1 Inference Speed

The comparison of runtime required to achieve the denoising results in our work is presented in the table below. Although diffusion models typically entail

high runtime due to the extensive inference steps involved, our approach only necessitates as few as three steps, resulting in a relatively low runtime compared to other methods such as I-PFN.

Table 5: Average Runtime. Comparison of the average denoising runtime of different denoising methods without I/O on a RTX 3090.

| Method | 10k (s) | 50k (s) | 1.5M (s) |
|-----------|---------|---------|----------|
| Bilateral | 0.26 | 2.70 | 295.42 |
| PD-Flow | 0.78 | 4.89 | 305.08 |
| ScoreDen. | 0.91 | 4.18 | 332.98 |
| I-PFN | 14.05 | 31.29 | 912.43 |
| Ours | 0.82 | 4.36 | 342.18 |

2.2 Data-To-Data vs Noise-To-Data

In this analysis, we are comparing our data-to-data denoising approach with noise-to-data denoising methods by tailoring generative point cloud diffusion models to the denoising task. A crucial distinction between these two types of generative models lies in the fact that noise-to-data models generate the noisy point cloud by applying Gaussian noise to each clean point from the original point cloud. This means that the correspondence between noisy and clean points is always known. In contrast, for real-world denoising, the noisy point cloud comes from a 3D reconstruction or scan affected by non-Gaussian noise, and there is no direct correspondence between noisy and clean points. Hence, learning the denoising process becomes complex. The effectiveness of diffusion bridges stems from their ability to integrate existing data by conditioning the diffusion process using prior distributions. To support this claim, we adapted PointVoxelDiffusion (PVD) [19] and PointDiffusionRefinement (PDR) [7] to the denoising task. For PVD, this is achieved using the same approach as the one employed by the authors of PVD for point cloud completion. This involves combining Gaussian noise with real-world noise prior by point cloud concatenation. In PDR, a distinct network is utilized to merge Gaussian noise with the real-world noise prior in a more advanced manner. The outcomes of these experiments are presented below, demonstrating a definite enhancement of our data-to-data approach over both noise-to-data approaches.

2.3 Shortest-path Interpolation

In our main paper, we address the challenge of linearly interpolating disordered data structures like point clouds by using shortest-path interpolation. This method establishes a unique correspondence between the noisy and the clean point clouds. We achieve this correspondence by employing the Hungarian algorithm or by

Table 6: Noise-to-Data vs Data-to-Data. Comparison of different diffusion approaches. The noise-to-data approaches (PVD & PDR) begin denoising from isotropic Gaussian noise, while our data-to-data approach starts denoising using the real-world noisy data directly. The metrics are calculated on the PU-Net test dataset [15] and are multiplied by 10^4 .

| Method | 10k | | | | | | 50k | | | | | |
|----------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| | 1% noise | | 2% noise | | 3% noise | | 1% noise | | 2% noise | | 3% noise | |
| | CD | P2M |
| PVD [19] | 25.9 | 22.12 | 27.11 | 23.13 | 29.53 | 25.25 | 5.50 | 4.74 | 7.57 | 6.64 | 10.0 | 8.88 |
| PDR [7] | 3.68 | 1.19 | 7.76 | 4.30 | 13.45 | 8.82 | 1.09 | 0.62 | 1.81 | 0.9 | 2.59 | 1.52 |
| Ours | 2.28 | 0.39 | 3.20 | 0.81 | 3.99 | 1.42 | 0.59 | 0.09 | 0.90 | 0.32 | 1.56 | 0.84 |

utilizing the assignments derived from EMD calculation. This approach proves to be an optimal solution to the Schroedinger’s Bridge Problem within our specific problem framework [9]. To validate this assumption, we retrained our network on object-level datasets, employing Chamfer-Distance based interpolation and present the results below. Our findings demonstrate that the EMD-based shortest path interpolation consistently outperforms the Chamfer-Distance based interpolation in all scenarios.

Table 7: Comparing Shortest-Path Interpolation Methods. Comparison of our EMD-based shortest path interpolation methods with a CD-based shortest path interpolation method. The metrics are calculated on the PU-Net dataset [15] under Gaussian noise.

| Shortest Path Method | 10k | | | | | | 50k | | | | | |
|-------------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| | 1% noise | | 2% noise | | 3% noise | | 1% noise | | 2% noise | | 3% noise | |
| | CD | P2M |
| CD-Based | 2.34 | 0.41 | 3.35 | 0.84 | 4.17 | 1.48 | 0.62 | 0.11 | 0.94 | 0.33 | 1.59 | 0.90 |
| EMD-Based | 2.28 | 0.39 | 3.20 | 0.81 | 3.99 | 1.42 | 0.59 | 0.09 | 0.90 | 0.32 | 1.56 | 0.84 |

2.4 Additional Qualitative Results on PC-Net

We show additional qualitative results in Fig. 3 on denoising chosen objects from the PC-Net test dataset under 3% of Gaussian noise.

2.5 Denoising on Non-Gaussian Artificial Noise

For additional experiments on the PU-Net and PC-Net test data, we employ various types of synthetic noise. To better grasp the impact of each noise type on clean data, we provide an overview in Fig. 4.

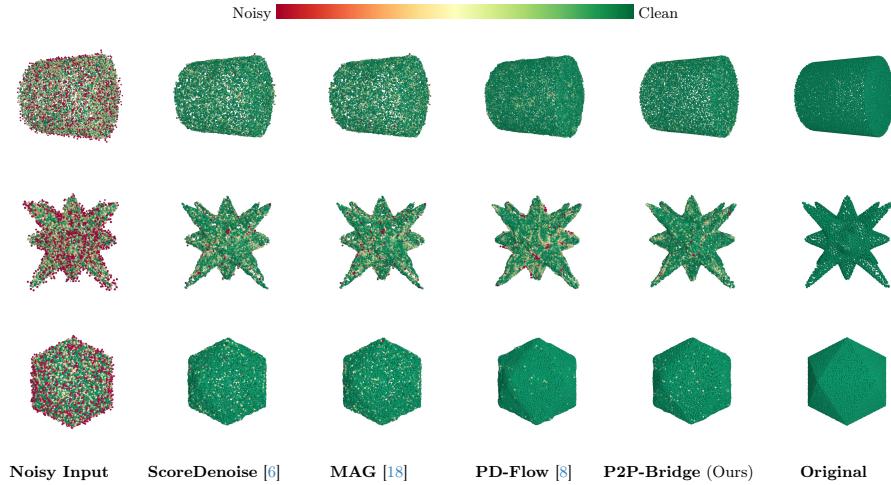
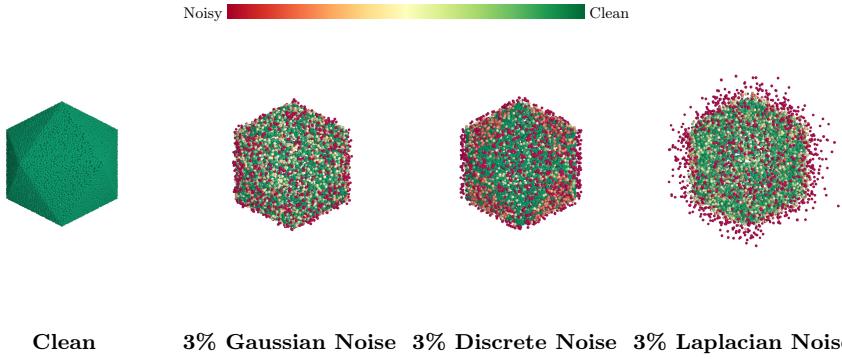


Fig. 3: PC-Net 3% Gaussian Noise. Qualitative comparison of our P2P-Bridge and recent deep-learning-based point cloud denoising methods on the PC-Net test set under 3% Gaussian noise.



Clean 3% Gaussian Noise 3% Discrete Noise 3% Laplacian Noise

Fig. 4: Comparison of different types of artificial noise.

Isotropic Gaussian Noise. Given a noise vector in three dimensions $\mathbf{x} = [n_x, n_y, n_z]^T$, where n_x , n_y and n_z are the noise components for each dimension, isotropic Gaussian noise with zero mean can be formulated using

$$p(\mathbf{n}) = \frac{1}{(2\pi\sigma^2)^{3/2}} \exp\left(-\frac{n_x^2 + n_y^2 + n_z^2}{2\sigma^2}\right), \quad (1)$$

where σ is the noise standard deviation. Isotropic Gaussian noise is the type of Gaussian noise we commonly refer to when discussing Gaussian noise.

Laplacian Noise. The Laplace distribution shares similarities with the Gaussian distribution, but it is characterized by a sharper peak and heavier tails. It is

commonly used to model the difference between two independent, identically distributed exponential random variables. We formulate Laplacian noise with zero mean and scale b for each noise component n_i with $i \in [x, y, z]$ using

$$p(n_i) = \frac{1}{2b} \exp\left(-\frac{|n_i|}{b}\right). \quad (2)$$

Discrete Noise. The nature of discrete noise lies in its representation of uncertainty or variability in a digital format, where the values it can take are distinct or quantized. Unlike continuous noise, which can assume any value within a given range, discrete noise is characterized by specific, separate values. Following ScoreDenoise, we use a discrete noise model

$$p(\mathbf{n}; s) = \begin{cases} 0.1 & \text{if } \mathbf{n} = (\pm s, 0, 0) \text{ or } (0, \pm s, 0) \text{ or } (0, 0, \pm s), \\ 0.4 & \text{if } \mathbf{n} = (0, 0, 0), \\ 0 & \text{otherwise,} \end{cases} \quad (3)$$

where s is controlling the amplitude of the noise.

Quantitative Results. Although all methods are trained on Gaussian noise removal, we also evaluate our method against Bilateral, ScoreDenoise, MAG, and PD-Flow on discrete and Laplacian noise. This shows how methods can generalize to not only different data but also different noise characteristics. The resulting metrics are listed in Tab. 8. Our method shows significantly better denoising performance than previous works, especially in the higher noise regimes.

Qualitative Results. Figure 5 shows the qualitative results on artificially corrupted objects using 3% noise strength of Gaussian, Laplace and discrete type.

Table 8: Object-Level Scores on Laplacian and Discrete Noise. We show the Chamfer distance (CD) and Point-2-Mesh distance (P2M) on the PU-Net (*top*) and PC-Net (*bottom*) datasets using discrete and Laplacian noise. Scores are multiplied by 10^4 .

| Method | Noise Amount | Num. of Points | | | | | | 10 · 10 ³ (sparse) | | | | | | 50 · 10 ³ (dense) | | | | | |
|-------------|--------------|--------------------------|-------------|-------------|-------------|-------------|--------------|-------------------------------|-------------|-------------|--------------|--------------|--------------|------------------------------|-----|----|-----|----|--|
| | | 1% | | 2% | | 3% | | 1% | | 2% | | 3% | | 1% | | 2% | | 3% | |
| | | CD | P2M | CD | P2M | CD | P2M | CD | P2M | CD | P2M | CD | P2M | CD | P2M | CD | P2M | | |
| PU-Net [15] | Laplacian | Bilateral [2] | 12.80 | 8.83 | 38.50 | 31.97 | 68.87 | 59.55 | 8.90 | 7.51 | 39.00 | 35.67 | 82.03 | 75.84 | | | | | |
| | | ScoreDenoise [6] | 6.68 | 3.51 | 32.96 | 27.29 | 94.23 | 84.06 | 3.17 | 2.16 | 24.00 | 21.44 | 95.75 | 88.10 | | | | | |
| | | MAG [18] | 6.69 | 3.52 | 32.97 | 27.30 | 94.34 | 84.16 | 3.18 | 2.16 | 24.02 | 21.46 | 95.90 | 88.24 | | | | | |
| | | PD-Flow [8] | 6.76 | 3.90 | 39.86 | 33.92 | 102.35 | 91.40 | 6.18 | 5.06 | 54.05 | 49.59 | 168.13 | 154.85 | | | | | |
| | | I-PFN [13] | 5.16 | 2.14 | 23.85 | 18.37 | 60.04 | 51.76 | 2.33 | 1.35 | 23.27 | 20.40 | 66.67 | 61.01 | | | | | |
| | Discrete | P2P-Bridge (Ours) | 4.22 | 1.63 | 7.80 | 4.69 | 16.20 | 12.04 | 1.82 | 1.05 | 14.55 | 12.04 | 59.54 | 52.24 | | | | | |
| | | Bilateral [2] | 2.60 | 0.76 | 5.62 | 3.05 | 11.76 | 8.74 | 0.99 | 0.44 | 2.84 | 2.13 | 8.68 | 7.79 | | | | | |
| PC-Net [10] | Laplacian | ScoreDenoise [6] | 2.43 | 0.50 | 4.07 | 1.63 | 7.90 | 4.87 | 0.73 | 0.20 | 2.90 | 2.13 | 9.80 | 8.78 | | | | | |
| | | MAG [18] | 2.44 | 0.51 | 4.07 | 1.63 | 7.90 | 4.87 | 0.74 | 0.21 | 2.90 | 2.14 | 9.81 | 8.79 | | | | | |
| | | PD-Flow [8] | 2.67 | 0.88 | 16.27 | 12.94 | 36.34 | 30.07 | 1.94 | 1.27 | 12.20 | 10.99 | 52.35 | 49.10 | | | | | |
| | | I-PFN [13] | 1.90 | 0.23 | 3.00 | 1.00 | 5.31 | 2.89 | 0.47 | 0.06 | 1.47 | 0.91 | 5.87 | 5.07 | | | | | |
| | | P2P-Bridge (Ours) | 2.14 | 0.40 | 2.89 | 0.92 | 4.23 | 1.94 | 0.53 | 0.12 | 1.30 | 0.71 | 4.05 | 3.29 | | | | | |
| | Discrete | Bilateral [2] | 15.38 | 10.47 | 35.85 | 28.04 | 61.46 | 49.92 | 21.40 | 19.72 | 91.13 | 90.06 | 272.65 | 280.39 | | | | | |
| | | ScoreDenoise [6] | 7.32 | 2.28 | 26.92 | 13.58 | 76.60 | 42.59 | 2.83 | 0.96 | 17.97 | 9.78 | 70.47 | 39.71 | | | | | |

2.6 Qualitative Results on ScanNet++

Fig. 6 shows more results from a closer point of view, highlighting the difference between the methods. Our method demonstrates better performance in both coarse and fine structures. However, it still lacks some details present in the ground truth, such as empty spaces between bars or small objects on surfaces.

2.7 Beyond Local Denoising

We additionally evaluate the effectiveness of our method in performing transformations that go beyond local denoising, including the mitigation of fitting errors such as shifts. Figure 7 depicts walls from an outside point of view, revealing large outlier surfaces. The figure highlights our method’s ability to identify and remove clusters of outliers.

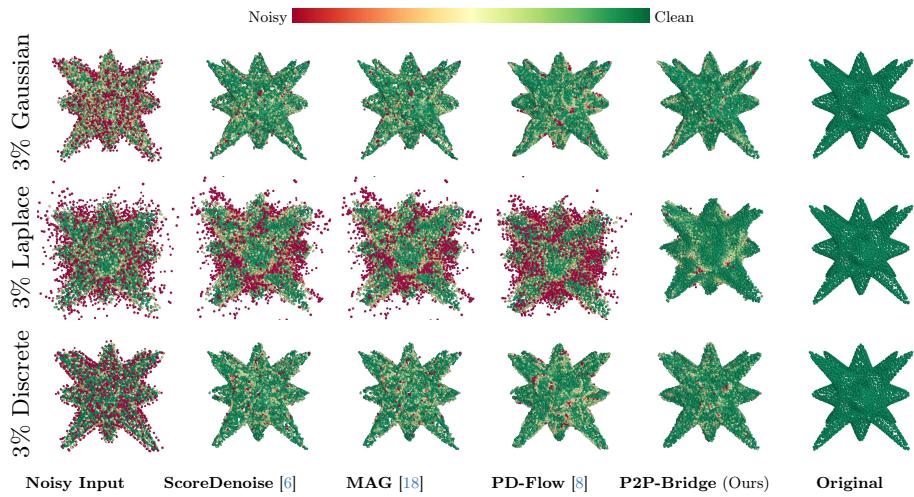


Fig. 5: PC-Net Varying Noise Types. Comparison of our method with recent deep learning-based point cloud denoising methods on an object from the PC-Net dataset. Varying noise types are applied to a point cloud with a size of 10k points.

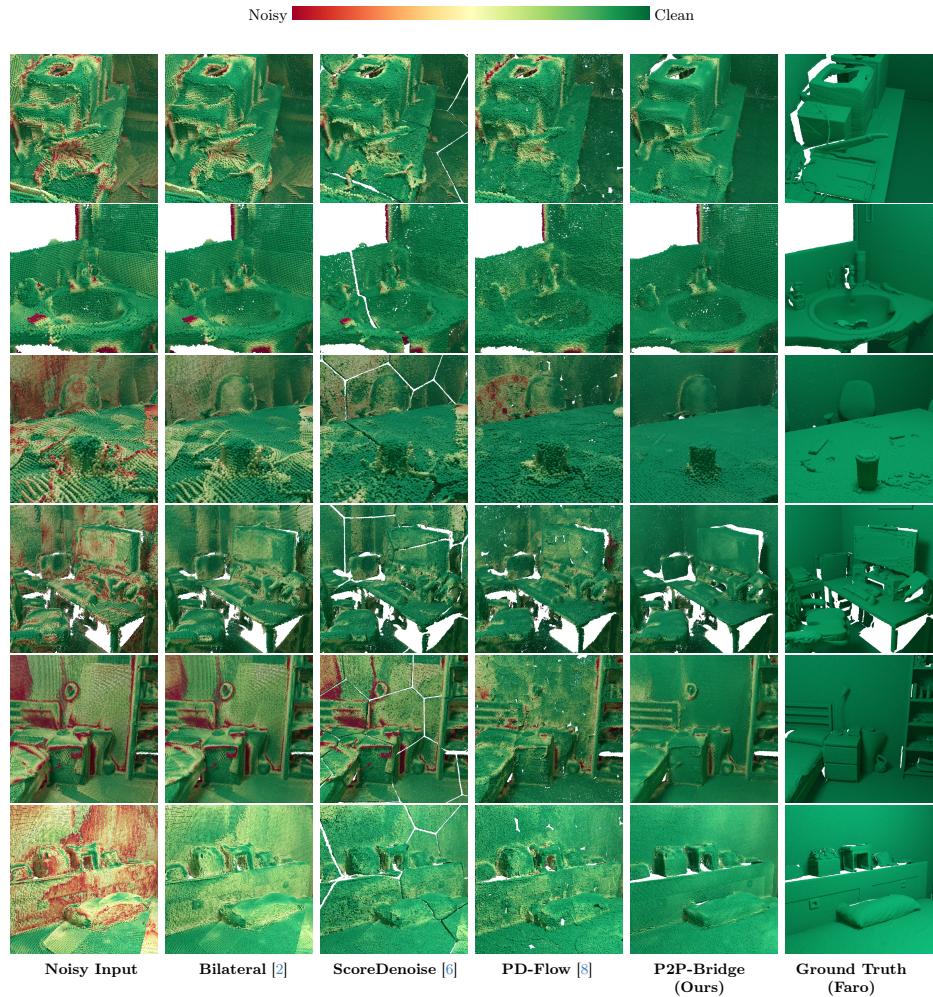


Fig. 6: ScanNet++ Details. Qualitative comparison on ScanNet++ [14] using noisy iPhone scans as input.

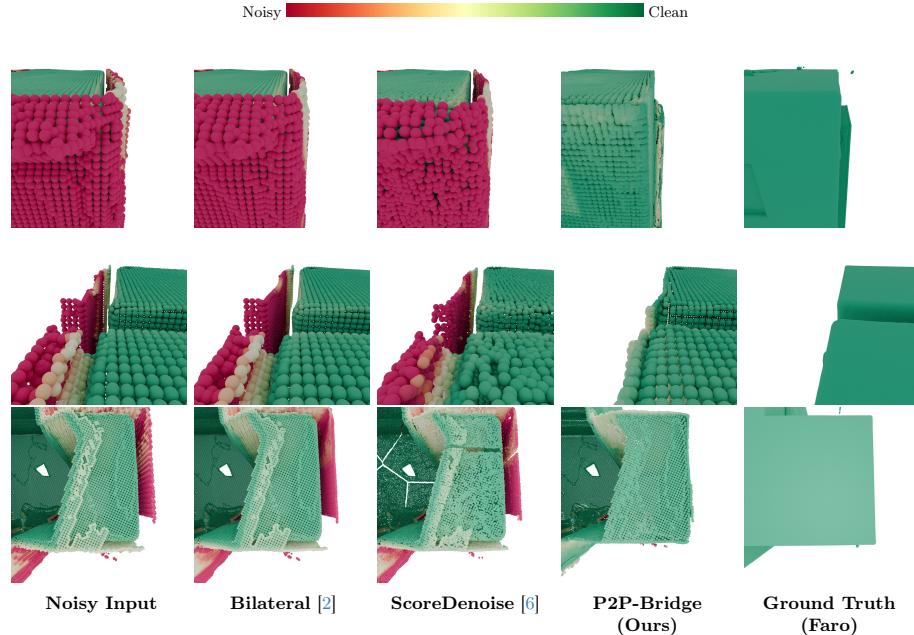


Fig. 7: Qualitative comparison between best performing methods on ScanNet++ [14] using noisy iPhone scans as input that got refined using 3DMatch [16]. This comparison focuses on the ability to remove outlier clusters, especially surfaces. These mostly occur alongside large surfaces such as walls, tables, or wardrobes.

References

1. Chen, Y., Hu, V.T., Gavves, E., Mensink, T., Mettes, P., Yang, P., Snoek, C.G.: PointMixup: Augmentation for Point Clouds. In: European Conference on Computer Vision (ECCV) (2020) [3](#)
2. Digne, J., de Franchis, C.: The Bilateral Filter for Point Clouds. In: Image Processing On Line (2017) [10](#), [12](#), [13](#)
3. Huang, S., Gojcic, Z., Usyatsov, M., Wieser, A., Schindler, K.: Predator: Registration of 3d point clouds with low overlap. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2021) [1](#)
4. Liu, G.H., Vahdat, A., Huang, D.A., Theodorou, E.A., Nie, W., Anandkumar, A.: I2SB: Image-to-Image Schrödinger Bridge. In: International Conference on Machine Learning (ICML) (2023) [4](#)
5. Liu, Z., Tang, H., Lin, Y., Han, S.: Point-Voxel CNN for Efficient 3D Deep Learning (2019) [3](#), [5](#)
6. Luo, S., Hu, W.: Score-Based Point Cloud Denoising. In: International Conference on Computer Vision (ICCV) (2021) [8](#), [10](#), [11](#), [12](#), [13](#)
7. Lyu, Z., Kong, Z., XU, X., Pan, L., Lin, D.: A Conditional Point Diffusion-Refinement Paradigm for 3D Point Cloud Completion. In: International Conference on Learning Representations (ICLR) (2022) [6](#), [7](#)
8. Mao, A., Du, Z., Wen, Y.H., Xuan, J., Liu, Y.J.: PD-Flow: A point cloud denoising framework with normalizing flows. In: European Conference on Computer Vision (ECCV) (2022) [8](#), [10](#), [11](#), [12](#)
9. Peyré, G., Cuturi, M.: Computational optimal transport. Foundations and Trends in Machine Learning (2019) [7](#)
10. Rakotosaona, M.J., La Barbera, V., Guerrero, P., Mitra, N.J., Ovsjanikov, M.: PointCleanNet: Learning to Denoise and Remove Outliers from Dense Point Clouds (2020) [10](#)
11. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2016) [2](#)
12. Schönberger, J.L., Zheng, E., Pollefeys, M., Frahm, J.M.: Pixelwise view selection for unstructured multi-view stereo. In: European Conference on Computer Vision (ECCV) (2016) [2](#)
13. de Silva Edirimuni, D., Lu, X., Shao, Z., Li, G., Robles-Kelly, A., He, Y.: Iterativepfn: True iterative point cloud filtering. In: International Conference on Computer Vision and Pattern Recognition (CVPR). pp. 13530–13539 (2023) [10](#)
14. Yeshwanth, C., Liu, Y.C., Nießner, M., Dai, A.: ScanNet++: A High-Fidelity Dataset of 3D Indoor Scenes. In: International Conference on Computer Vision (ICCV) (2023) [12](#), [13](#)
15. Yu, L., Li, X., Fu, C.W., Cohen-Or, D., Heng, P.A.: PU-Net: Point Cloud Up-sampling Network. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2018) [7](#), [10](#)
16. Zeng, A., Song, S., Nießner, M., Fisher, M., Xiao, J., Funkhouser, T.: 3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions. In: CVPR (2017) [2](#), [13](#)
17. Zeng, X., Vahdat, A., Williams, F., Gojcic, Z., Litany, O., Fidler, S., Kreis, K.: LION: Latent Point Diffusion Models for 3D Shape Generation. In: International Conference on Neural Information Processing Systems (NeurIPS) (2022) [3](#)
18. Zhao, Y., Zheng, H., Wang, Z., Luo, J., Lam, E.Y.: Point Cloud Denoising via Momentum Ascent in Gradient Fields (2023) [8](#), [10](#), [11](#)

19. Zhou, L., Du, Y., Wu, J.: 3D Shape Generation and Completion Through Point-Voxel Diffusion. In: International Conference on Computer Vision (ICCV) (2021)
[6](#), [7](#)