IterativePFN:

True Iterative Point Cloud Filtering

Dasith de Silva Edirimuni[1] Xuequan Lu[1] Zhiwen Shao[2]
Gang Li[1] Antonio Robles-Kelly[1,4] Ying He[3]

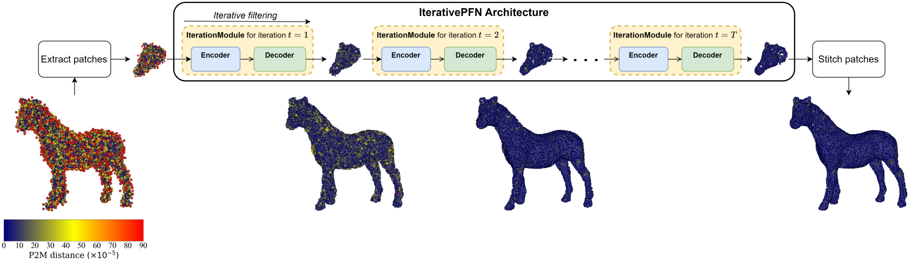
[1]Deakin University, [2]China University of Mining and Technology, [3]Nanyang Technological University, [4]Defence Science and Technology Group, Australia

JUNE 18-22, 2023 CVPR VANCOUVER, CANADA

Poster: WED-PM-112

At a glance

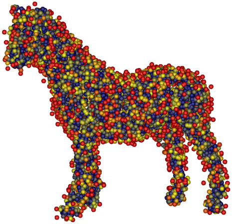
- Current methods $\leftrightarrow$ iterative filtering only at test time
- Our method $\leftrightarrow$ models iterative filtering at train + test time



- Adaptive ground truth loss
- Generalized patch stitching mechanism

Overview

Filtering/denoising is a fundamental point cloud processing task

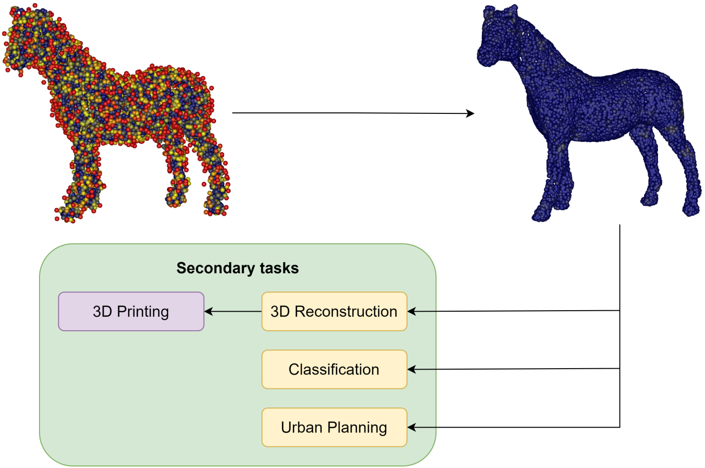


Overview

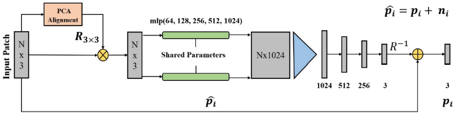
Filtering/denoising is a fundamental point cloud processing task

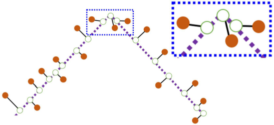
Overview

Filtering/denoising is a fundamental point cloud processing task



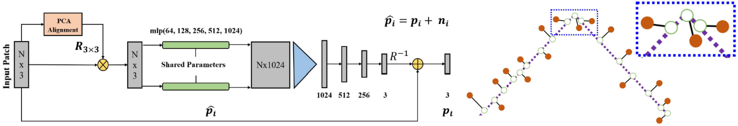
Displacement-based methods → Pointfilter, IEEE TVCG, 2021



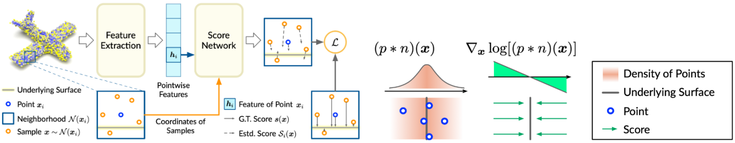


Current works

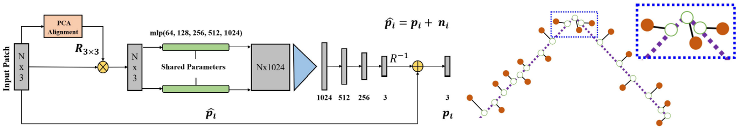
Displacement-based methods → Pointfilter, IEEE TVCG, 2021



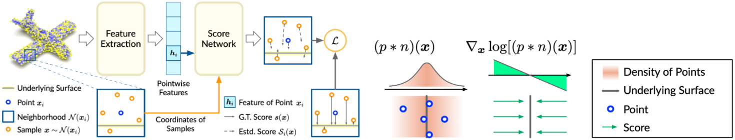
Probability-based methods → ScoreDenoise, ICCV, 2021



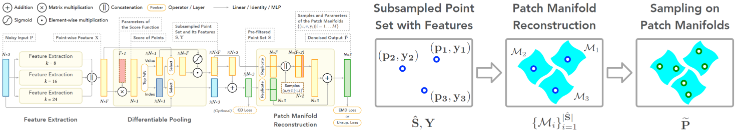
Displacement-based methods → Pointfilter, IEEE TVCG, 2021



Probability-based methods → ScoreDenoise, ICCV, 2021



Resampling-based methods → DMRDenoise, ACM MM, 2020



Current works

Displacement-based methods infer displacements to filter noisy points

Their filtering objective is expressed as,

$$\tilde{\boldsymbol{x}}_i = \boldsymbol{x}_i + \boldsymbol{d}_i \tag{1}$$

At test-time $\rightarrow$ iterate process:

$$\tilde{\boldsymbol{x}}_i^{(t)} = \tilde{\boldsymbol{x}}_i^{(t-1)} + \boldsymbol{d}_i^{(t)}, t = 1, \cdots, T \tag{2}$$

Probabilistic score-based methods infer $\mathcal{S}_i(\boldsymbol{x}) \rightarrow \nabla_{\boldsymbol{x}} \log[(p * n)(\boldsymbol{x}_i)]$

$$\tilde{\boldsymbol{x}}_i^{(t)} = \tilde{\boldsymbol{x}}_i^{(t-1)} + \alpha^{(t)} \mathcal{E}_i(\tilde{\boldsymbol{x}}_i^{(t-1)}), \ t = 1, \cdots, T \tag{3}$$

where $\mathcal{E}_i(\boldsymbol{x}) = (1/K) \sum_{\boldsymbol{x}_j \in kNN(\boldsymbol{x}_i)} \mathcal{S}_j(\boldsymbol{x})$.

Current works

Displacement-based methods infer displacements to filter noisy points

Their filtering objective is expressed as,

$$\tilde{\boldsymbol{x}}_i = \boldsymbol{x}_i + \boldsymbol{d}_i \tag{1}$$

At test-time → iterate process:

$$\tilde{\boldsymbol{x}}_i^{(t)} = \tilde{\boldsymbol{x}}_i^{(t-1)} + \boldsymbol{d}_i^{(t)}, t = 1, \cdots, T \tag{2}$$

Probabilistic score-based methods infer $\mathcal{S}_i(\boldsymbol{x}) \rightarrow \nabla_{\boldsymbol{x}} \log[(p * n)(\boldsymbol{x}_i)]$

$$\tilde{\boldsymbol{x}}_i^{(t)} = \tilde{\boldsymbol{x}}_i^{(t-1)} + \alpha^{(t)} \mathcal{E}_i(\tilde{\boldsymbol{x}}_i^{(t-1)}), \ t = 1, \cdots, T \tag{3}$$

where $\mathcal{E}_i(\boldsymbol{x}) = (1/K) \sum_{\boldsymbol{x}_j \in kNN(\boldsymbol{x}_i)} \mathcal{S}_j(\boldsymbol{x})$.

Current works

Displacement-based methods infer displacements to filter noisy points

Their filtering objective is expressed as,

$$\tilde{\boldsymbol{x}}_i = \boldsymbol{x}_i + \boldsymbol{d}_i \tag{1}$$

At test-time $\rightarrow$ iterate process:

$$\tilde{\boldsymbol{x}}_i^{(t)} = \tilde{\boldsymbol{x}}_i^{(t-1)} + \boldsymbol{d}_i^{(t)}, t = 1, \cdots, T \tag{2}$$

Probabilistic score-based methods infer $\mathcal{S}_i(\boldsymbol{x}) \rightarrow \nabla_{\boldsymbol{x}} \log[(p * n)(\boldsymbol{x}_i)]$

$$\tilde{\boldsymbol{x}}_i^{(t)} = \tilde{\boldsymbol{x}}_i^{(t-1)} + \alpha^{(t)} \mathcal{E}_i(\tilde{\boldsymbol{x}}_i^{(t-1)}), \ t = 1, \cdots, T \tag{3}$$

where $\mathcal{E}_i(\boldsymbol{x}) = (1/K) \sum_{\boldsymbol{x}_j \in kNN(\boldsymbol{x}_i)} \mathcal{S}_j(\boldsymbol{x})$.

Current works

Displacement-based methods infer displacements to filter noisy points

Their filtering objective is expressed as,

$$\tilde{\boldsymbol{x}}_i = \boldsymbol{x}_i + \boldsymbol{d}_i \tag{1}$$

At test-time $\rightarrow$ iterate process:

$$\tilde{\boldsymbol{x}}_i^{(t)} = \tilde{\boldsymbol{x}}_i^{(t-1)} + \boldsymbol{d}_i^{(t)}, t = 1, \cdots, T \tag{2}$$

Probabilistic score-based methods infer $\mathcal{S}_i(\boldsymbol{x}) \rightarrow \nabla_{\boldsymbol{x}} \log[(p * n)(\boldsymbol{x}_i)]$

$$\tilde{\boldsymbol{x}}_i^{(t)} = \tilde{\boldsymbol{x}}_i^{(t-1)} + \alpha^{(t)} \mathcal{E}_i(\tilde{\boldsymbol{x}}_i^{(t-1)}),\ t = 1, \cdots, T \tag{3}$$

where $\mathcal{E}_i(\boldsymbol{x}) = (1/K) \sum_{\boldsymbol{x}_j \in kNN(\boldsymbol{x}_i)} \mathcal{S}_j(\boldsymbol{x})$.

Motivation

Current works have two main limitations:

1. Iterative only at test time
2. Filtered results do not quickly converge to the clean surface

Motivation

Current works have two main limitations:

1. Iterative only at test time
2. Filtered results do not quickly converge to the clean surface

Motivation

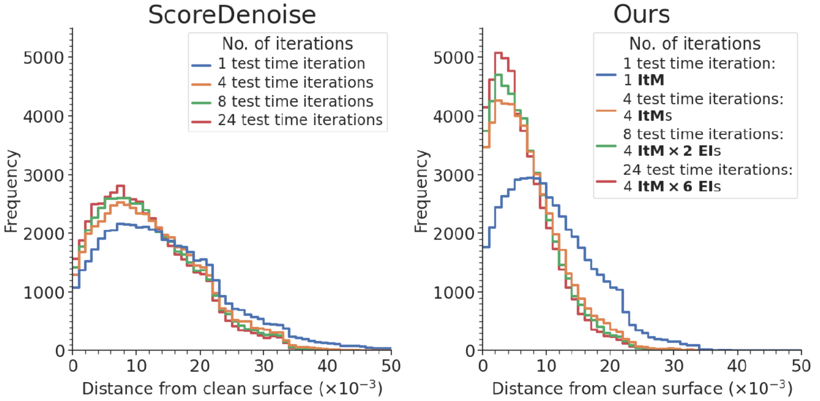
Current works have two main limitations:

1. Iterative only at test time
2. Filtered results do not quickly converge to the clean surface

Motivation

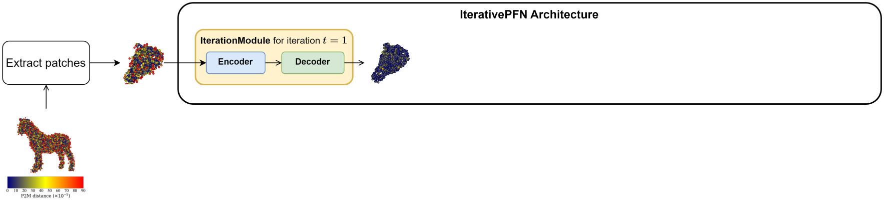
Current works have two main limitations:

1. Iterative only at test time
2. Filtered results do not quickly converge to the clean surface



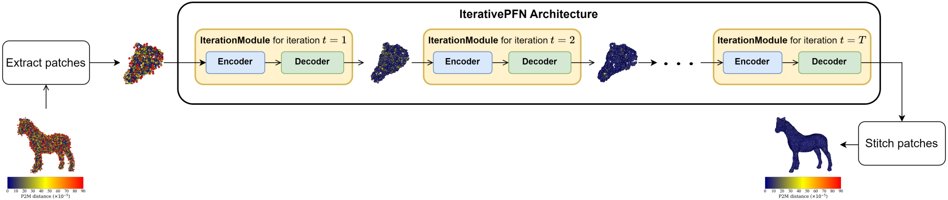
IterativePFN network

We propose an iterative filtering mechanism that is truly iterative at train and test times

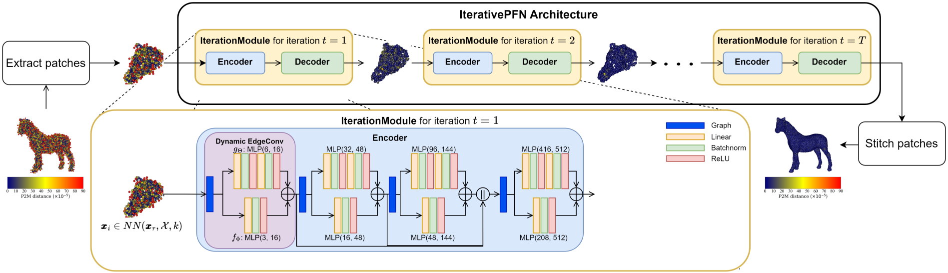


IterativePFN network

We propose an iterative filtering mechanism that is truly iterative at train and test times

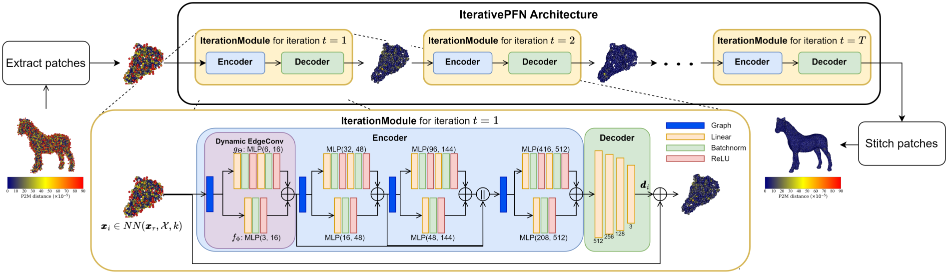


We propose an iterative filtering mechanism that is truly iterative at train and test times



IterativePFN network

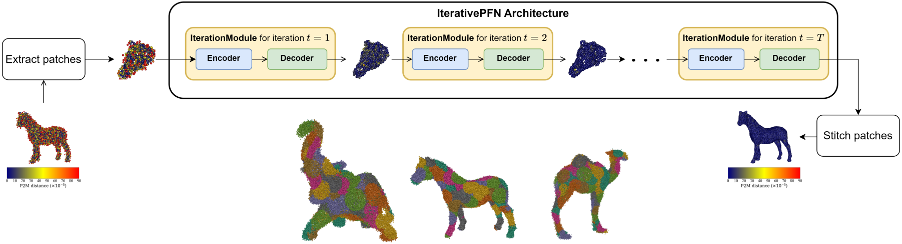
We propose an iterative filtering mechanism that is truly iterative at train and test times



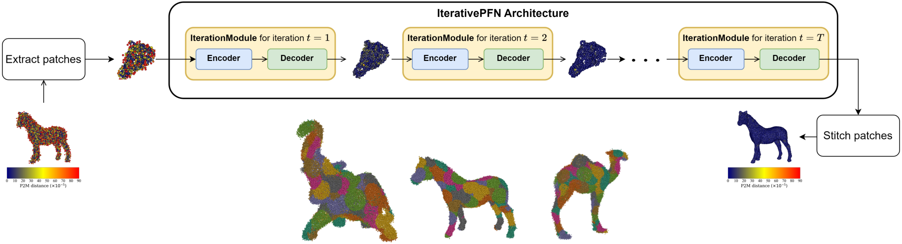
Each IterationModule only needs filtered positions from the previous iteration as input

$$\tilde{\boldsymbol{x}}_i^{(t)} = \tilde{\boldsymbol{x}}_i^{(t-1)} + \boldsymbol{d}_i^{(t)}, t = 1, \cdots, T \qquad (4)$$

Pre-processing

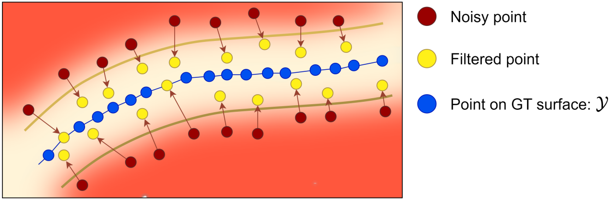


Pre-processing



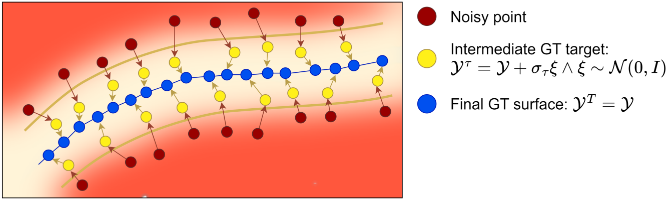
$$w_i = \frac{\exp\left(-\left\|\boldsymbol{x}_i - \boldsymbol{x}_r\right\|_2^2 / r_s^2\right)}{\sum_i \exp\left(-\left\|\boldsymbol{x}_i - \boldsymbol{x}_r\right\|_2^2 / r_s^2\right)}$$

Training objective: Adaptive Ground Truth loss function



$$L_i^{PCN} = \alpha \min_{\boldsymbol{x}_j \in \mathcal{Y}} \|\boldsymbol{d}_i - (\boldsymbol{x}_j - \boldsymbol{x}_i)\|_2^2 + (1-\alpha) \max_{\boldsymbol{x}_j \in \mathcal{Y}} \|\boldsymbol{d}_i - (\boldsymbol{x}_j - \boldsymbol{x}_i)\|_2^2$$

Training objective: Adaptive Ground Truth loss function



$$L_i^{(\tau)}(\mathcal{Y}^{(\tau)}) = \left\| \boldsymbol{d}_i^{(\tau)} - \left[NN(\boldsymbol{x}_i^{(\tau-1)}, \mathcal{Y}^{(\tau)}) - \boldsymbol{x}_i^{(\tau-1)} \right] \right\|_2^2,$$

Training objective: Adaptive Ground Truth loss function

$$L_i^{(\tau)}(\mathcal{Y}^{(\tau)}) = \left\| \boldsymbol{d}_i^{(\tau)} - \left[NN(\boldsymbol{x}_i^{(\tau-1)}, \mathcal{Y}^{(\tau)}) - \boldsymbol{x}_i^{(\tau-1)} \right] \right\|_2^2,$$

- Gaussian weights based on position from patch center

$$w_i = \frac{\exp\left(- \left\| \boldsymbol{x}_i - \boldsymbol{x}_r \right\|_2^2 / r_s^2\right)}{\sum_i \exp\left(- \left\| \boldsymbol{x}_i - \boldsymbol{x}_r \right\|_2^2 / r_s^2\right)},$$

- Single IterationModule loss $\leftrightarrow$ weighted average across points

$$L^{(\tau)} = \sum_i w_i L_i^{(\tau)},$$

- Sum loss contributions across all ItMs $\rightarrow$ allows joint training

$$\mathcal{L}_a = \sum_{\tau=1}^{T} L^{(\tau)}.$$

Training objective: Adaptive Ground Truth loss function

$$L_i^{(\tau)}(\mathcal{Y}^{(\tau)}) = \left\| \boldsymbol{d}_i^{(\tau)} - \left[NN(\boldsymbol{x}_i^{(\tau-1)}, \mathcal{Y}^{(\tau)}) - \boldsymbol{x}_i^{(\tau-1)} \right] \right\|_2^2 ,$$

- Gaussian weights based on position from patch center

$$w_i = \frac{\exp\left(- \left\| \boldsymbol{x}_i - \boldsymbol{x}_r \right\|_2^2 / r_s^2 \right)}{\sum_i \exp\left(- \left\| \boldsymbol{x}_i - \boldsymbol{x}_r \right\|_2^2 / r_s^2 \right)},$$

- Single IterationModule loss $\leftrightarrow$ weighted average across points

$$L^{(\tau)} = \sum_i w_i L_i^{(\tau)},$$

- Sum loss contributions across all ItMs $\rightarrow$ allows joint training

$$\mathcal{L}_a = \sum_{\tau=1}^{T} L^{(\tau)}.$$

Training objective: Adaptive Ground Truth loss function

$$L_i^{(\tau)}(\mathcal{Y}^{(\tau)}) = \left\| \boldsymbol{d}_i^{(\tau)} - \left[NN(\boldsymbol{x}_i^{(\tau-1)}, \mathcal{Y}^{(\tau)}) - \boldsymbol{x}_i^{(\tau-1)} \right] \right\|_2^2,$$

- Gaussian weights based on position from patch center

$$w_i = \frac{\exp\left(- \|\boldsymbol{x}_i - \boldsymbol{x}_r\|_2^2 / r_s^2\right)}{\sum_i \exp\left(- \|\boldsymbol{x}_i - \boldsymbol{x}_r\|_2^2 / r_s^2\right)},$$

- Single IterationModule loss $\leftrightarrow$ weighted average across points

$$L^{(\tau)} = \sum_i w_i L_i^{(\tau)},$$

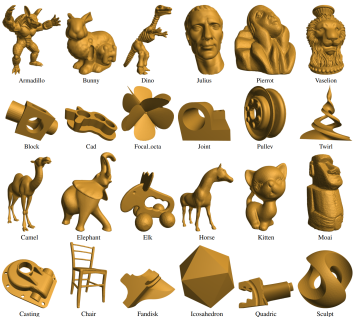
- Sum loss contributions across all ItMs $\rightarrow$ allows joint training

$$\mathcal{L}_a = \sum_{\tau=1}^{T} L^{(\tau)}.$$

Datasets

We use both synthetic and real-world scanned data to analyze our method's performance

- Training set (40 models)
  - Gaussian noise for training
  - 3 resolutions (10K, 30K, 50K)
  - Noise scales 0.5% - 2% of BSR
- Test set (169 models)
  - 20 synthetic noisy models
  - 2 resolutions (10K, 50K)
  - Noise scales 1% - 2.5% of BSR
  - 5 different noise patterns
  - 4 raw outdoor laser-scanned scenes
  - 72 + 73 raw Kinect v1 and Kinect v2 scanned models

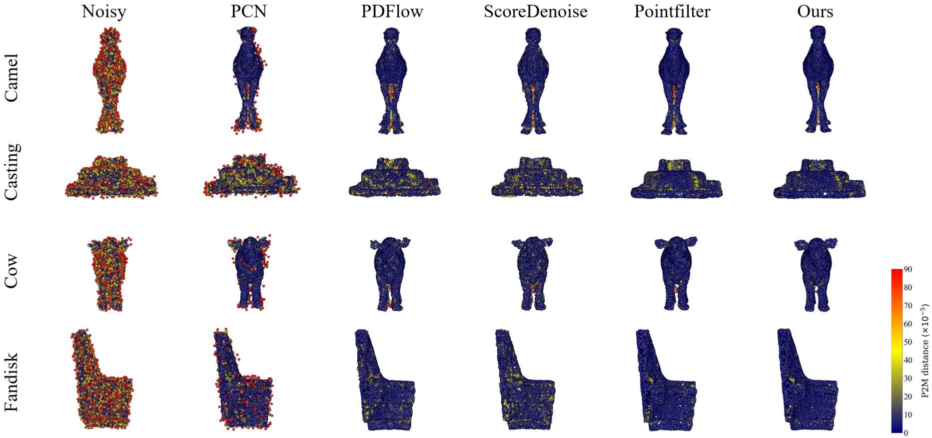


Example train and test models from synthetic PUNet dataset[a]

[a] Yu et al. PU-Net: Point Cloud Upsampling

Results on PUNet test set

First we look at results on our synthetic dataset:



- Our method effectively filters both complex shapes such as Casting and simpler shapes such as Fandisk

Results on PUNet test set

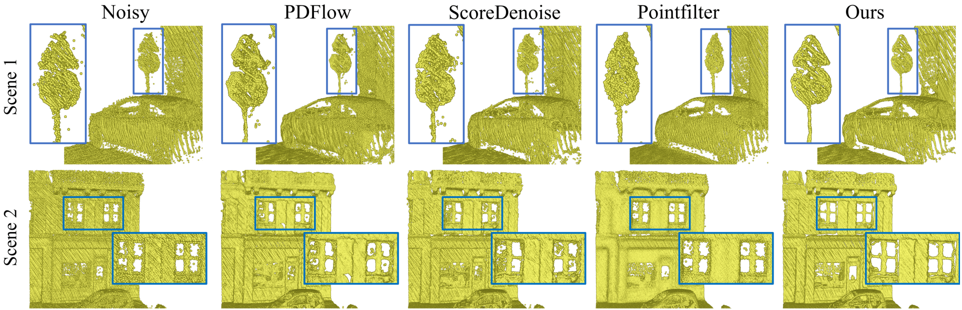
Method	10K points						50K points					
	1% noise		2% noise		2.5% noise		1% noise		2% noise		2.5% noise	
	CD	P2M	CD	P2M	CD	P2M	CD	P2M	CD	P2M	CD	P2M
Noisy	36.9	16.03	79.39	47.72	105.02	70.03	18.69	12.82	50.48	41.36	72.49	62.03
PCN	36.86	15.99	79.26	47.59	104.86	69.87	11.03	6.46	19.78	13.7	32.03	24.86
GPDNet	23.1	7.14	42.84	18.55	58.37	30.66	10.49	6.35	32.88	25.03	50.85	41.34
DMRDenoise	47.12	21.96	50.85	25.23	52.77	26.69	12.05	7.62	14.43	9.7	16.96	11.9
PDFlow	21.26	6.74	32.46	13.24	36.27	17.02	6.51	4.16	12.7	9.21	18.74	14.26
ScoreDenoise	25.22	7.54	36.83	13.8	42.32	19.04	7.16	4.0	12.89	8.33	14.45	9.58
Pointfilter	24.61	7.3	35.34	11.55	40.99	15.05	7.58	4.32	9.07	5.07	10.99	6.29
Ours	**20.56**	**5.01**	**30.43**	**8.45**	**33.52**	**10.45**	**6.05**	**3.02**	**8.03**	**4.36**	**10.15**	**5.88**

Table: Filtering results on the PUNet dataset. CD and P2M distances are multiplied by 10^5

- Our method outperforms others across resolutions and noise scales

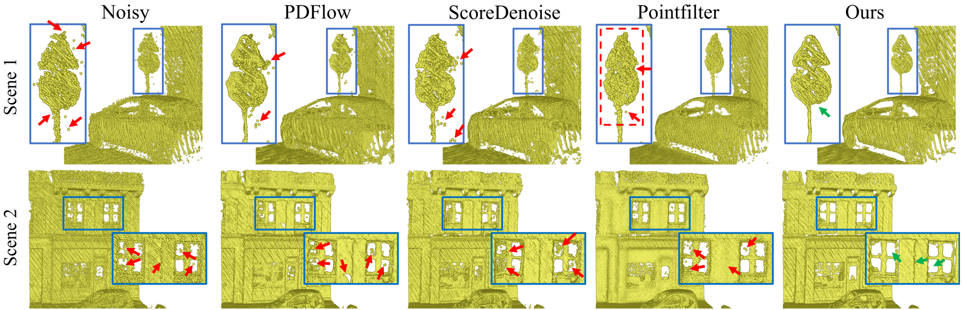
Visual results on raw laser-scanned data

We next look at results on the Rue Madame dataset



Visual results on raw laser-scanned data

We next look at results on the Rue Madame dataset



- Our method effectively filters points while others smear sharp features or leave behind outliers

Ablation: Iteration number and alternative Fixed GT loss

Ablation	10K points					
	1% noise		2% noise		2.5% noise	
	CD	P2M	CD	P2M	CD	P2M
$\mathcal{L}_a$ & 1 it.	21.95	5.42	32.38	9.55	36.98	12.71
$\mathcal{L}_a$ & 2 it.	21.13	5.14	30.82	8.67	34.33	11.0
$\mathcal{L}_a$ & 4 it.	20.56	5.01	30.43	8.45	**33.52**	**10.45**
$\mathcal{L}_a$ & 8 it.	**19.78**	**4.9**	**30.12**	**8.3**	33.88	10.78
$\mathcal{L}_a$ & 12 it.	20.49	5.23	30.64	8.87	34.46	11.25
$\mathcal{L}_a$ & DPFN	21.03	5.05	30.96	8.53	35.2	11.4
$\mathcal{L}_b$ & 4 it.	20.64	5.04	30.59	8.54	34.17	10.87

Table: Ablation results for different iteration numbers and different loss functions. CD and P2M distances are multiplied by 10^5

- At high iteration numbers $\rightarrow$ the network over-specializes on the training noise
- 4 iterations is optimal
- To investigate impact of AGT loss $\mathcal{L}_a$, we consider

$$\mathcal{L}_b = \sum_{\tau=1}^{T} \left[\sum_i w_i \left(\left\| \boldsymbol{d}_i^{(\tau)} - (NN(\boldsymbol{x}_i^{(\tau-1)}, \mathcal{Y}) - \boldsymbol{x}_i^{(\tau-1)}) \right\|_2^2 \right) \right],$$

Ablation: Iteration number and alternative Fixed GT loss

Ablation	10K points					
	1% noise		2% noise		2.5% noise	
	CD	P2M	CD	P2M	CD	P2M
$\mathcal{L}_a$ & 1 it.	21.95	5.42	32.38	9.55	36.98	12.71
$\mathcal{L}_a$ & 2 it.	21.13	5.14	30.82	8.67	34.33	11.0
$\mathcal{L}_a$ & 4 it.	<u>20.56</u>	<u>5.01</u>	<u>30.43</u>	<u>8.45</u>	**33.52**	**10.45**
$\mathcal{L}_a$ & 8 it.	**19.78**	**4.9**	**30.12**	**8.3**	<u>33.88</u>	<u>10.78</u>
$\mathcal{L}_a$ & 12 it.	20.49	5.23	30.64	8.87	34.46	11.25
$\mathcal{L}_a$ & DPFN	21.03	5.05	30.96	8.53	35.2	11.4
$\mathcal{L}_b$ & 4 it.	20.64	5.04	30.59	8.54	34.17	10.87

Table: Ablation results for different iteration numbers and different loss functions. CD and P2M distances are multiplied by 10^5

- At high iteration numbers $\rightarrow$ the network over-specializes on the training noise
- 4 iterations is optimal
- To investigate impact of AGT loss $\mathcal{L}_a$, we consider

$$\mathcal{L}_b = \sum_{\tau=1}^{T} \left[\sum_i w_i \left(\left\| \boldsymbol{d}_i^{(\tau)} - (NN(\boldsymbol{x}_i^{(\tau-1)}, \mathcal{Y}) - \boldsymbol{x}_i^{(\tau-1)}) \right\|_2^2 \right) \right],$$

Ablation: Iteration number and alternative Fixed GT loss

Ablation	10K points					
	1% noise		2% noise		2.5% noise	
	CD	P2M	CD	P2M	CD	P2M
$\mathcal{L}_a$ & 1 it.	21.95	5.42	32.38	9.55	36.98	12.71
$\mathcal{L}_a$ & 2 it.	21.13	5.14	30.82	8.67	34.33	11.0
$\mathcal{L}_a$ & 4 it.	20.56	5.01	30.43	8.45	**33.52**	**10.45**
$\mathcal{L}_a$ & 8 it.	**19.78**	**4.9**	**30.12**	**8.3**	33.88	10.78
$\mathcal{L}_a$ & 12 it.	20.49	5.23	30.64	8.87	34.46	11.25
$\mathcal{L}_a$ & DPFN	21.03	5.05	30.96	8.53	35.2	11.4
$\mathcal{L}_b$ & 4 it.	20.64	5.04	30.59	8.54	34.17	10.87

Table: Ablation results for different iteration numbers and different loss functions. CD and P2M distances are multiplied by 10^5

- At high iteration numbers $\rightarrow$ the network over-specializes on the training noise
- 4 iterations is optimal
- To investigate impact of AGT loss $\mathcal{L}_a$, we consider

$$\mathcal{L}_b = \sum_{\tau=1}^{T}\left[\sum_i w_i\left(\left\|\boldsymbol{d}_i^{(\tau)} - (NN(\boldsymbol{x}_i^{(\tau-1)},\mathcal{Y}) - \boldsymbol{x}_i^{(\tau-1)})\right\|_2^2\right)\right],$$

Ablation: Iteration number and alternative Fixed GT loss

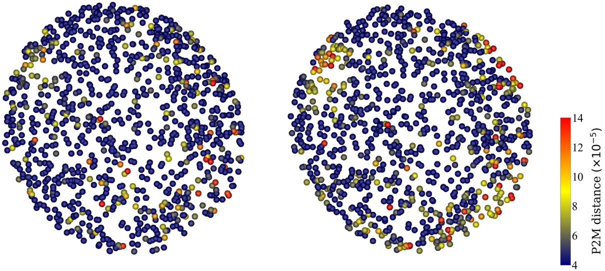
Ablation	10K points					
	1% noise		2% noise		2.5% noise	
	CD	P2M	CD	P2M	CD	P2M
$\mathcal{L}_a$ & 1 it.	21.95	5.42	32.38	9.55	36.98	12.71
$\mathcal{L}_a$ & 2 it.	21.13	5.14	30.82	8.67	34.33	11.0
$\mathcal{L}_a$ & 4 it.	20.56	5.01	30.43	8.45	**33.52**	**10.45**
$\mathcal{L}_a$ & 8 it.	**19.78**	**4.9**	**30.12**	**8.3**	33.88	10.78
$\mathcal{L}_a$ & 12 it.	20.49	5.23	30.64	8.87	34.46	11.25
$\mathcal{L}_a$ & DPFN	21.03	5.05	30.96	8.53	35.2	11.4
$\mathcal{L}_b$ & 4 it.	20.64	5.04	30.59	8.54	34.17	10.87

Table: Ablation results for different iteration numbers and different loss functions. CD and P2M distances are multiplied by 10^5

- At high iteration numbers $\rightarrow$ the network over-specializes on the training noise
- 4 iterations is optimal
- To investigate impact of AGT loss $\mathcal{L}_a$, we consider

$$\mathcal{L}_b = \sum_{\tau=1}^{T} \left[\sum_i w_i \left(\left\| \boldsymbol{d}_i^{(\tau)} - (NN(\boldsymbol{x}_i^{(\tau-1)}, \mathcal{Y}) - \boldsymbol{x}_i^{(\tau-1)}) \right\|_2^2 \right) \right],$$

Ablation: With/without patch stitching



(a) With patch stitching

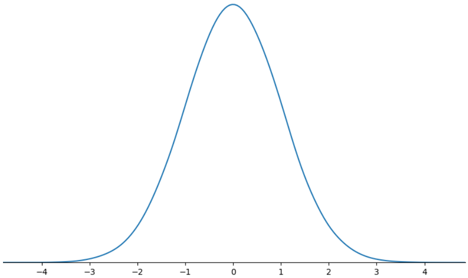
(b) Without patch stitching

Figure: Visual results of a filtered patch, with and without stitching

Ablation	10K points					
	1% noise		2% noise		2.5% noise	
	CD	P2M	CD	P2M	CD	P2M
without PS	21.19	5.45	32.38	10.2	38.67	14.98
with PS	**20.56**	**5.01**	**30.43**	**8.45**	**33.52**	**10.45**

Table: Ablation results with and without patch stitching (PS). CD and P2M distances are multiplied by 10^5

Limitations and future work



- Generating adaptive targets requires noise distribution that is easy to replicate
- Generalize approach to use noisy data simulating real world noise

Thanks for watching!

IterativePFN: True Iterative Point Cloud Filtering

Dasith de Silva Edirimuni[1] Xuequan Lu[1] Zhiwen Shao[2]
Gang Li[1] Antonio Robles-Kelly[1,4] Ying He[3]

[1]Deakin University, [2]China University of Mining and Technology, [3]Nanyang Technological University, [4]Defence Science and Technology Group, Australia

- Please visit our project page for more information:
  https://ddsediri.github.io/projects/IterativePFN
- Code is available at:
  https://github.com/ddsediri/IterativePFN