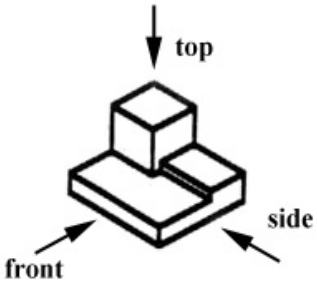
Deep Point Cloud Upsampling

Presenter: Li Xianzhi (李贤芝)
Department of Computer Science and Engineering
The Chinese University of Hong Kong

OUTLINE

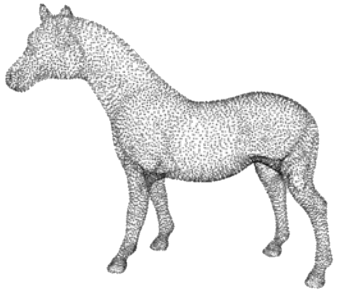
- Background
- Our works
  - PU-Net --- accepted by CVPR, 2018
  - EC-Net --- accepted by ECCV, 2018
  - PU-GAN --- accepted by ICCV, 2019
- Future works

3D representations:

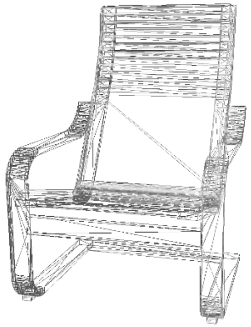


multi-view images

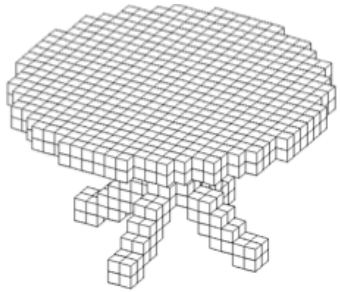
depth maps



point cloud

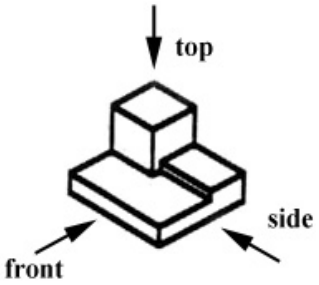


polygonal mesh



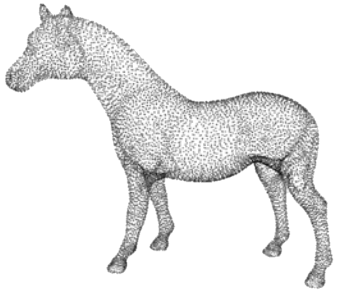
volume

3D representations:

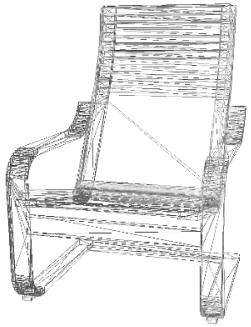


multi-view images

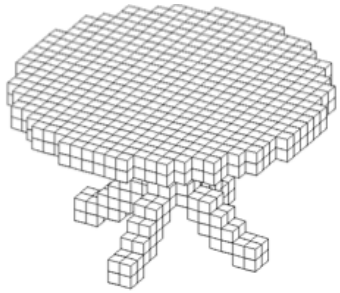
depth maps



point cloud



polygonal mesh



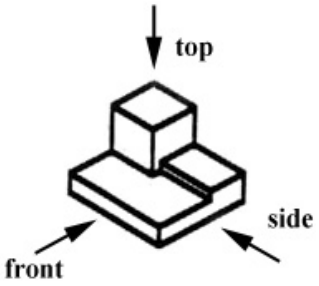
volume

- occlusion issue

- complex

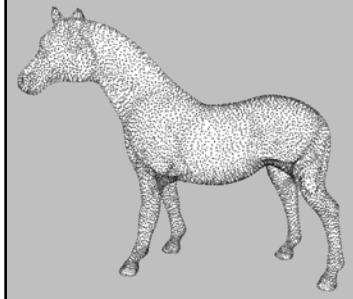
- low resolution

3D representations:

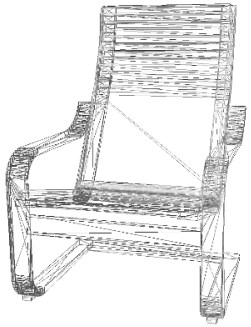


multi-view images

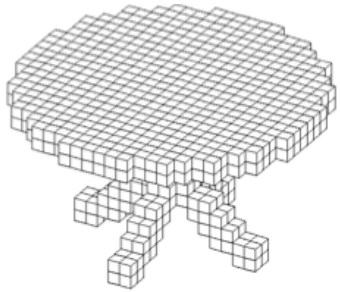
depth maps



point cloud



polygonal mesh



volume

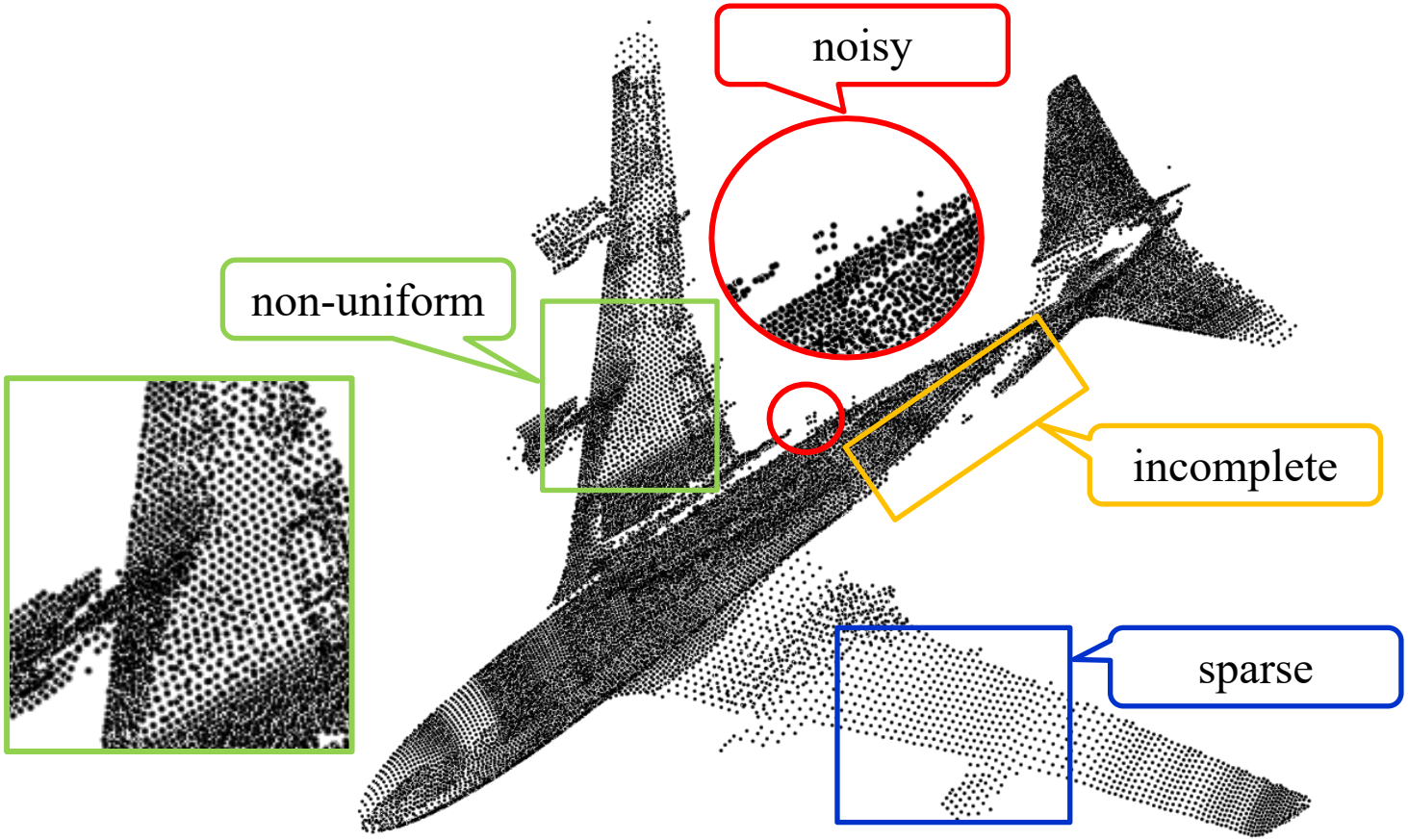
- occlusion issue

- simple & flexible
- accessible

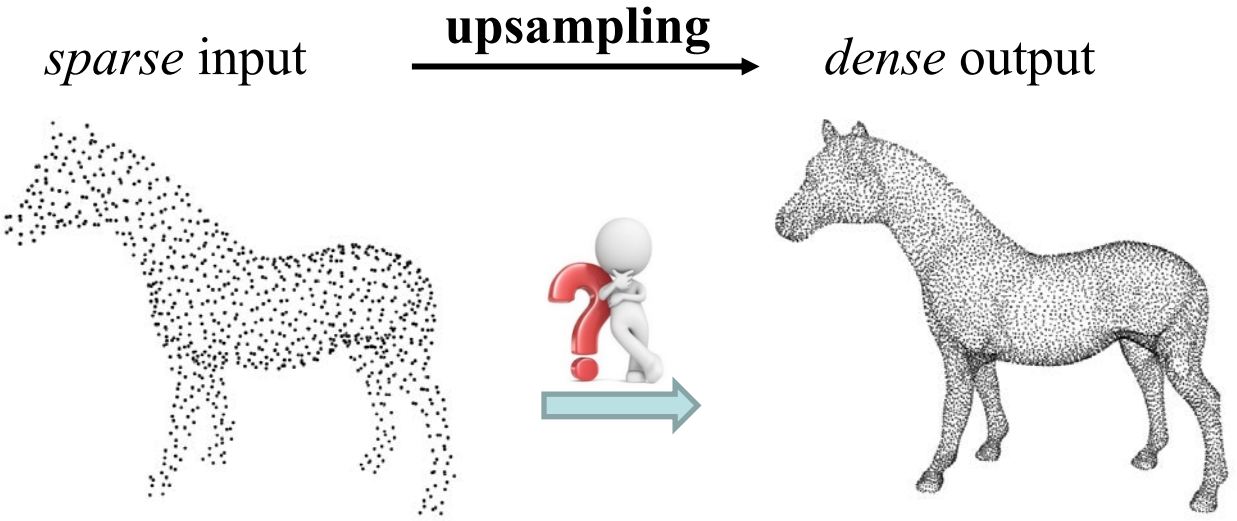
- complex

- low resolution

Real-scanned point cloud:



Point cloud upsampling:



Applications:

- Better point cloud rendering
- Be helpful for mesh reconstruction
- Improve recognition accuracy

Point cloud upsampling:

sparse input **upsampling** *dense* output

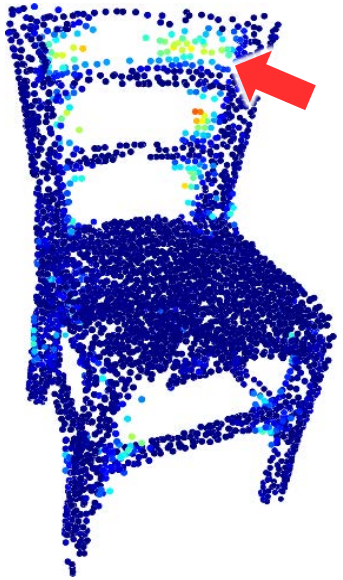
Requirements:

- Generated points should be located on the underlying surface.
- Generated points should have a uniform distribution.

Related Works

Point cloud upsampling:

- Assume that the underlying surface is smooth:
  - Interpolate points at vertices of a Voronoi diagram [1]
  - Resampling points via a locally optimal projection (LOP) [2]
  - Address the point density problem via a weighted LOP [3]
- Rely on extra geometric attributes, e.g. normal:
  - Edge-aware point set resampling [4]
  - Fast surface reconstruction via a continuous version of LOP [5]



hand-crafted features → lack of semantic information

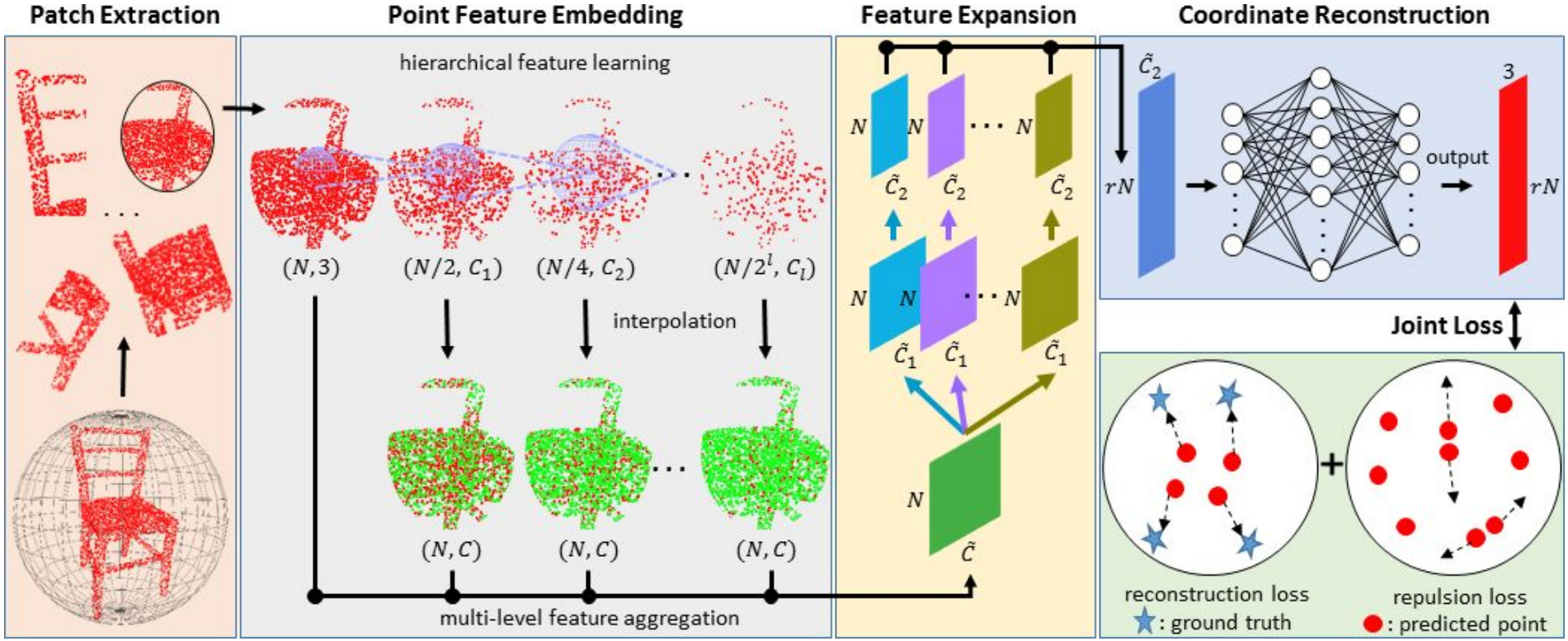
[1] M. Alexa, et al. “Computing and rendering point set surfaces.” TVCG, 2003.
[2] Y. Lipman, et al. “Parameterization-free projection for geometry reconstruction.” SIGGRAPH, 2007.
[3] H. Huang, et al. “Consolidation of unorganized point clouds for surface reconstruction.” SIGGRAPH Asia, 2009.
[4] H. Huang, et al. “Edge-aware point set resampling.” TOG, 2013.
[5] R. Preiner, et al. “Continuous projection for fast L1 reconstruction.” SIGGRAPH, 2014.

PU-Net: Point cloud Upsampling Network

CVPR，2018

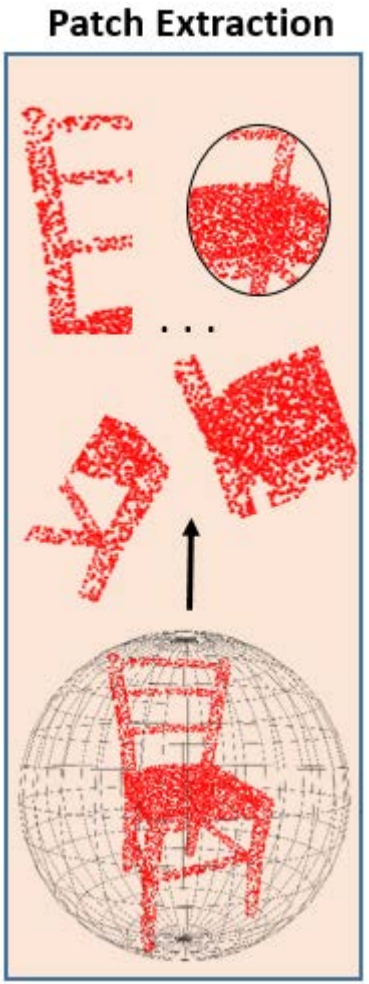
Our work: PU-Net

- How to prepare training data?
- How to expand the number of points?
- How to design loss functions to guide the network training?



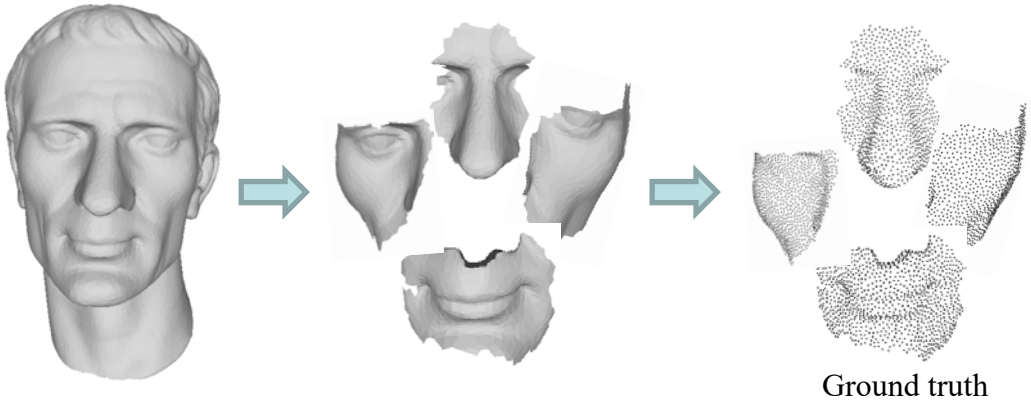
Our work: PU-Net

1. Patch Extraction



Generate ground truth:

- Randomly select M points on the surface of mesh.
- Grow a surface patch in a ring-by-ring manner.
- Poisson disk sampling to generate $\widehat{N}$ points on each patch as ground truth.



Ground truth

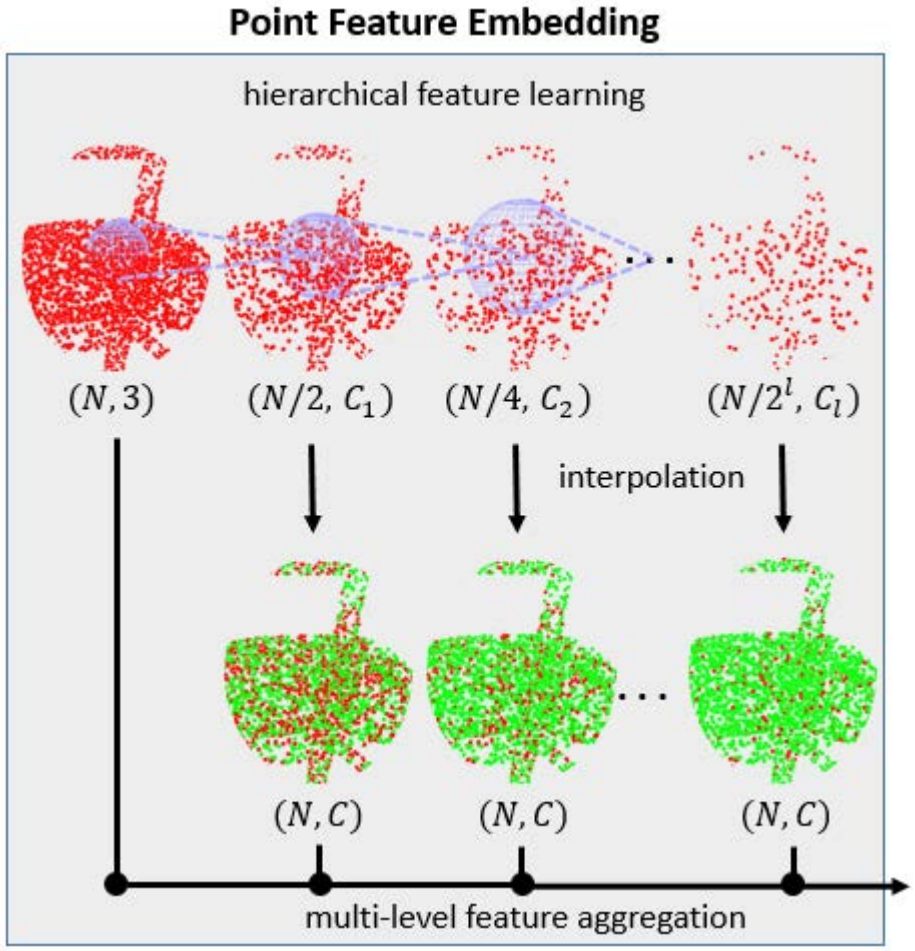
Generate input:

- No "correct pairs" of input and ground truth.
- On-the-fly input generation scheme: input points are randomly sampled from the ground truth point sets with a downsampling rate of r.

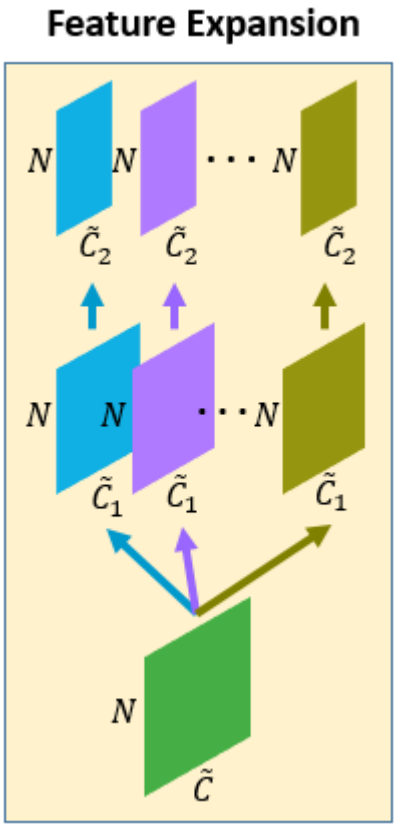
Our work: PU-Net

2. Point Feature Embedding

- Hierarchical feature learning
- Feature restoration by interpolation
  - Features of red points are extracted by hierarchical manner
  - Features of green points are interpolated using features from the nearest points.
- Multi-level feature aggregation
  - More helpful for upsampling task



3. Feature Expansion



The dimension of embedded feature f is $N \times \tilde{C}$, then the feature expansion operation can be represented as:

$$f' = \mathcal{RS}\left(\left[\mathcal{C}_1^2\left(\mathcal{C}_1^1(f)\right), \ldots, \mathcal{C}_r^2\left(\mathcal{C}_r^1(f)\right)\right]\right)$$

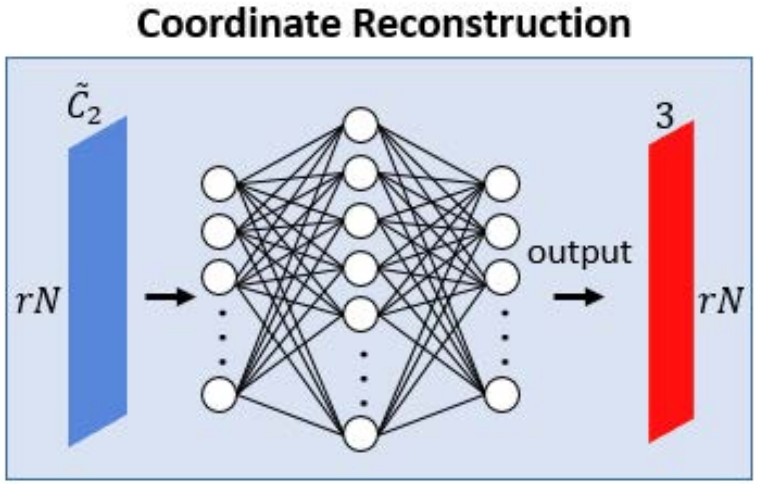
where $\mathcal{C}_i^1(\cdot)$ and $\mathcal{C}_i^2(\cdot)$ are two sets of 1×1 convolution, r is the upsampling rate, and $\mathcal{RS}(\cdot)$ is a reshape operation to convert an $N \times r\tilde{C}_2$ tensor to a tensor of size $rN \times \tilde{C}_2$

- The reason why we use two convolutions:
  Break the high correlation among the r feature sets generated from the first convolution $\mathcal{C}_i^1(\cdot)$.

Our work: PU-Net

4. Coordinate Reconstruction

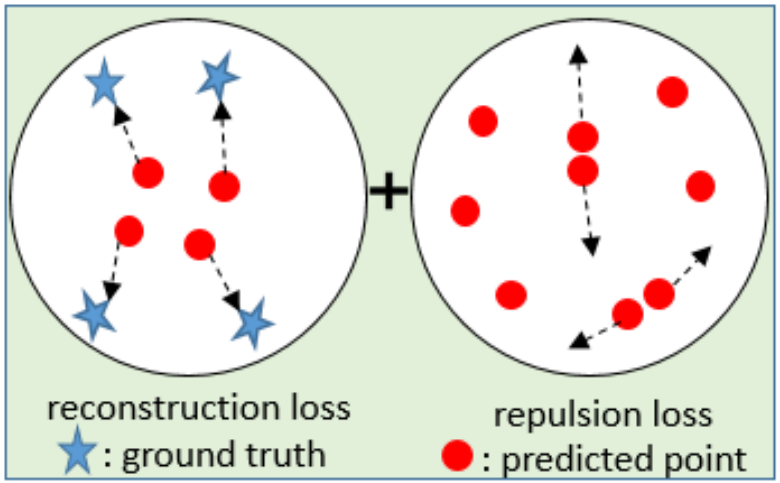
Regress the 3D coordinates via a series of fully connected layers.



Our work: PU-Net

Requirements of point cloud upsampling:

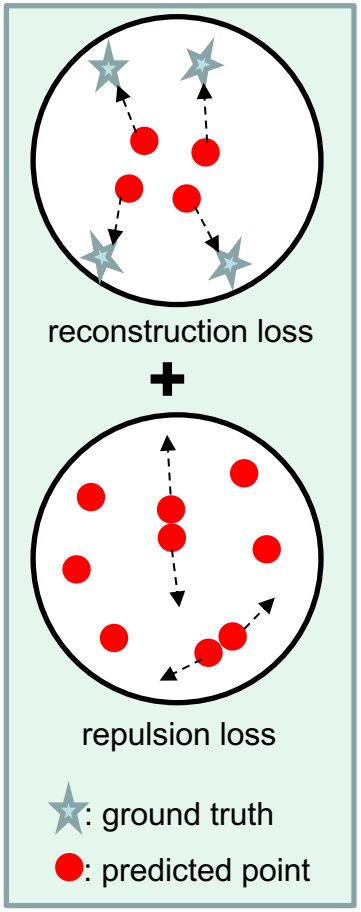
- The generated points should describe the underlying geometry surface.
- The generated point should be informative and should not clutter together.



Our work: PU-Net

Joint loss function: $L(\boldsymbol{\theta}) = L_{rec} + \alpha L_{rep} + \beta \|\boldsymbol{\theta}\|^2$

- reconstruction loss (Earth Mover's distance)
  - make the generated points locate on the underlying surface
  - $L_{rec} = d_{EMD}(S_p, S_{gt}) = \min_{\emptyset: S_p \to S_{gt}} \sum_{x_i \in S_p} \|x_i - \emptyset(x_i)\|_2$
    - S_p: predicted point; S_{gt}: ground truth point;
    - $\emptyset: S_p \to S_{gt}$ indicates the bijection mapping
- repulsion loss
  - make the generated points have a more uniform distribution
  - $L_{rep} = \sum_{i=0}^{\widehat{N}} \sum_{i' \in K(i)} \eta(\|x_{i'} - x_i\|) w(\|x_{i'} - x_i\|)$
    - $\widehat{N}$: the number of output points; $K(i)$: k-nearest neighborhood of x_i
    - repultion term: $\eta(r) = -r$
    - fast-decaying weight function: $w(r) = e^{-r^2/h^2}$



Our work: PU-Net

Datasets:

- No public benchmark dataset for point cloud upsampling
- Training:
  - collect 40 objects from Visionair repository, cut 100 patches for each object
  - Poisson disk sampling on each patch to generate $rN = 4096$ points as ground truth
  - then randomly select $N = 1024$ points on the ground truth and add Gaussian noise as input
- Testing:
  - objects from Visionair, SHREC15, ModelNet40 and ShapeNet
  - use Monte-Carlo random sampling approach to sample 5,000 points on each object as input

Our work: PU-Net

Evaluation metrics:

- To evaluate the surface deviation: *Deviation*

  1, Find the closest point $\hat{x}_i$ on the mesh for each predicted point x_i, and calculate the distance.

  2, Compute the mean and standard deviation over all the points.

- To evaluate the point uniformity: *normalized uniformity coefficient (NUC)*

  1, Put D equal-size disks on the object surface ($D = 9000$ in our experiments).

  2, Calculate the standard deviation of the number of points inside the disks.

  3, Normalize the density of each object and then compute the overall uniformity of the point sets over all the K objects in the testing dataset.

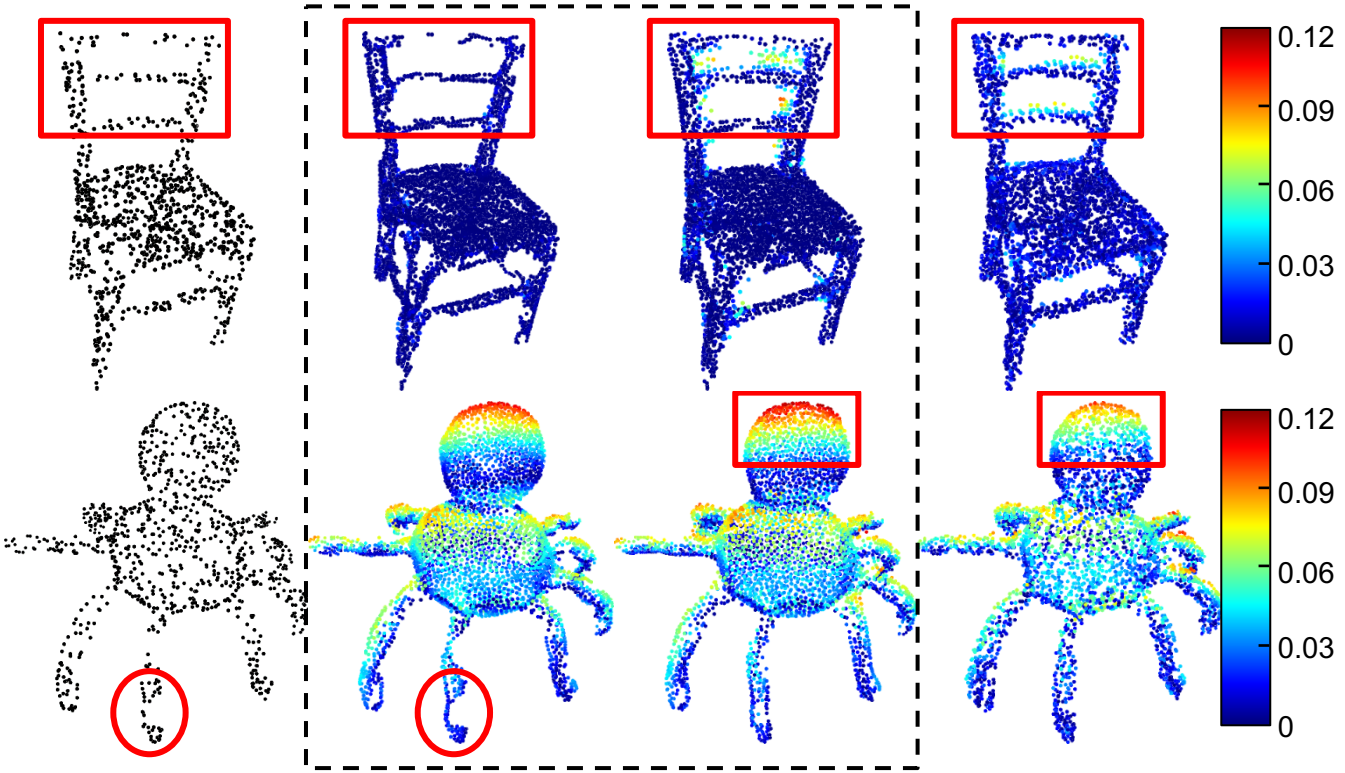
$$avg = \frac{1}{K * D} \sum_{k=1}^{K} \sum_{i=1}^{D} \frac{n_i^k}{N^k * p},$$

$$NUC = \sqrt{\frac{1}{K * D} \sum_{k=1}^{K} \sum_{i=1}^{D} (\frac{n_i^k}{N^k * p} - avg)^2},$$

N^k: the total number of points on the k-th object; n_i^k: the number of points within the i-th disk of the k-th object; p is the percentage of the disk area over the total object surface area.

Our work: PU-Net

Comparison with the optimization-based method:

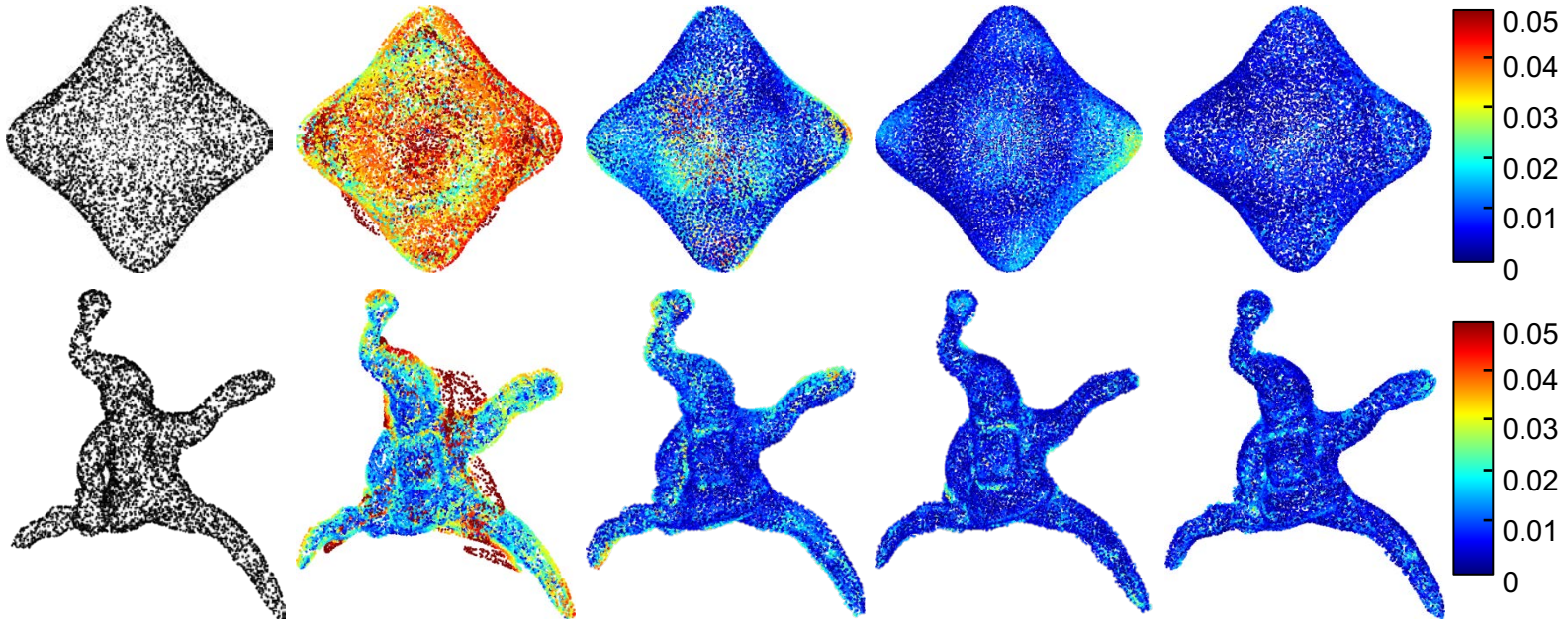


Visual comparisons with EAR method [1]. We color-code all points to show the deviation from ground truth mesh.

[1] H. Huang, et al. “Edge-aware point set resampling.” TOG, 2013.

Our work: PU-Net

Comparison with deep learning-based baselines:



Visual comparisons with deep learning-based baselines. We modify the original point cloud recognition networks by using our feature expansion module and the loss functions. The colors on points reveal the surface distance errors, where blue indicates low error and red indicates high error.

[1] Charles R. Qi, et al. "PointNet: deep learning on point sets for 3D classification and segmentation." CVPR, 2017.
[2] Charles R. Qi, et al. "PointNet++: deep hierarchical feature learning on point sets in a metric space." NIPS, 2017.

Our work: PU-Net

Comparison with deep learning-based baselines :

Table 1. Quantitative comparison on our collected dataset.

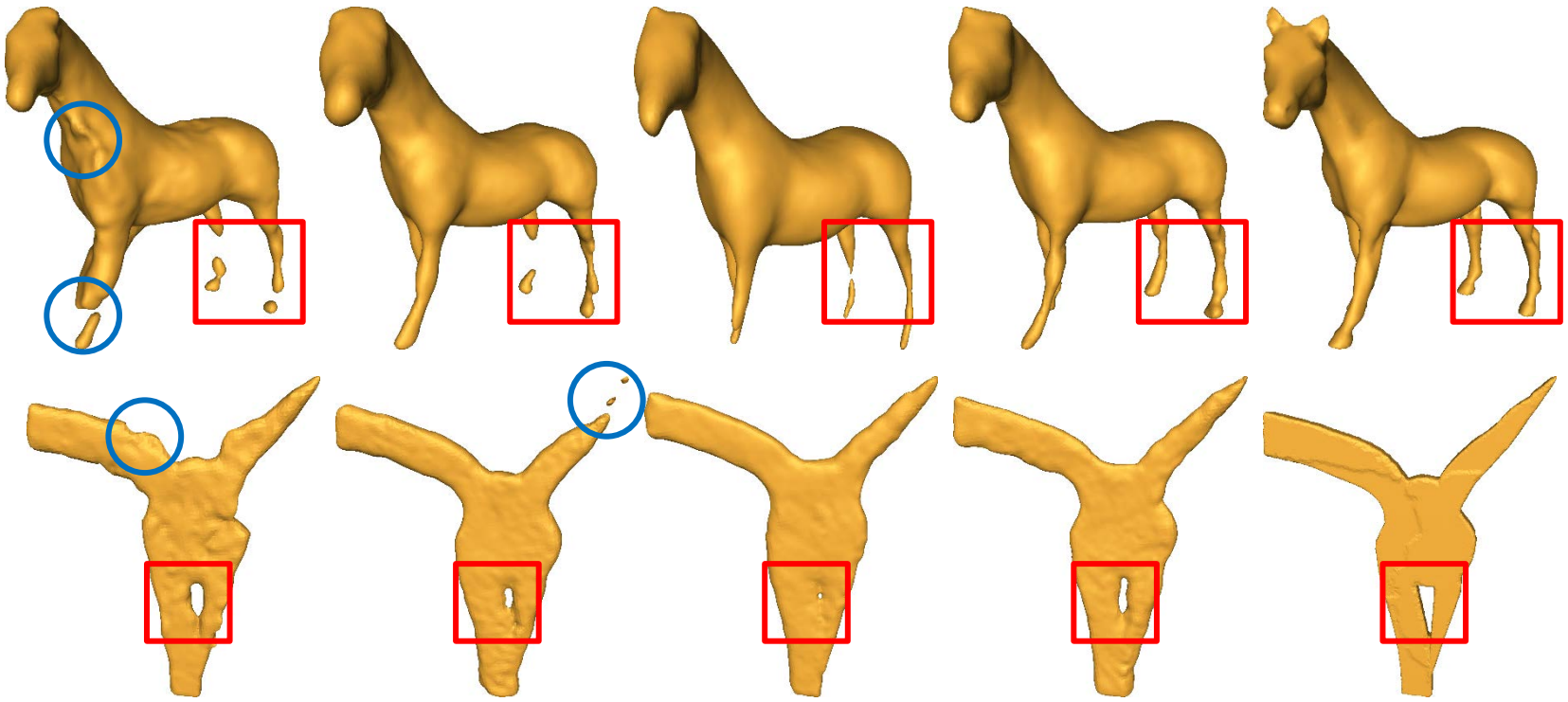
Method	NUC with different p						Deviation (10^{-2})		Time (s)
	0.2%	0.4%	0.6%	0.8%	1.0%	1.2%	mean	std	
Input	0.316	0.224	0.185	0.164	0.150	0.142	-	-	-
PointNet	0.409	0.334	0.295	0.270	0.252	0.239	2.27	3.18	**0.05**
PointNet++	0.215	0.178	0.160	0.150	0.143	0.139	1.01	0.83	0.16
PointNet++(MSG)	0.208	0.169	0.152	0.143	0.137	0.134	0.78	0.61	0.45
PU-Net (Ours)	**0.174**	**0.138**	**0.122**	**0.115**	**0.112**	**0.110**	**0.63**	**0.53**	0.15

Table 2. Quantitative comparison on SHREC15 dataset.

Method	NUC with different p						Deviation (10^{-2})		Time (s)
	0.2%	0.4%	0.6%	0.8%	1.0%	1.2%	mean	std	
Input	0.315	0.222	0.184	0.165	0.153	0.146	-	-	-
PointNet	0.490	0.405	0.360	0.330	0.309	0.293	2.03	2.94	**0.05**
PointNet++	0.259	0.218	0.197	0.185	0.176	0.170	0.88	0.75	0.16
PointNet++(MSG)	0.250	0.199	0.175	0.161	0.153	0.148	0.69	0.59	0.45
PU-Net (Ours)	**0.219**	**0.173**	**0.154**	**0.144**	**0.140**	**0.137**	**0.52**	**0.45**	0.15

Our work: PU-Net

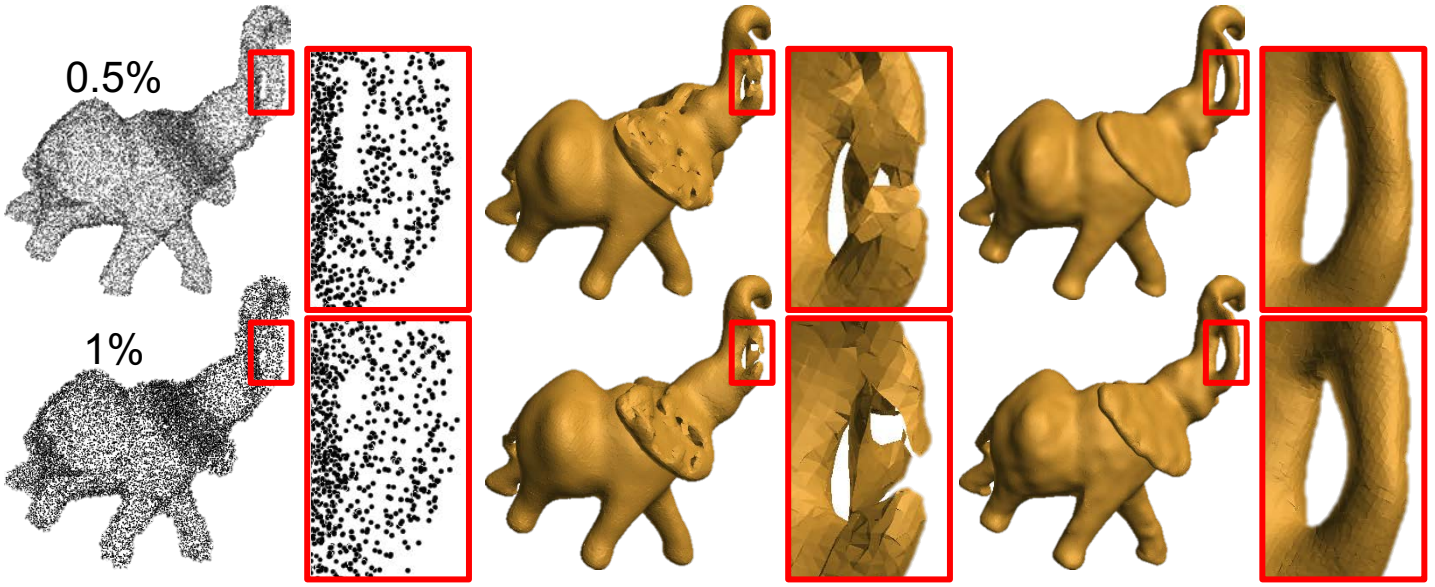
Results of surface reconstruction:



Surface reconstruction results from the upsampled point clouds.

Our work: PU-Net

Robustness to noise:

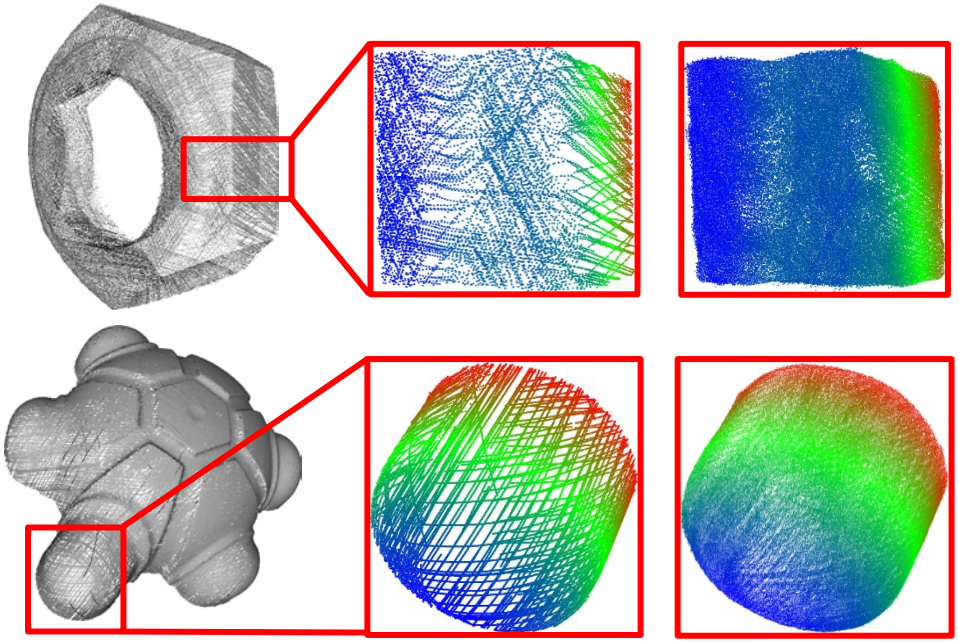


(a) inputs: noisy point clouds (b) reconstructed directly from inputs (c) reconstructed from network outputs

Input sparse point sets are contaminated by different level of Gaussian noise. Surface reconstruction results show that our upsampling method is robust to noise.

Our work: PU-Net

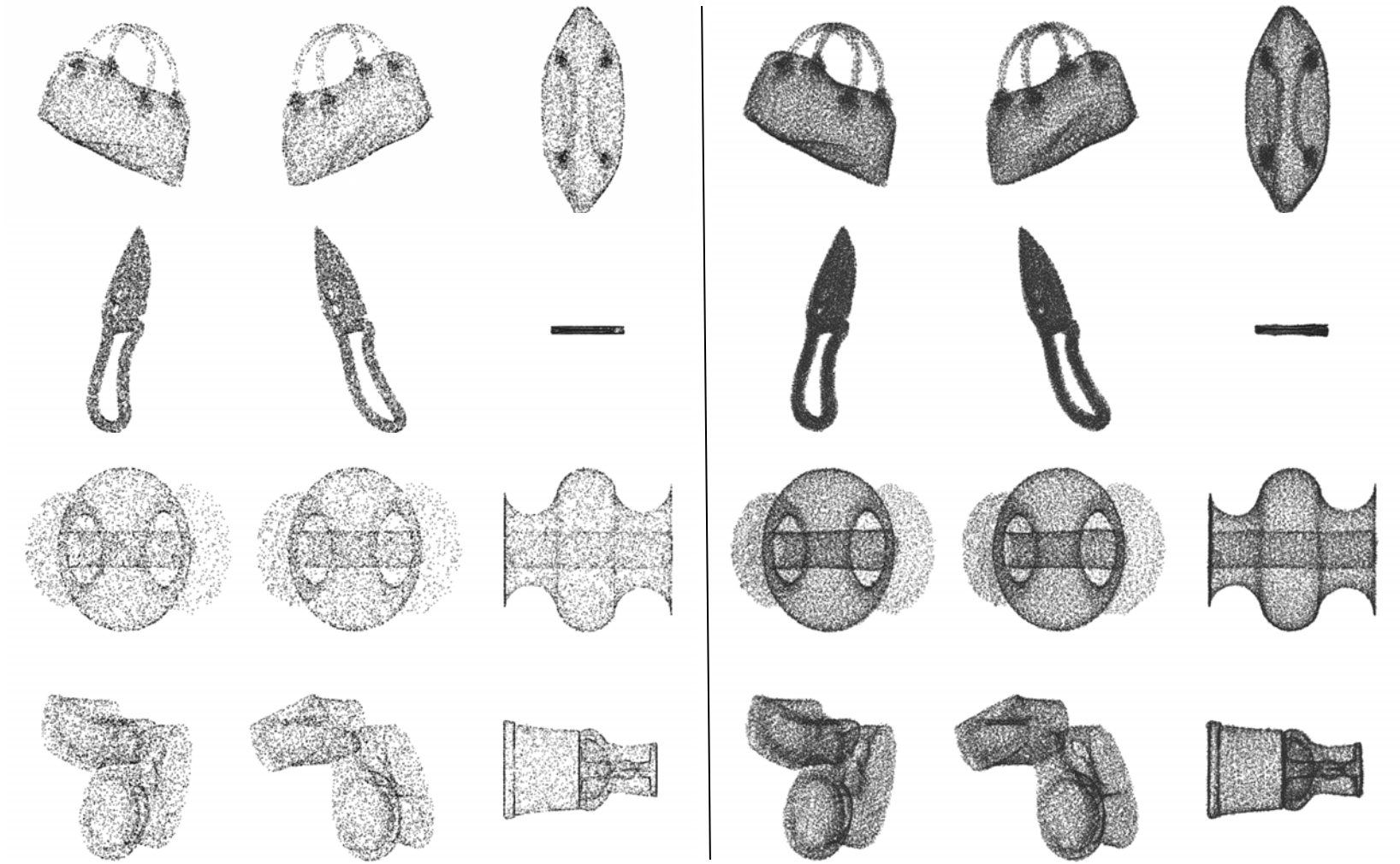
Results on real-scanned models:



Results on real scanned point clouds. We color-code input patches and upsampling results to show the depth information. Blue points are more close to us.

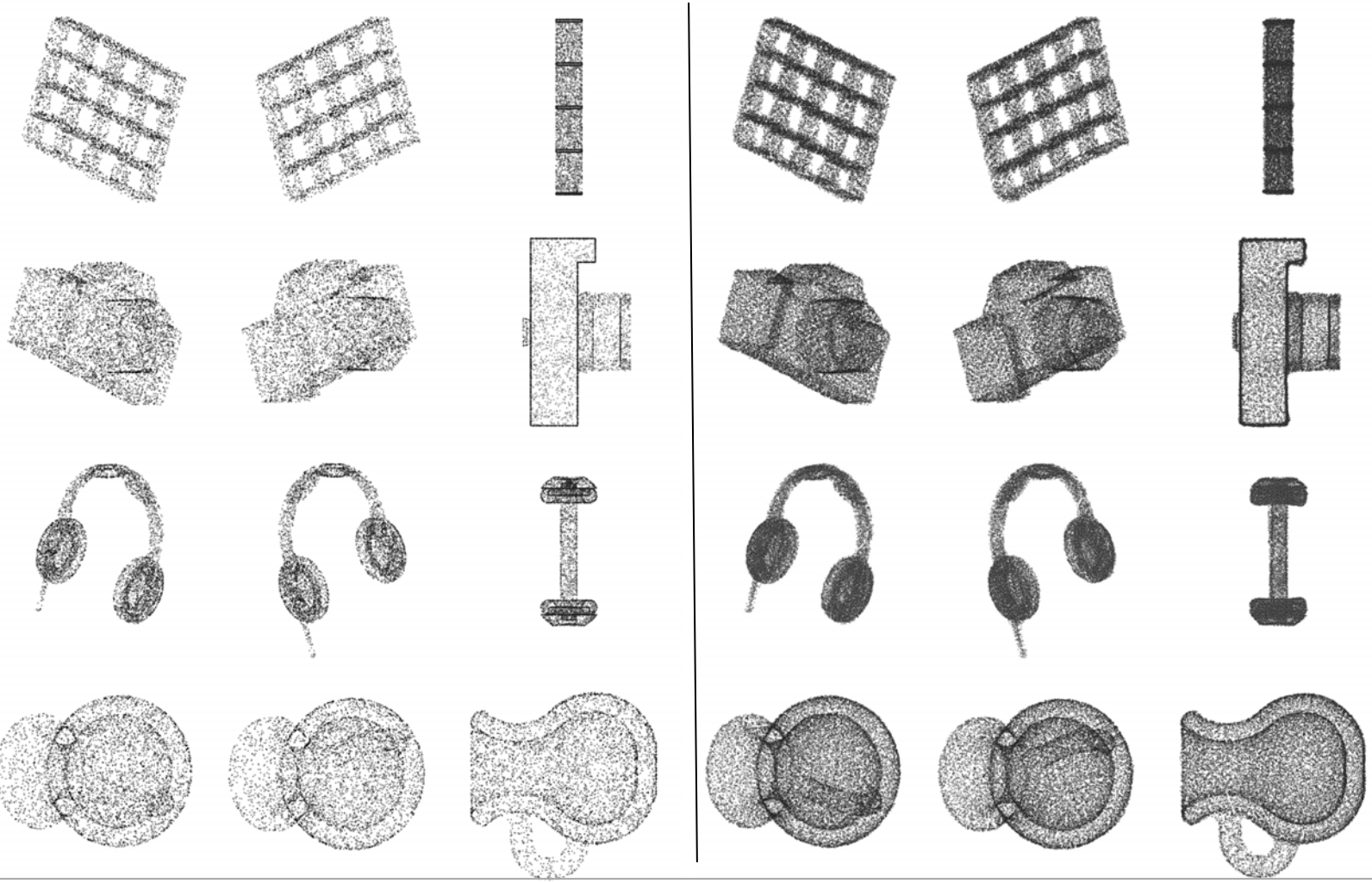
Our work: PU-Net

More results on ModelNet40 dataset:

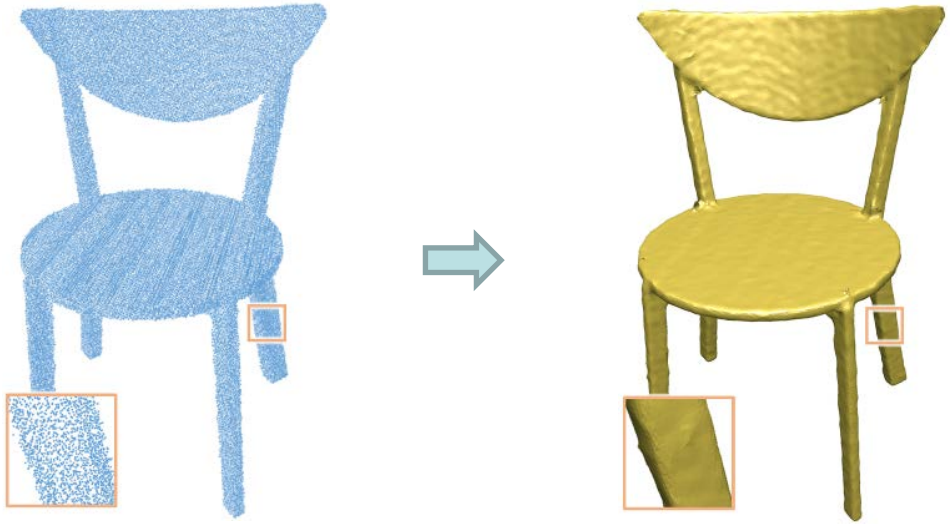


Our work: PU-Net

More results on ModelNet40 dataset :



Upsampling problems are typically more severe near sharp edges!



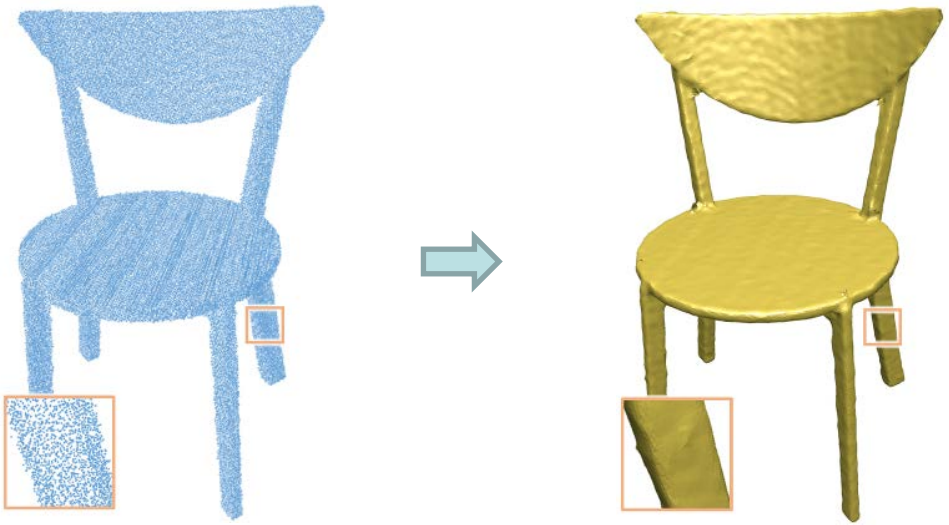
input

reconstructed from PU-Net

EC-Net: an Edge-aware Point Set Consolidation Network

ECCV，2018

Upsampling problems are typically more severe near sharp edges



input

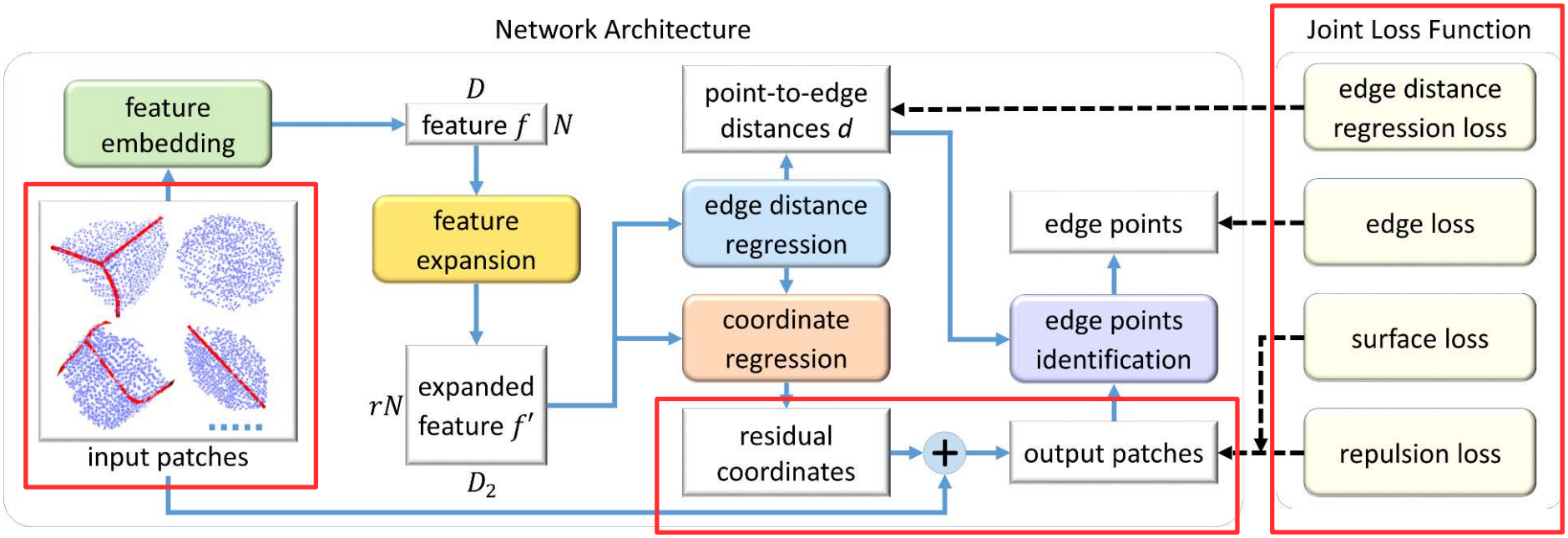
reconstructed from PU-Net

Edge-aware Point Cloud Upsampling Network (EC-Net):

- ✓ Upsample points
- ✓ Detect edge points
- ✓ Arrange more points on edges

Our work: EC-Net

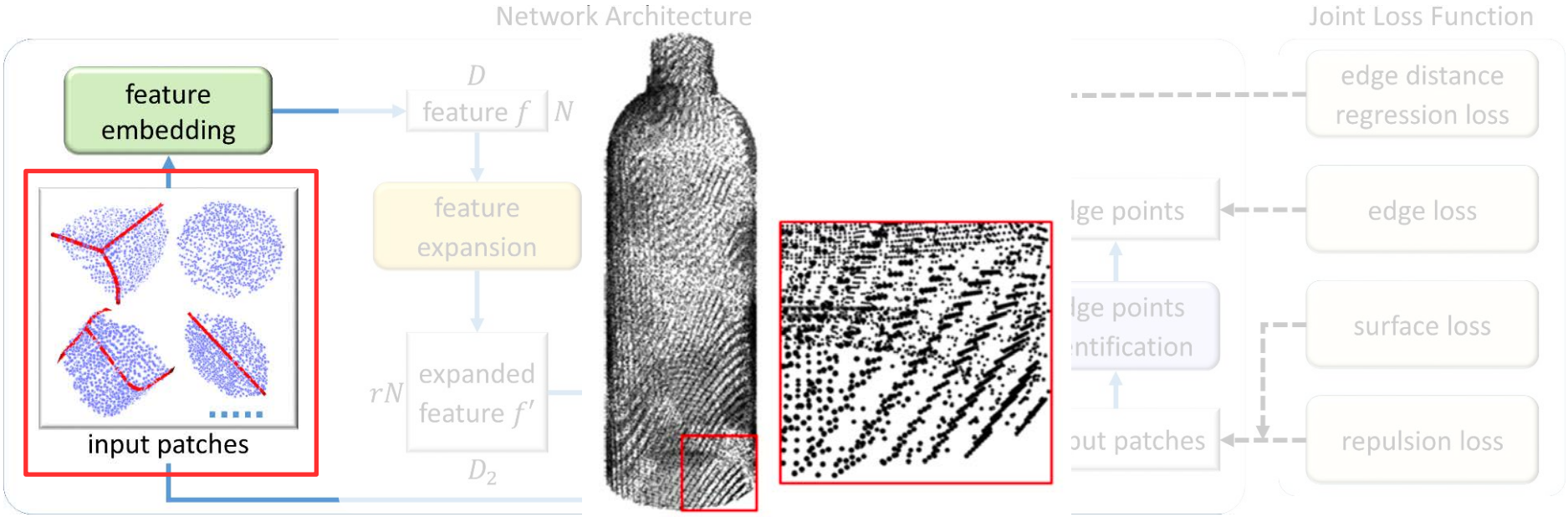
Edge-aware Point Cloud Upsampling Network (EC-Net):



- Training data preparation
- Point coordinate regression
- Joint loss function

Our work: EC-Net

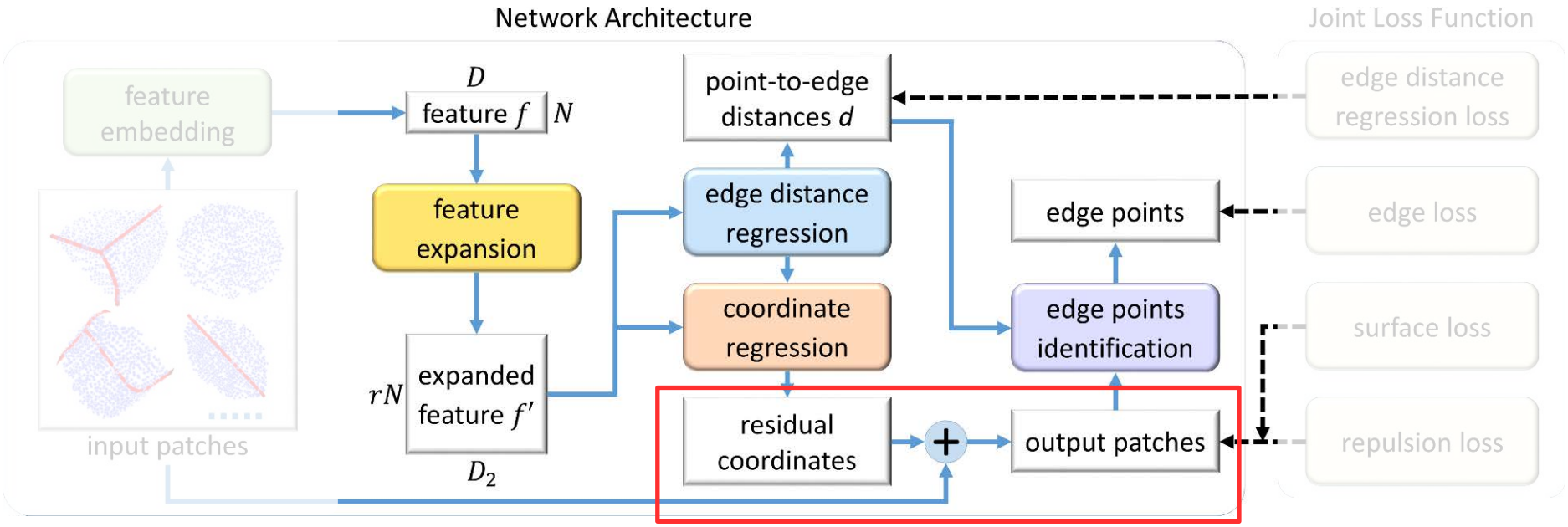
Edge-aware Point Cloud Upsampling Network (EC-Net):



- Training data preparation: *virtual scanning* to generate points from meshes, rather than sampling
  (1) Put virtual camera
  (2) Generate depth map, add quantization noise
  (3) Back-projection

Our work: EC-Net

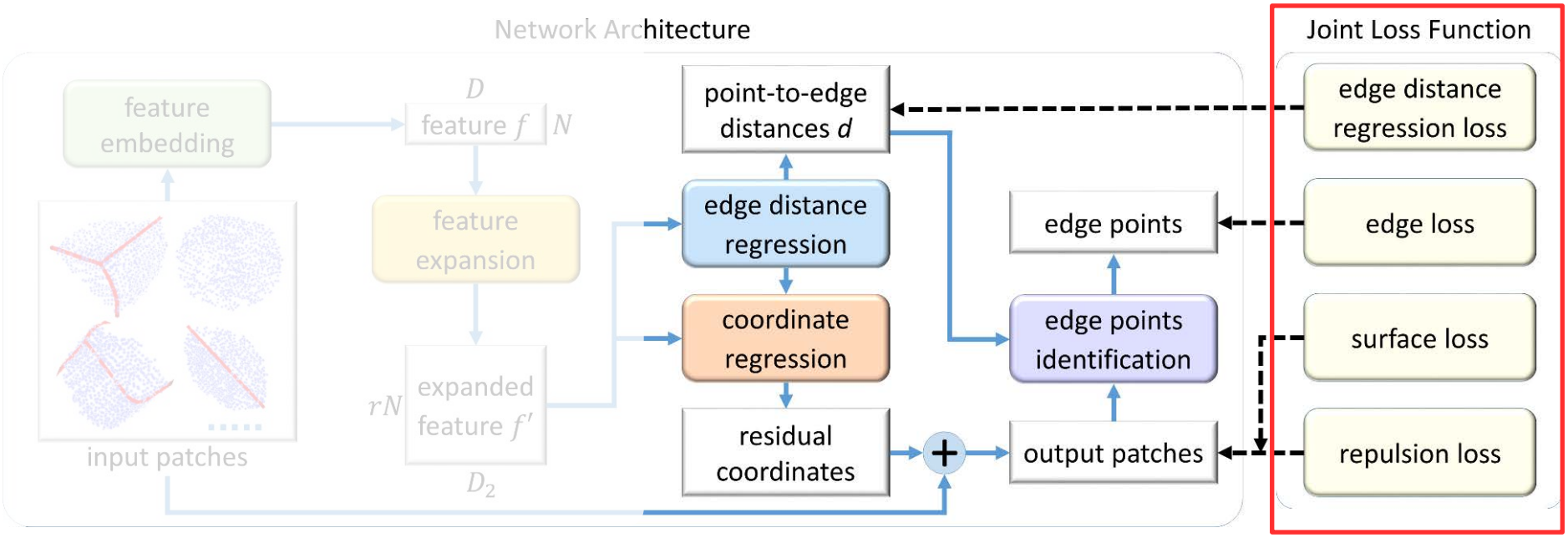
Edge-aware Point Cloud Upsampling Network (EC-Net):



- Point coordinate regression: regress *residual* coordinates, rather than directly regress point coordinates

Our work: EC-Net

Edge-aware Point Cloud Upsampling Network (EC-Net):

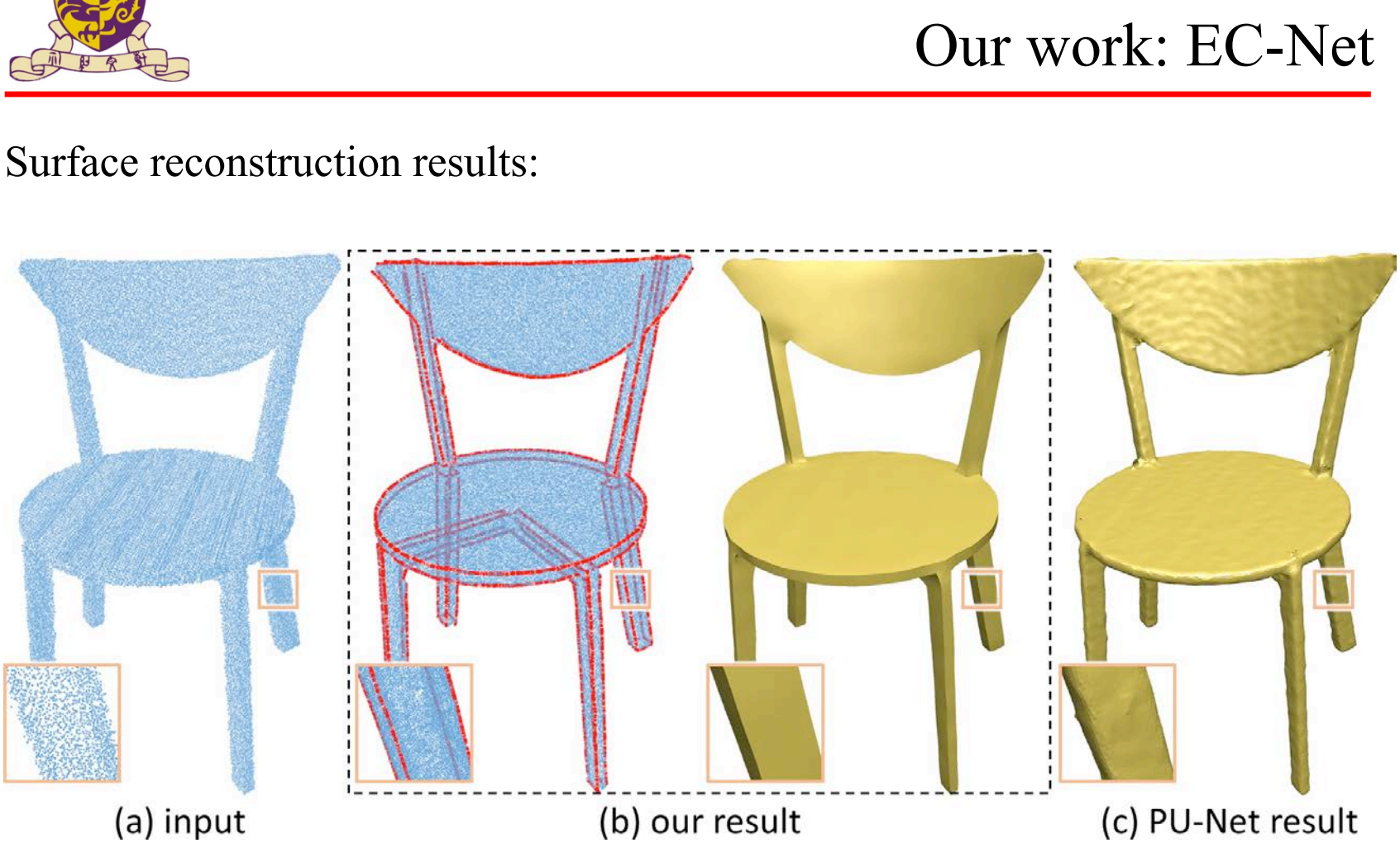


- Joint loss function: further propose the *edge distance regression loss* & *edge loss*

Our work: EC-Net

Edge-aware Joint Loss Function:

- *Repulsion loss*: the same as that in PU-Net
- *Surface loss*:
  - $L_{surf} = \frac{1}{\widetilde{N}} \sum_{1 \le i \le \widetilde{N}} d_T^2(x_i, T)$ where $\widetilde{N}$ is the number of output points
  - $d_T^2(x_i, T) = \min_{t \in T} d_t(x_i, t)$ is the minimum shortest distance from each point x_i to all the mesh triangles T
- *Edge distance regression loss*: regress point-to-edge distance
  - $L_{regr} = \frac{1}{\widetilde{N}} \sum_{1 \le i \le \widetilde{N}} \left[\Gamma_b \left(d_E^2(x_i, E) \right) - \Gamma_b(d_i) \right]^2$, where $\Gamma_b(x) = \max(0, \min(x, b))$
- *Edge loss*: encourage detected edge points locating along edges
  - $L_{edge} = \frac{1}{\widetilde{N}_{edge}} \sum_{1 \le i \le \widetilde{N}_{edge}} d_E^2(x_i, E)$ where $\widetilde{N}_{edge}$ is the number of edge points
  - $d_E^2(x_i, E) = \min_{e \in E} d_e(x_i, e)$ is the minimum shortest distance from each point x_i to all the edge segments E



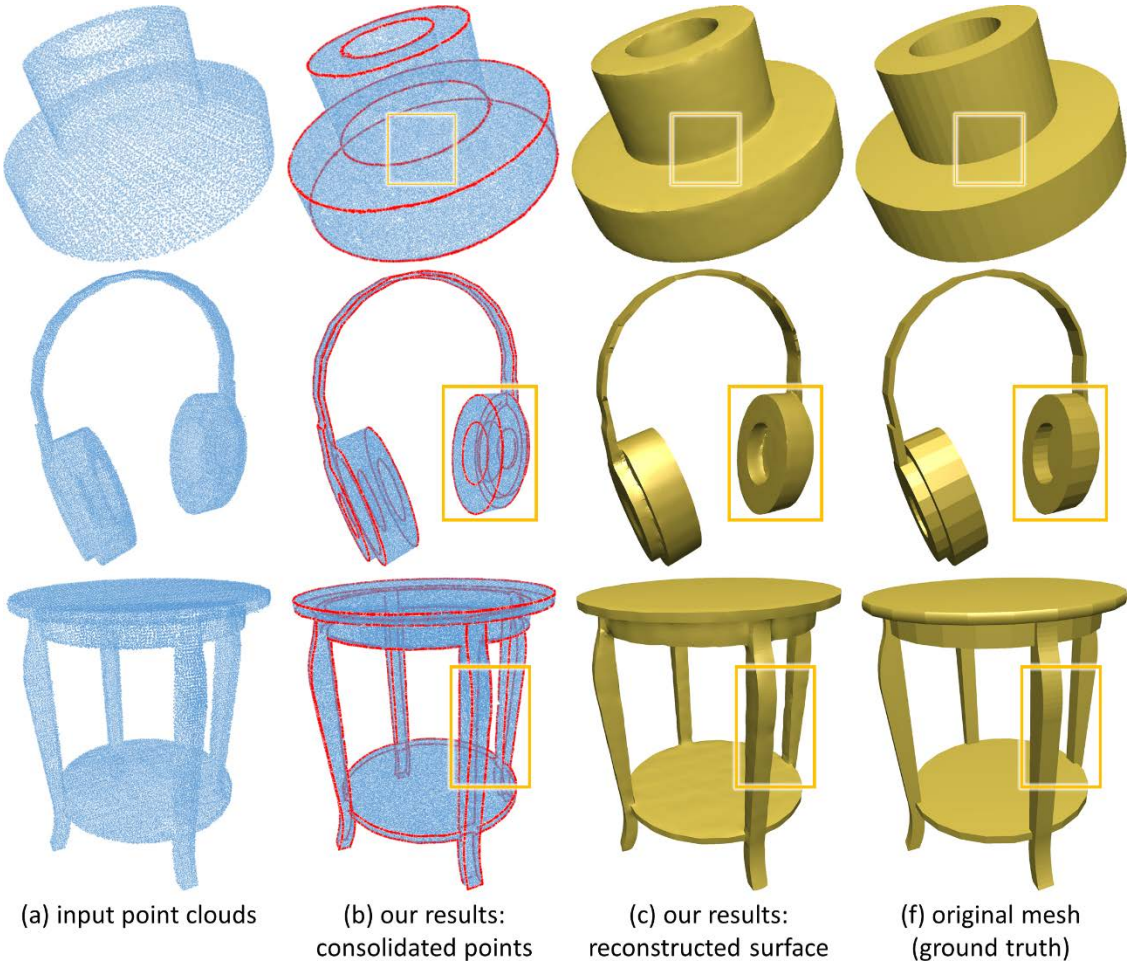
Our work: EC-Net

Surface reconstruction results:

(a) input (b) our result (c) PU-Net result

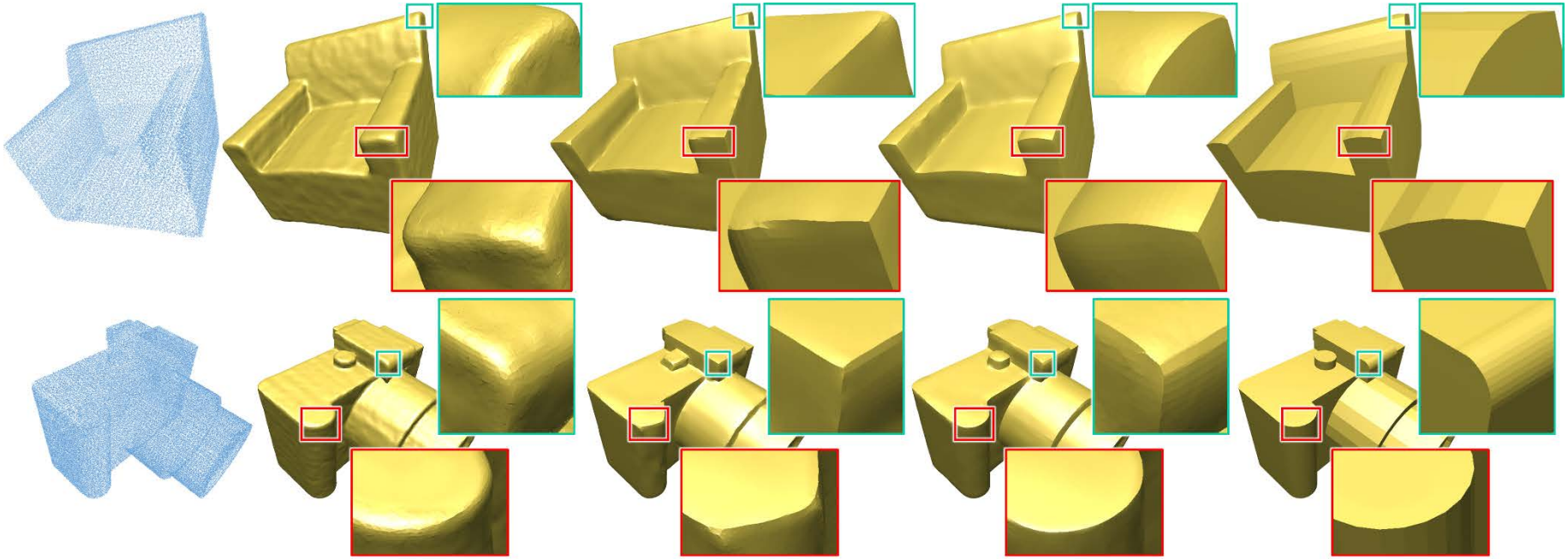
Our work: EC-Net

Surface reconstruction results:

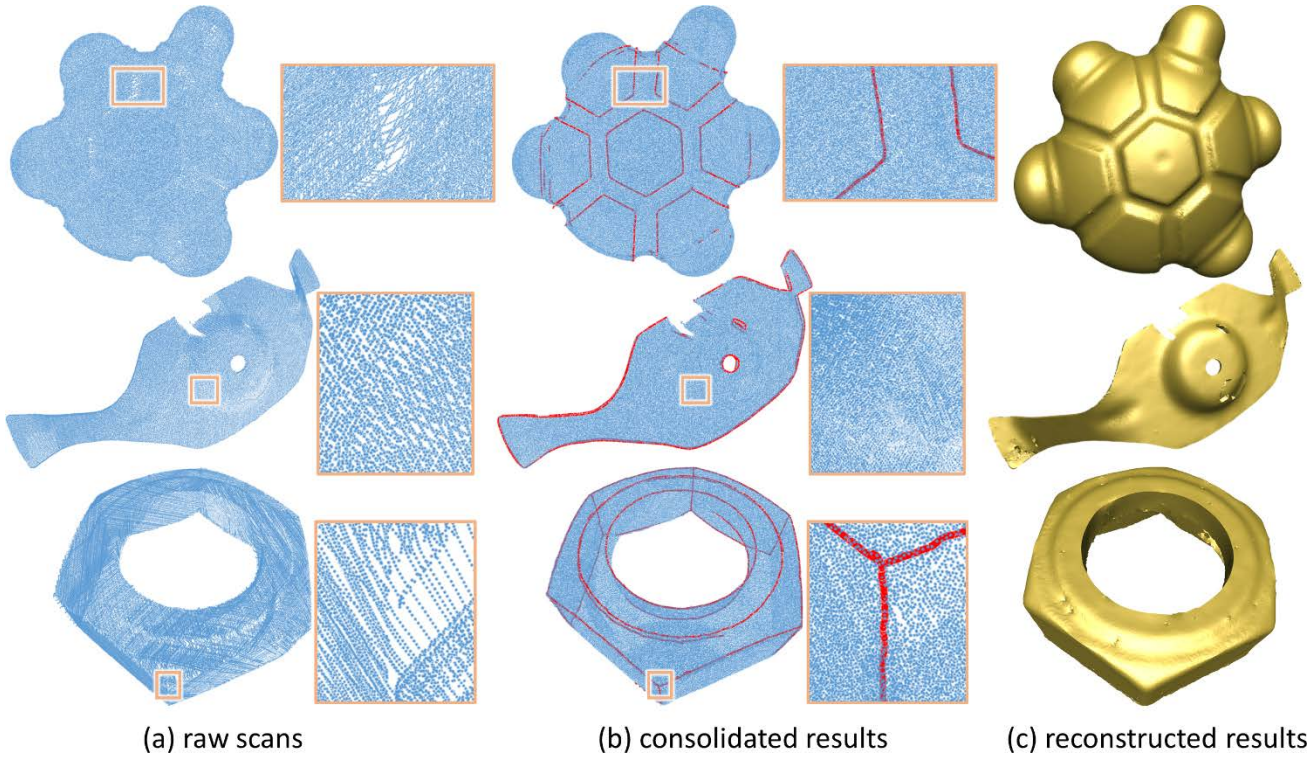


Our work: EC-Net

Comparison with other methods:



(a) input point cloud (b) PU-Net result (c) EAR result (d) our result (e) original mesh



Our work: EC-Net

Results on real scans:

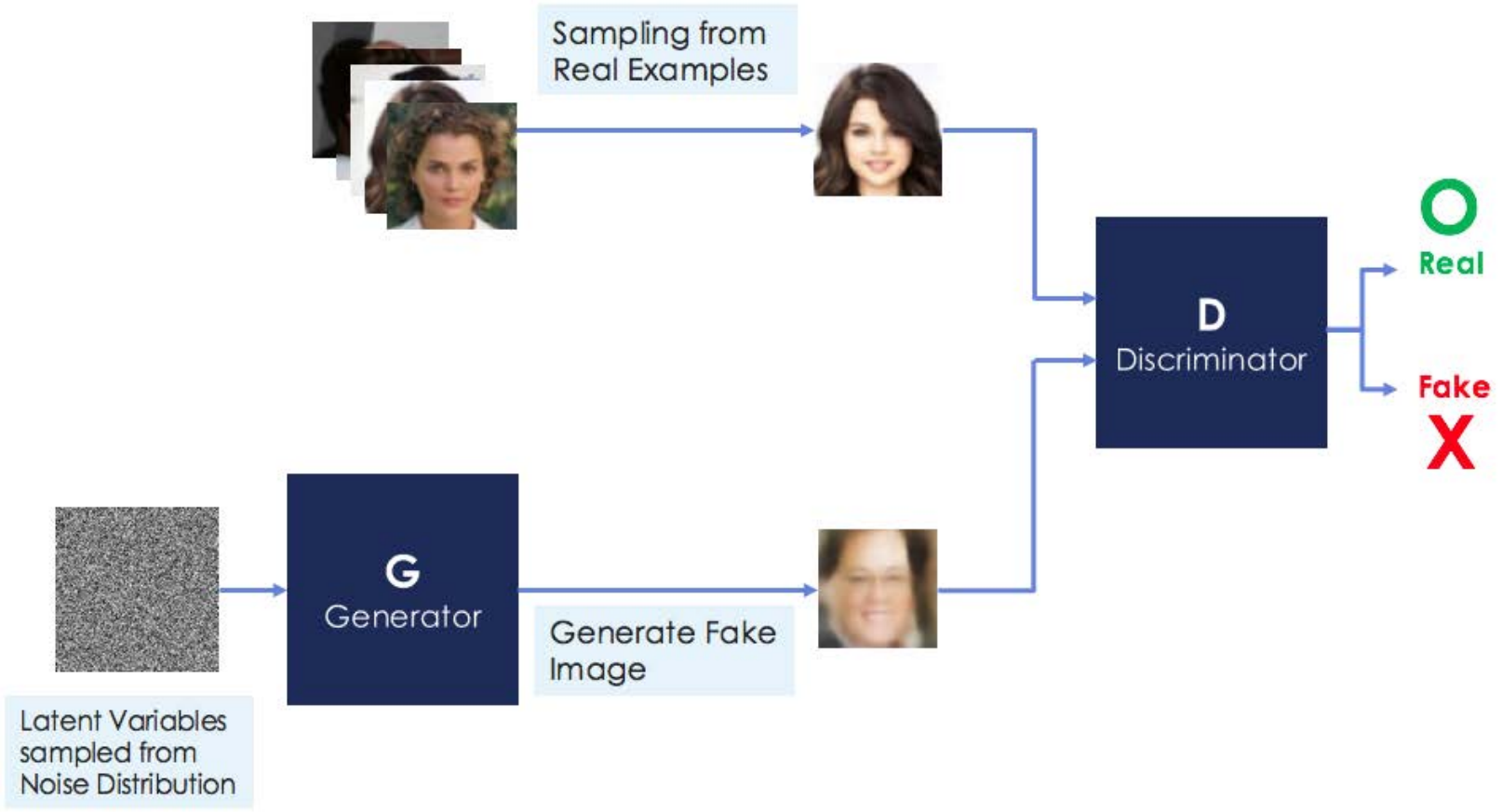
(a) raw scans (b) consolidated results (c) reconstructed results

PU-GAN: a Point Cloud Upsampling Adversarial Network

ICCV，2019

Our work: PU-GAN

Generative adversarial nets (GAN) [1]:



[1] Goolfellow, Ian, Pouget-Abadie, Jean, et al. "Generative adversarial nets." NIPS, 2014.

Our work: PU-GAN

Applications of GANs:

Style-transfer [1]

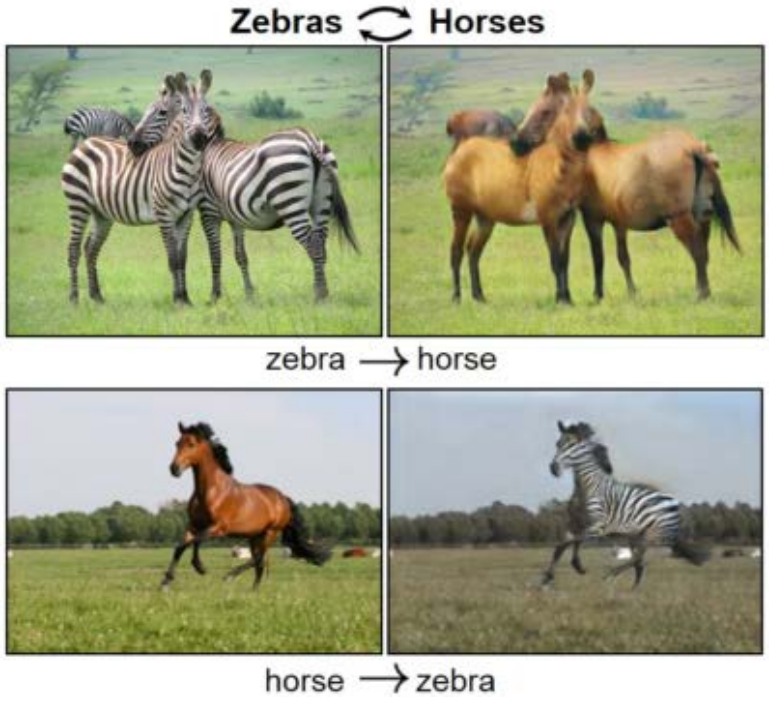
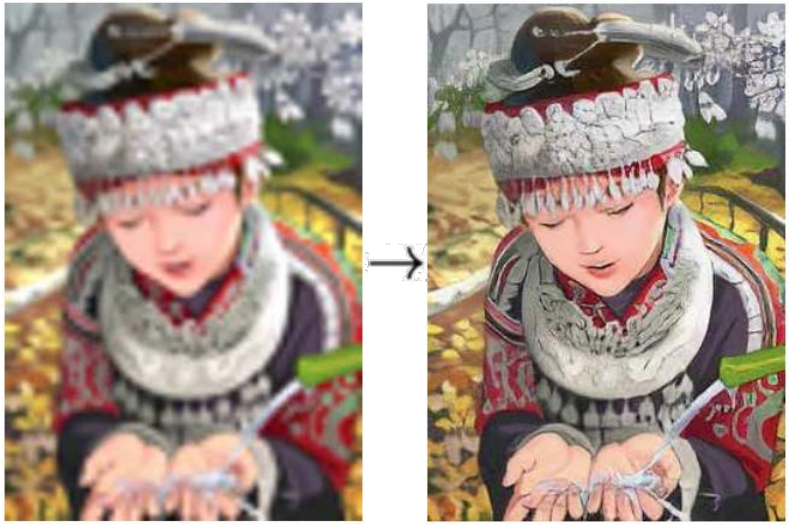


Image super-resolution [2]

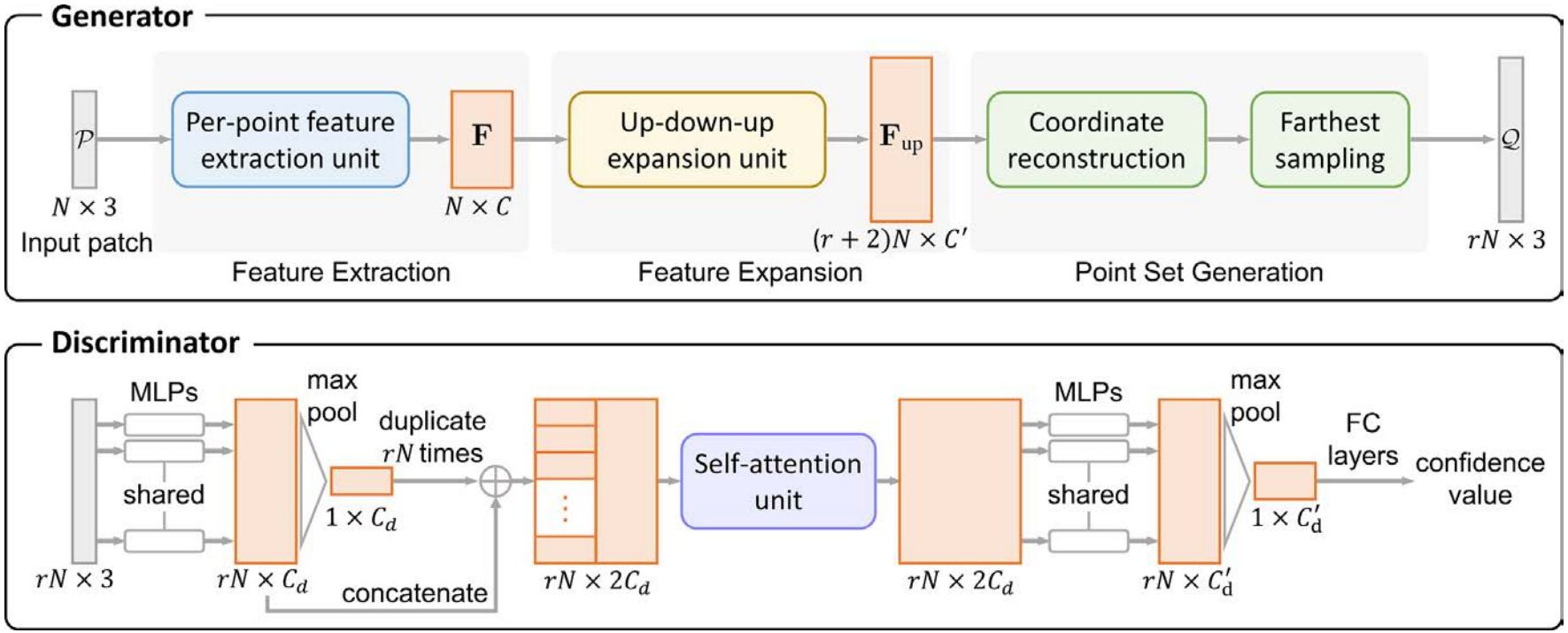


.....

[1] P. Isola, et al. “Image-to-image translation with conditional adversarial networks.” CVPR 2017.
[2] C. Ledig, et al. “Photo-realistic single image super-resolution using a generative adversarial network.” CVPR 2017.

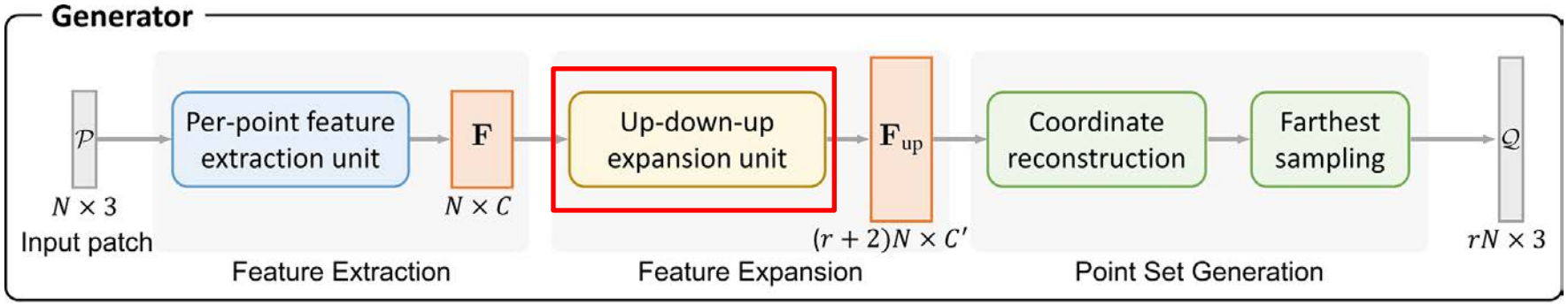
Our work: PU-GAN

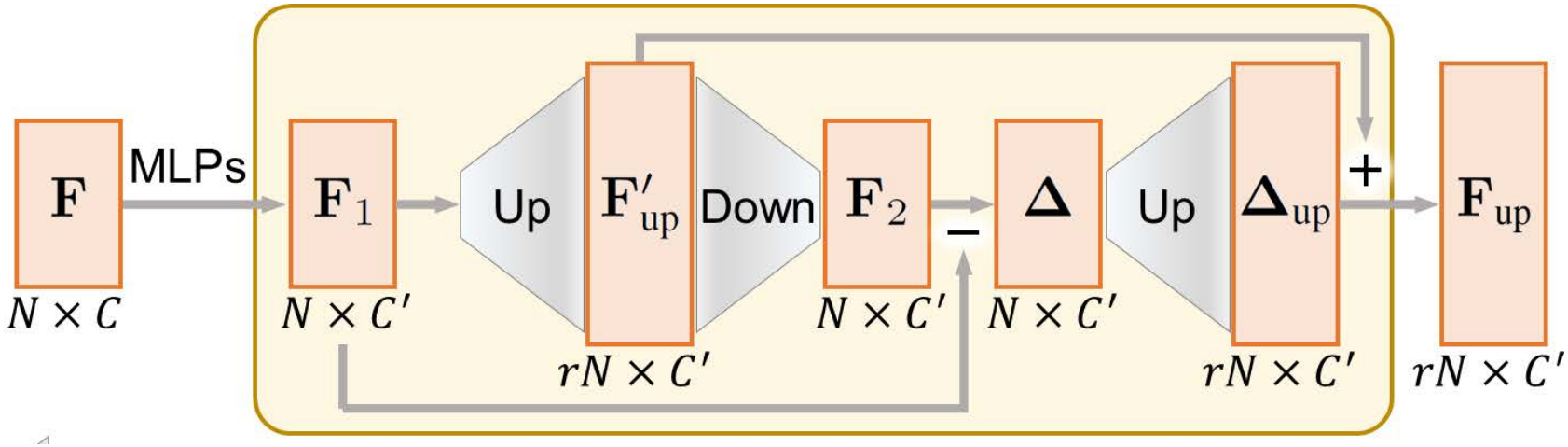
Point cloud upsampling adversarial network (PU-GAN):



Our work: PU-GAN

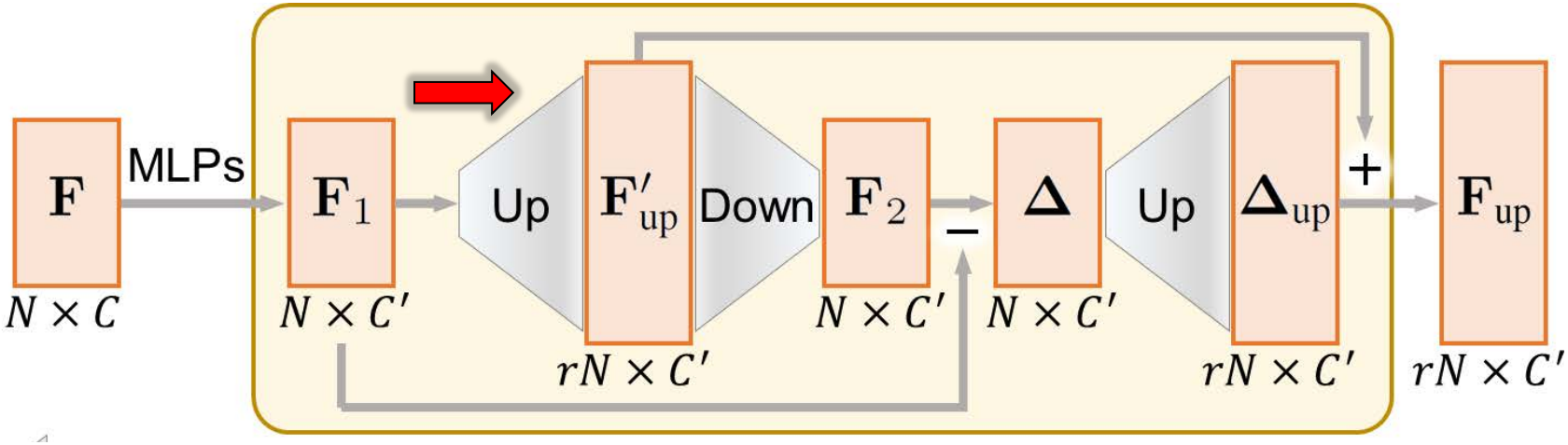
Point cloud upsampling adversarial network (PU-GAN):





Our work: PU-GAN

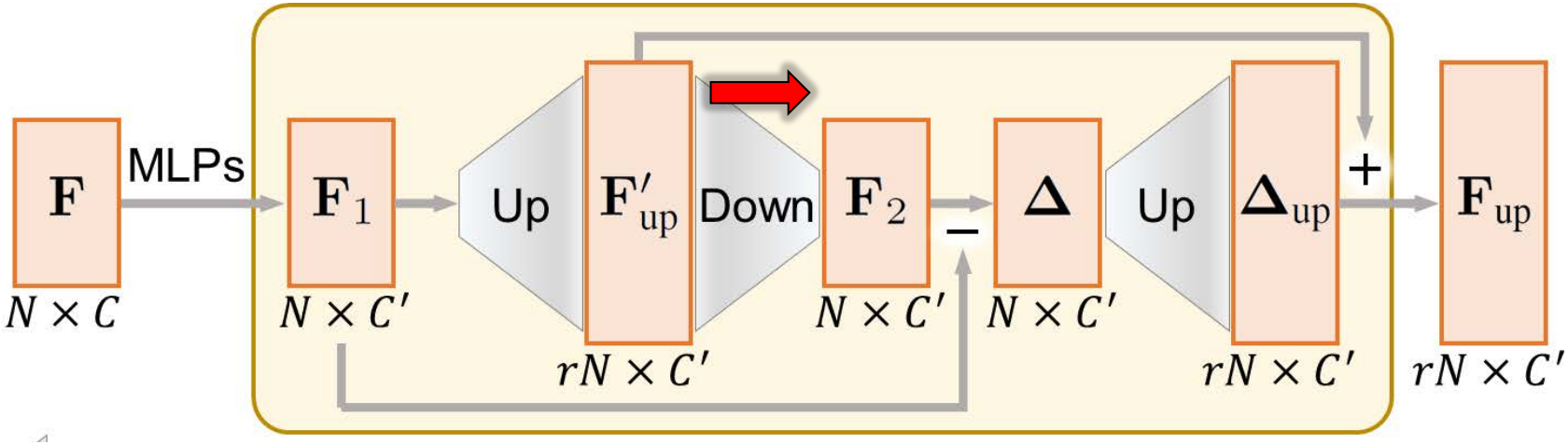
Up-down-up expansion unit:



scale up: $\mathbf{F}'_{\text{up}} = \text{Up}(\mathbf{F}_1)$

Our work: PU-GAN

Up-down-up expansion unit:

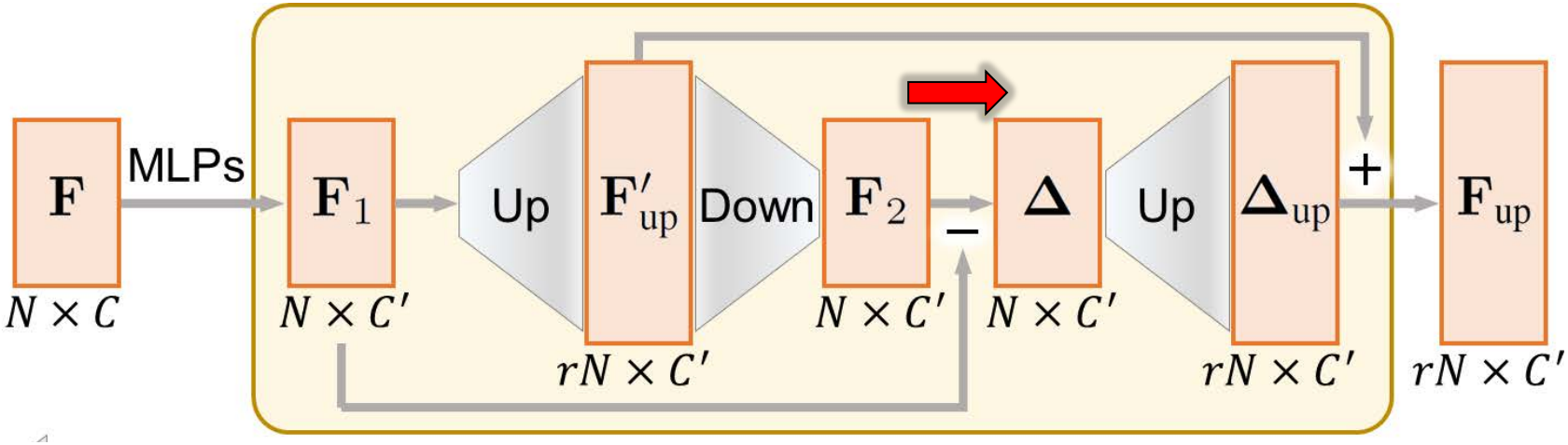


$$\text{scale up:} \quad \mathbf{F}'_{up} = \text{Up}(\mathbf{F}_1)$$

$$\text{scale down:} \quad \mathbf{F}_2 = \text{Down}(\mathbf{F}'_{up})$$

Our work: PU-GAN

Up-down-up expansion unit:



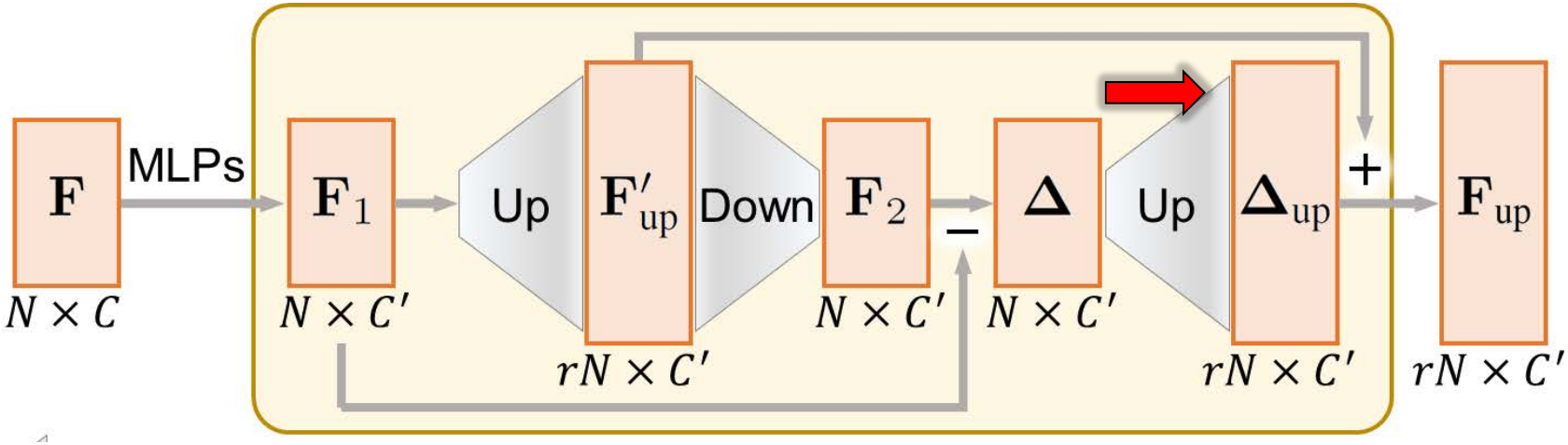
$$\text{scale up:} \quad \mathbf{F}'_{\text{up}} = \text{Up}(\mathbf{F}_1)$$

$$\text{scale down:} \quad \mathbf{F}_2 = \text{Down}(\mathbf{F}'_{\text{up}})$$

$$\text{residual:} \quad \Delta = \mathbf{F}_2 - \mathbf{F}_1$$

Our work: PU-GAN

Up-down-up expansion unit:



$$\text{scale up:} \quad \mathbf{F}'_{\text{up}} = \text{Up}(\mathbf{F}_1)$$

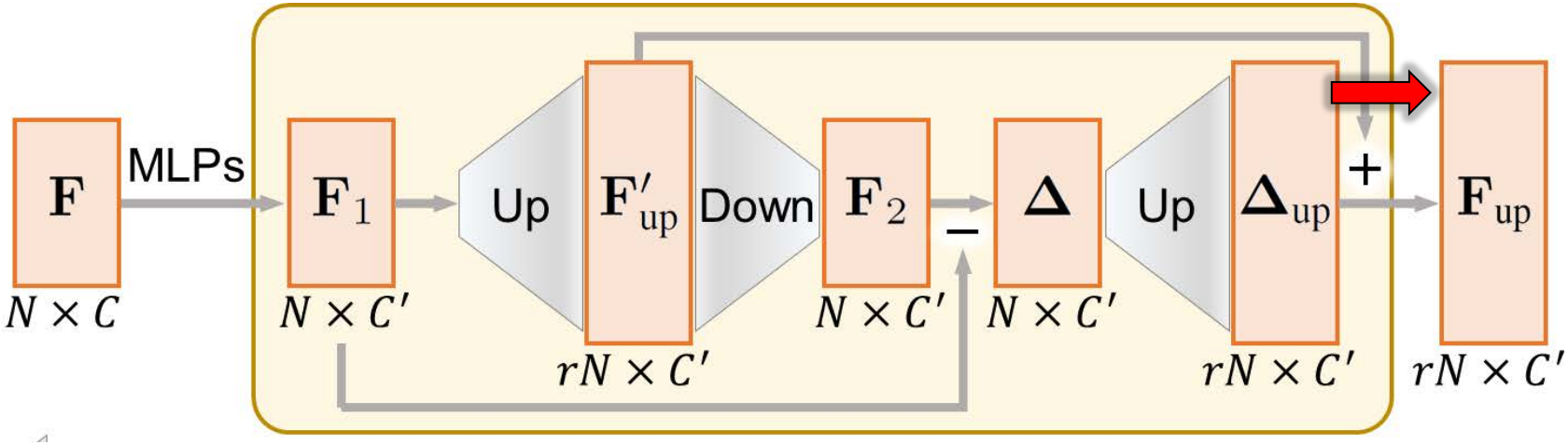
$$\text{scale down:} \quad \mathbf{F}_2 = \text{Down}(\mathbf{F}'_{\text{up}})$$

$$\text{residual:} \quad \Delta = \mathbf{F}_2 - \mathbf{F}_1$$

$$\text{scale residual up:} \quad \Delta_{\text{up}} = \text{Up}(\Delta)$$

Our work: PU-GAN

Up-down-up expansion unit:



$$\text{scale up:} \quad \mathbf{F}'_{\text{up}} = \text{Up}(\mathbf{F}_1)$$

$$\text{scale down:} \quad \mathbf{F}_2 = \text{Down}(\mathbf{F}'_{\text{up}})$$

$$\text{residual:} \quad \Delta = \mathbf{F}_2 - \mathbf{F}_1$$

$$\text{scale residual up:} \quad \Delta_{\text{up}} = \text{Up}(\Delta)$$

$$\text{output feature map:} \quad \mathbf{F}_{\text{up}} = \mathbf{F}'_{\text{up}} + \Delta_{\text{up}}$$

Our work: PU-GAN

Loss functions:

- Adversarial loss:

$$\mathcal{L}_{\text{gan}}(G) = \frac{1}{2}[D(Q) - 1]^2$$

$$\mathcal{L}_{\text{gan}}(D) = \frac{1}{2}[D(Q)^2 + (D(\hat{Q}) - 1)^2]$$

- Reconstruction loss (underlying surface):

$$\mathcal{L}_{\text{rec}} = \min_{\phi: Q \to \hat{Q}} \sum_{q_i \in Q} \|q_i - \phi(q_i)\|_2$$

- Uniform loss:

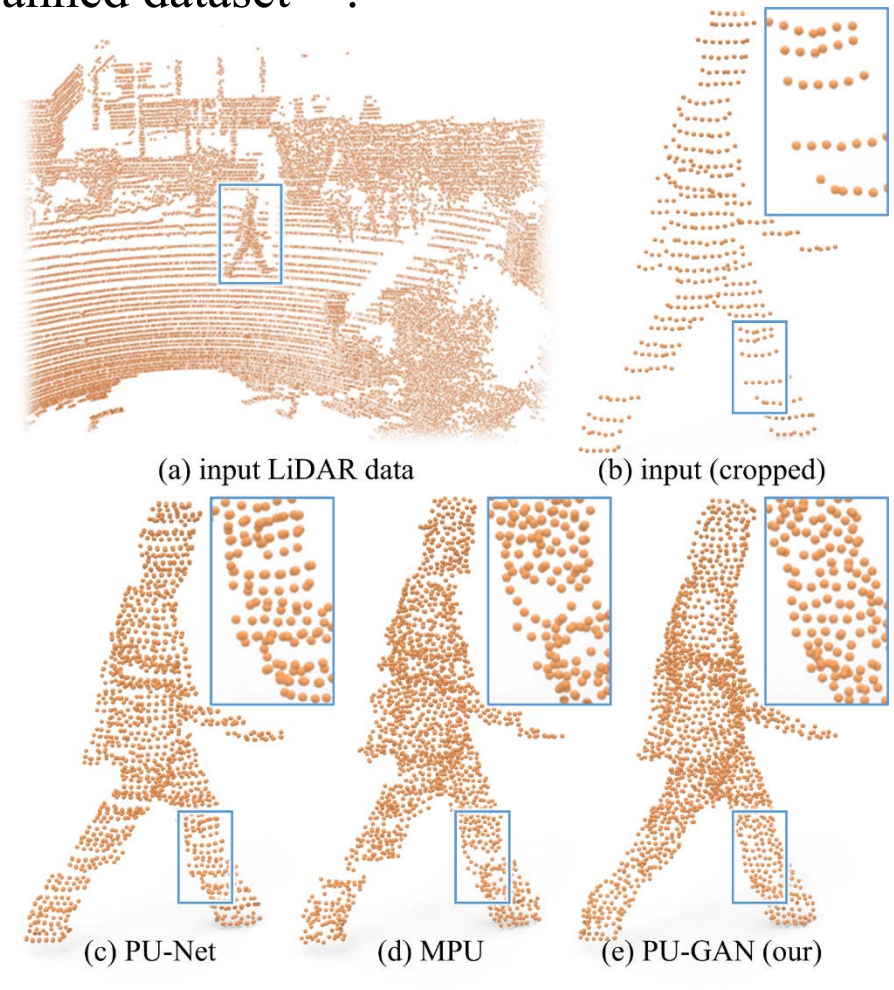
$$\mathcal{L}_{\text{uni}} = \sum_{j=1}^{M} U_{\text{imbalance}}(S_j) \cdot U_{\text{clutter}}(S_j)$$

Global point coverage

Local point distribution

Our work: PU-GAN

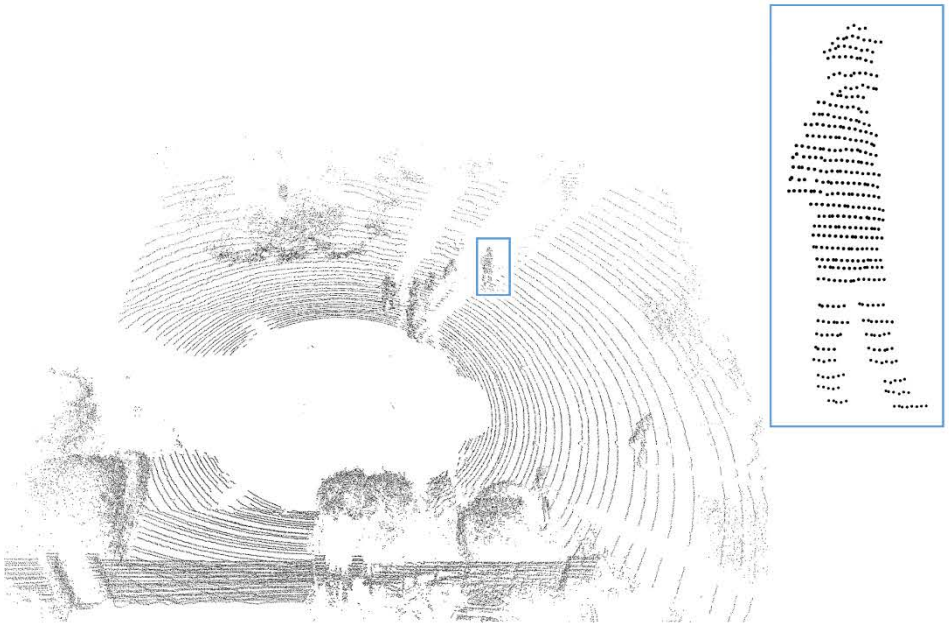
Results on real-scanned dataset [1]:



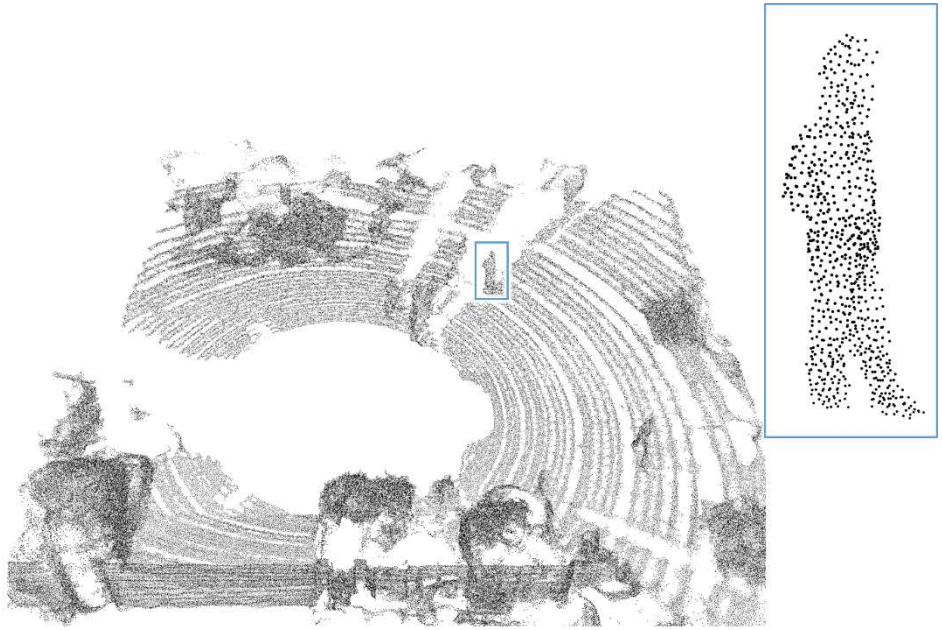
[1] A. Geiger, et al. “Vision meets robotics: The KITTI dataset.” The International Journal of Robotics Research. 2013.

Our work: PU-GAN

Results on real-scanned dataset [1]:



LiDAR Input

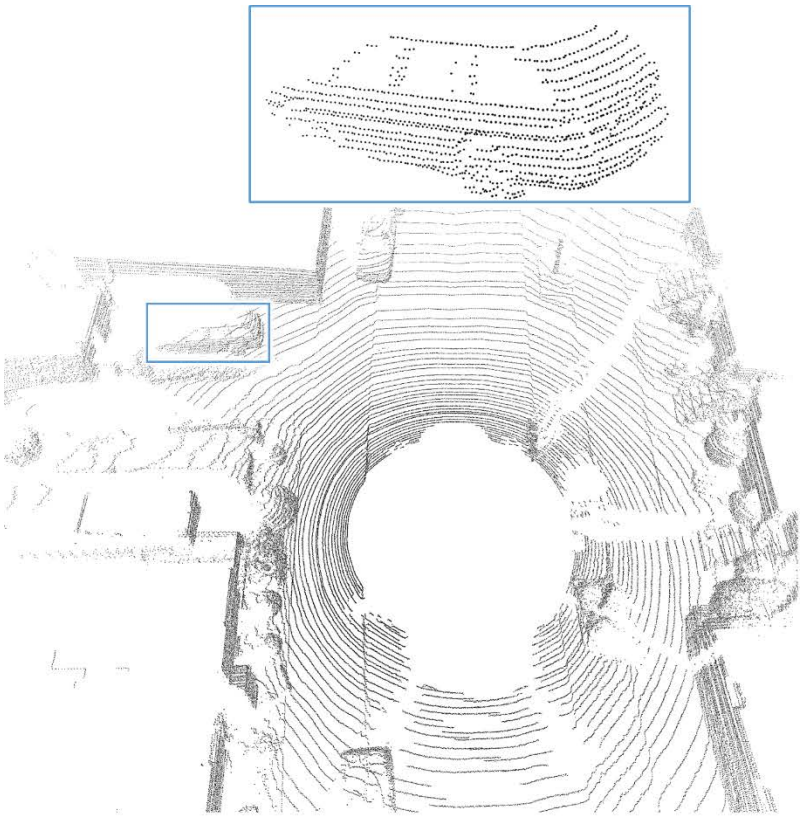


PU-GAN

[1] A. Geiger, et al. “Vision meets robotics: The KITTI dataset.” The International Journal of Robotics Research. 2013.

Our work: PU-GAN

Results on real-scanned dataset [1]:



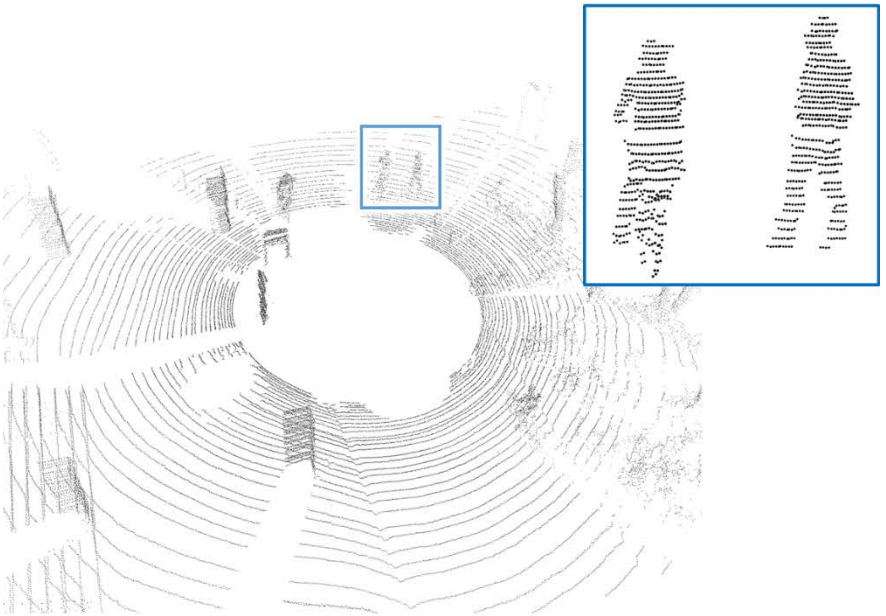
LiDAR Input

PU-GAN

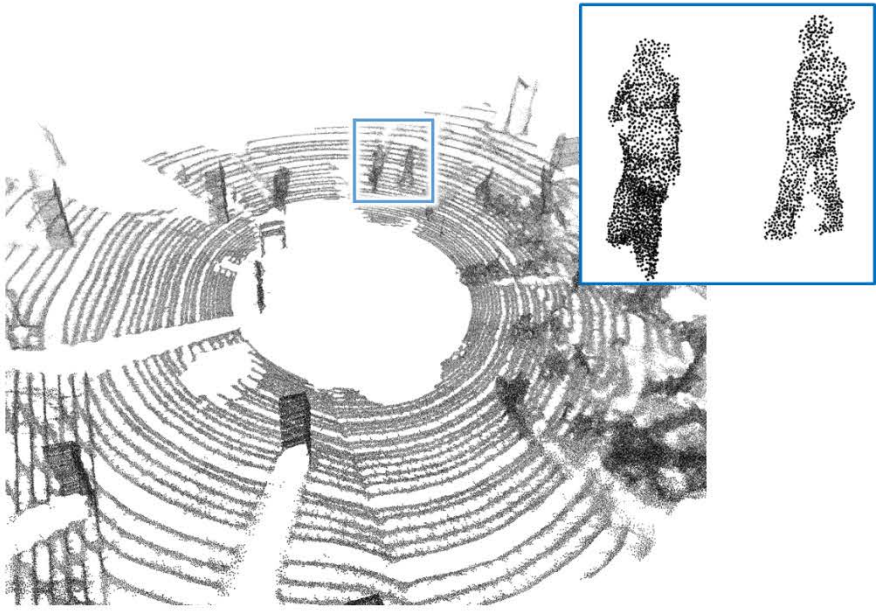
[1] A. Geiger, et al. “Vision meets robotics: The KITTI dataset.” The International Journal of Robotics Research. 2013.

Our work: PU-GAN

Results on real-scanned dataset [1]:



LiDAR Input



PU-GAN

[1] A. Geiger, et al. “Vision meets robotics: The KITTI dataset.” The International Journal of Robotics Research. 2013.

Conclusions

Conclusions:

- Deep neural networks demonstrate powerful capabilities in point cloud upsampling.

  Codes of our works are available:

PU-Net

EC-Net

PU-GAN

- A space rich of open problems and opportunities.
  - point cloud denoise / point cloud completion
  - weakly-supervised / unsupervised learning
  - domain adaptation / transfer learning

Thank you!

Personal webpage:

https://nini-lxz.github.io/