CS5182 Computer Graphics Geometry Representation Tutorial

2024/25 Semester A

City University of Hong Kong (DG)

Explicit Representation: Point Cloud

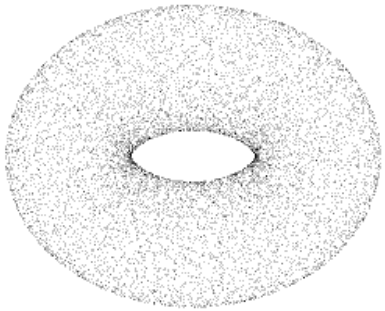
- Defination
  - A point cloud is a collection of points in 3D space that represents the **surface** shape of an object. Each point contains **position** and other relevant **attribute** information (i.e., normal, color)
- Data Structure

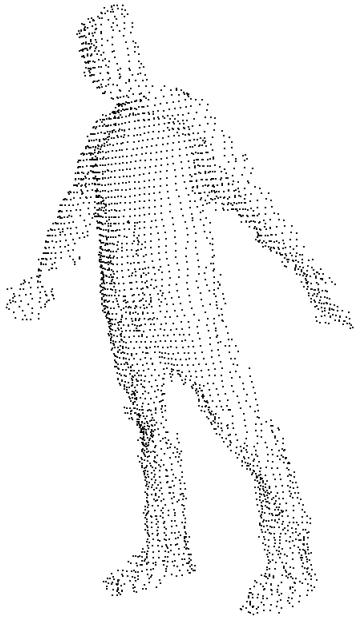
x_1, y_1, z_1	r_1, g_1, b_1	n_1^x, n_1^y, n_1^z
x_2, y_2, z_2	r_2, g_2, b_2	n_2^x, n_2^y, n_2^z
.....	.....	.....
.....	.....	.....
.....	.....	.....
x_N, y_N, z_N	r_N, g_N, b_N	n_N^x, n_N^y, n_N^z

Position Color Normal

Relevant Attributes

.....



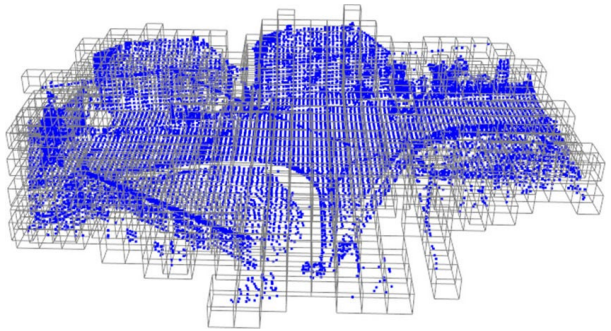


- Properties
  - Unstructured.
  - Sampled from 2D manifolds.
  - No topology between different points.

Explicit Representation: Point Cloud

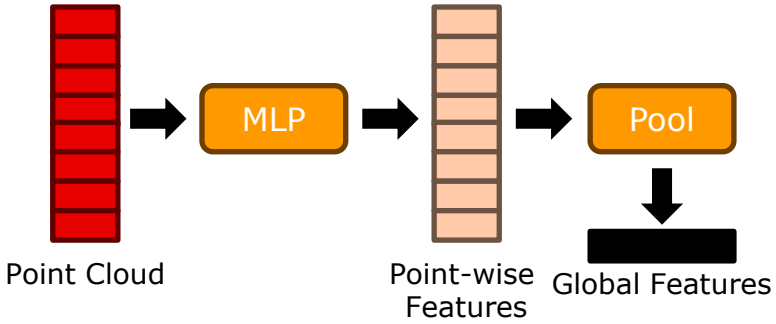
- Processing
  - Voxelization

Transform the point cloud to occupancy volumes, 3D 'images'.



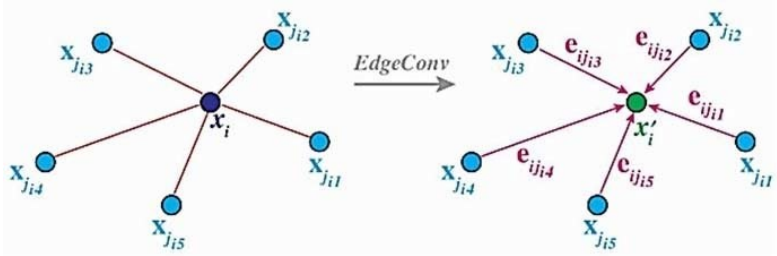
  - MLP + PoolLayer

Individually process each point, then put each feature into a pool layer to get the global feature.



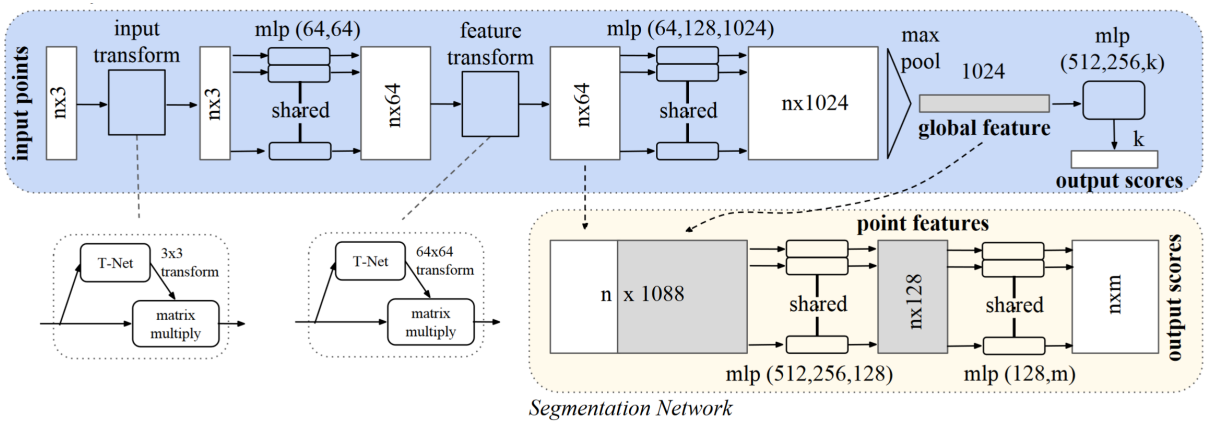
  - Constructing Graph

Construct graph from the point cloud, then utilize graph processing method to deal with it.



Explicit Representation: Point Cloud

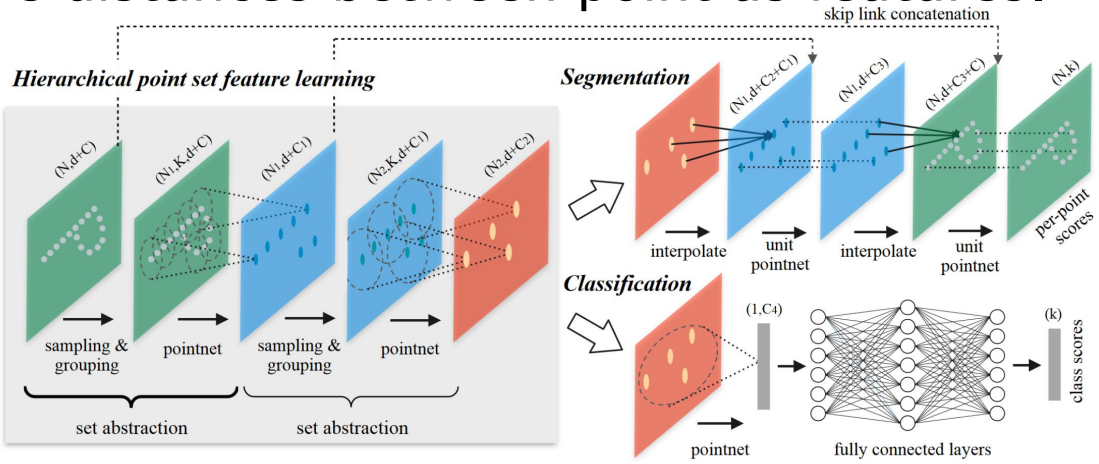
- PointNet(CVPR 2017)
  - The first point-based architecture for point cloud feature extraction, consuming raw point sets as input.
  - Apply shared multi-layers perceptrons (MLPs) for point-wise feature embedding.
  - Achieve permutation-invariance by the symmetric max-pooling operation to generate global features.
  - Without considering neighborhood information, limiting its representation ability.



Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." CVPR 2017.

Explicit Representation: Point Cloud

- PointNet++ (NIPS 2017)
  - Motivation: The previous PointNet ignores modeling local neighborhood information.
  - Apply Mini-PointNet to extract local features.
  - Progressively down-scale the input point cloud by FPS and adopted ball query to search K-nearest-nerghbors of each centroid point.
  - Aggregate features within KNN patches while considering relative distances between point as features.



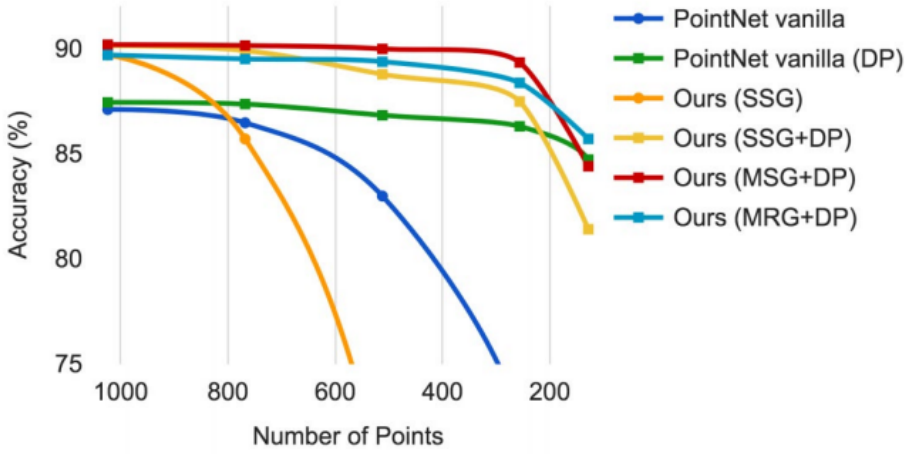
Qi, Charles Ruizhongtai, et al. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space." NIPS. 2017.

Explicit Representation: Point Cloud

- PointNet++ (NIPS 2017)
  - Performance

Method	Input	Accuracy (%)
Subvolume [21]	vox	89.2
MVCNN [26]	img	90.1
PointNet (vanilla) [20]	pc	87.2
PointNet [20]	pc	89.2
Ours	pc	90.7
Ours (with normal)	pc	**91.9**

Table 2: ModelNet40 shape classification.

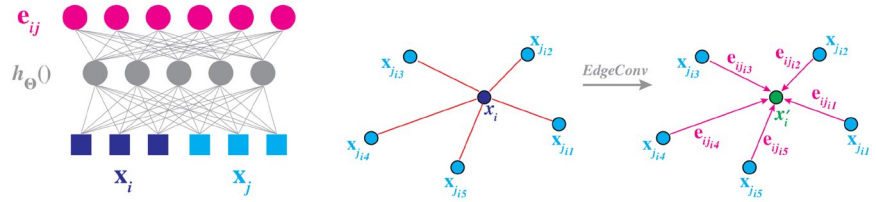


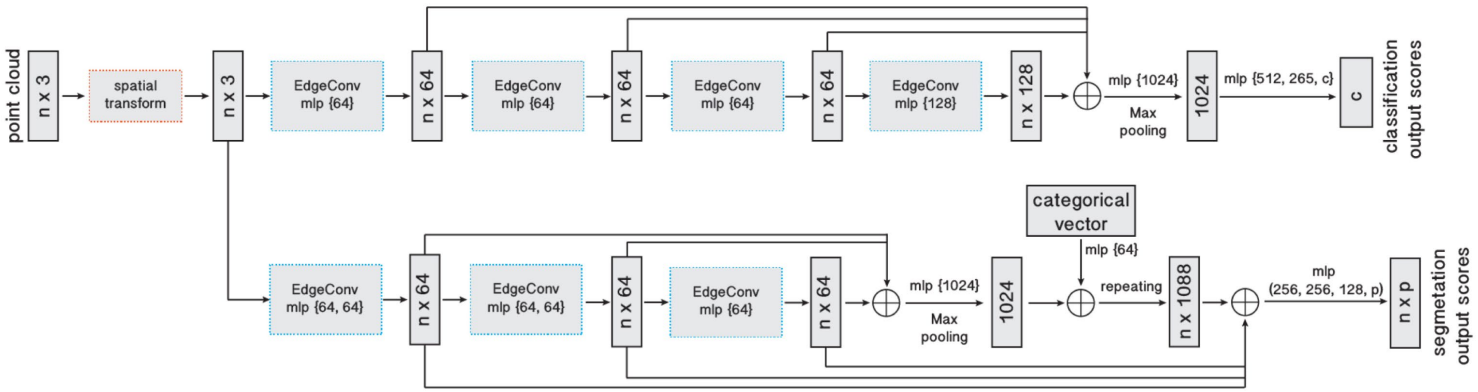
Qi, Charles Ruizhongtai, et al. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space." NIPS. 2017.

Explicit Representation: Point Cloud

- DGCNN (TOG 2019)
  - Motivation: Previous methods defined static graph by KNN, while DGCNN dynamically constructs local graph in the feature space.
  - DGCNN proposes a powerful EdgeConv operator for dynamic point feature extraction.

$$\mathbf{x}_i' = \square_{j:(i,j)\in\mathcal{E}} h_\Theta(\mathbf{x}_i, \mathbf{x}_j).$$

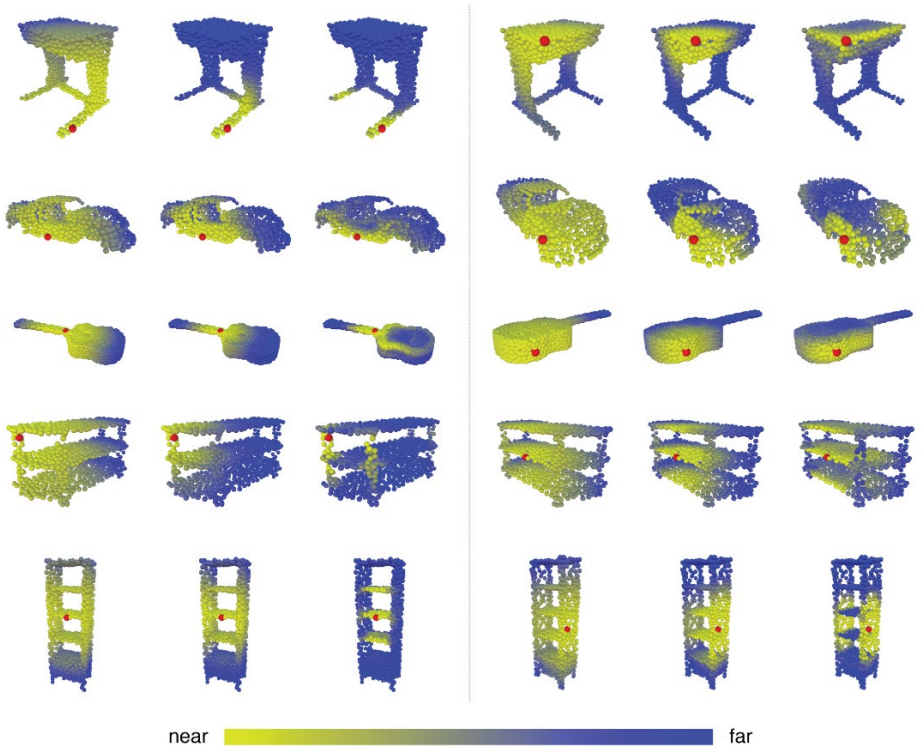




Wang, Yue, et al. "Dynamic graph cnn for learning on point clouds." TOG. 2019.

Explicit Representation: Point Cloud

- DGCNN (TOG 2019)
  - Performance



Distance in feature space. **Left**: Euclidean distance in the input space. **Middle**: Distance after the point cloud transform stage. **Right**: Distance in the feature space of the last layer.

Wang, Yue, et al. "Dynamic graph cnn for learning on point clouds." TOG. 2019.

Explicit Representation: Point Cloud

- DGCNN (TOG 2019)
  - Performance

Table 2. Classification Results on ModelNet40

	Mean Class Accuracy	Overall Accuracy
3DShapeNets (Wu et al. 2015)	77.3	84.7
VoxNet (Maturana and Scherer 2015)	83.0	85.9
Subvolume (Qi et al. 2016)	86.0	89.2
VRN (single view) (Brock et al. 2016)	88.98	–
VRN (multiple views) (Brock et al. 2016)	91.33	–
ECC (Simonovsky and Komodakis 2017)	83.2	87.4
PointNet (Qi et al. 2017b)	86.0	89.2
PointNet++ (Qi et al. 2017c)	–	90.7
Kd-net (Klokov and Lempitsky 2017)	–	90.6
PointCNN (Li et al. 2018a)	88.1	92.2
PCNN (Atzmon et al. 2018)	–	92.3
Ours (baseline)	88.9	91.7
Ours	**90.2**	**92.9**
Ours (2048 points)	**90.7**	**93.5**

Table 3. Complexity, Forward Time, and Accuracy of Different Models

	Model size(MB)	Time(ms)	Accuracy(%)
PointNet (Baseline) (Qi et al. 2017b)	9.4	6.8	87.1
PointNet (Qi et al. 2017b)	40	16.6	89.2
PointNet++ (Qi et al. 2017c)	12	163.2	90.7
PCNN (Atzmon et al. 2018)	94	117.0	92.3
Ours (Baseline)	11	19.7	91.7
Ours	21	27.2	92.9

Table 6. Part Segmentation Results on ShapeNet Part Dataset

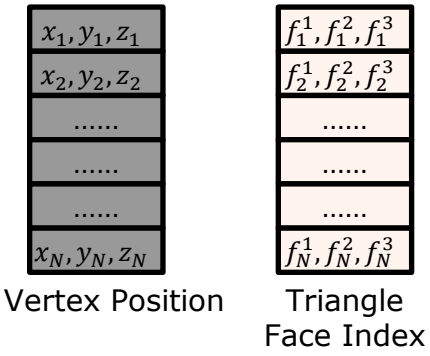
	mean	AREO	BAG	CAP	CAR	CHAIR	EAR PHONE	GUITAR	KNIFE	LAMP	LAPTOP	MOTOR	MUG	PISTOL	ROCKET	SKATE BOARD	TABLE
# shapes		2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271
PointNet	83.7	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
PointNet++	85.1	82.4	79.0	**87.7**	77.3	**90.8**	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
Kd-Net	82.3	80.1	74.6	74.3	70.3	88.6	73.5	90.2	87.2	81.0	94.9	57.4	86.7	78.1	51.8	69.9	80.3
LocalFeatureNet	84.3	**86.1**	73.0	54.9	77.4	88.8	55.0	90.6	86.5	75.2	**96.1**	57.3	91.7	83.1	53.9	72.5	**83.8**
PCNN	85.1	82.4	80.1	85.5	79.5	**90.8**	73.2	91.3	86.0	85.0	95.7	73.2	94.8	83.3	51.0	75.0	81.8
PointCNN	**86.1**	84.1	**86.45**	86.0	**80.8**	90.6	**79.7**	**92.3**	**88.4**	**85.3**	**96.1**	**77.2**	**95.3**	**84.2**	**64.2**	**80.0**	83.0
Ours	85.2	84.0	83.4	86.7	77.8	90.6	74.7	91.2	87.5	82.8	95.7	66.3	94.9	81.1	63.5	74.5	82.6

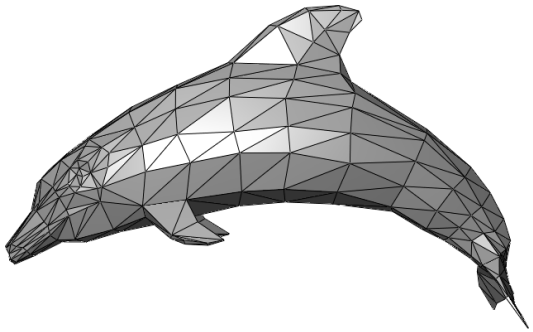
Metric is mIoU(%) on points.

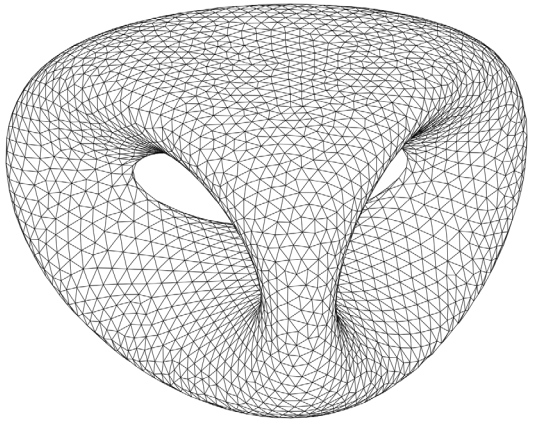
Wang, Yue, et al. "Dynamic graph cnn for learning on point clouds." TOG. 2019.

Explicit Representation: Triangle Mesh

- Definition
  - Triangle mesh is a type of polygon mesh, and it comprises a set of triangles that are connected by their shared edges or vertices.
- Data Structure





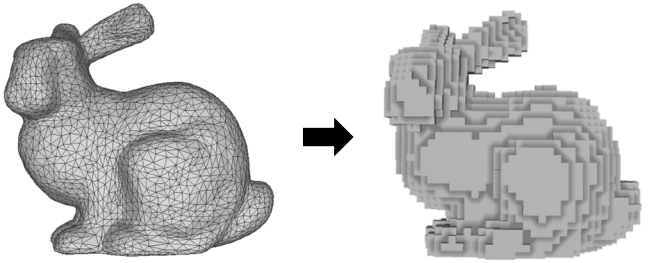


- Properties
  - Topology between different vertices and triangles;
  - Discrete representation of 2D manifolds;
  - Small number of triangles can represent large, simple surfaces.

Explicit Representation: Triangle Mesh

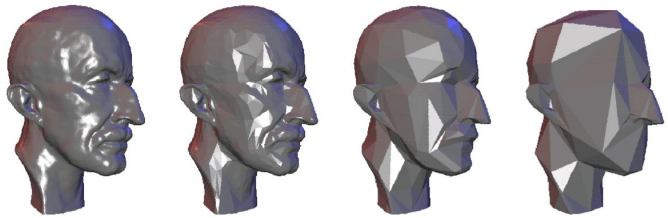
- Processing
  - Voxelization

  According to the occupancy at each grid, the triangle mesh is converted to voxel data, a 3D 'image', where the image methods could be applied directly.



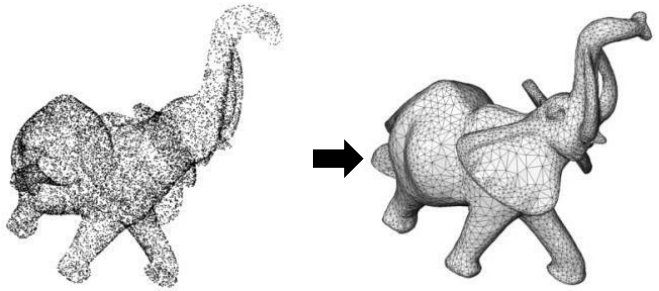
  - Mesh Simplification

  Reduce the complexity of a triangle mesh while preserving the overall shape and appearance of the shape.



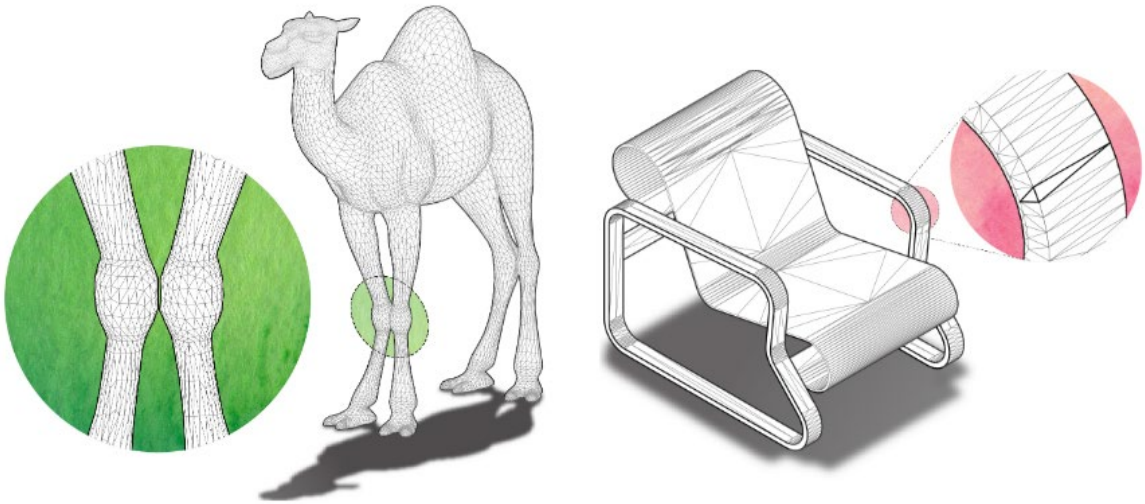
  - Surface Reconstruction

  Reconstruct triangle meshes from the given point clouds.



Explicit Representation: Triangle Mesh

- MeshCNN (TOG 2019)
  - Motivation:
    - Polygonal meshes leverage non-uniformity to represent large flat regions as well as sharp, intricate features.
    - The non-uniformity make the neural networks with convolution and pooling operations difficult.



Non-uniform representation.

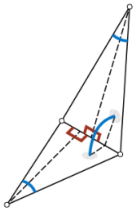
Hanocka, Rana, et al. “Meshcnn: a network with an edge.” TOG. 2019.

Explicit Representation: Triangle Mesh

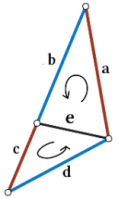
- MeshCNN (TOG 2019)
  - MeshCNN combines specialized convolution and pooling layers on the mesh edges.
  - Convolutions are applied on edges and the four edges of their incident triangles, and the pooling is applied via an edge collapse.
  - The input edge feature is a 5D vector for every edge: the dihedral angle, two inner angles, and two edge-length ratios for each face.
  - Invariant convolution:

$$e \cdot k_0 + \sum_{j=1}^{4} k_j \cdot e^j, \quad (e^1, e^2, e^3, e^4) = (|a-c|, a+c, |b-d|, b+d).$$

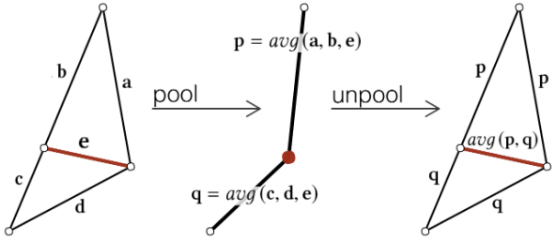
  - Mesh Pooling



Input Edge Features



Mesh Convolution



Mesh Pooling & Unpooling

Hanocka, Rana, et al. “Meshcnn: a network with an edge.” TOG. 2019.

Explicit Representation: Triangle Mesh

- MeshCNN (TOG 2019)
  - Performance

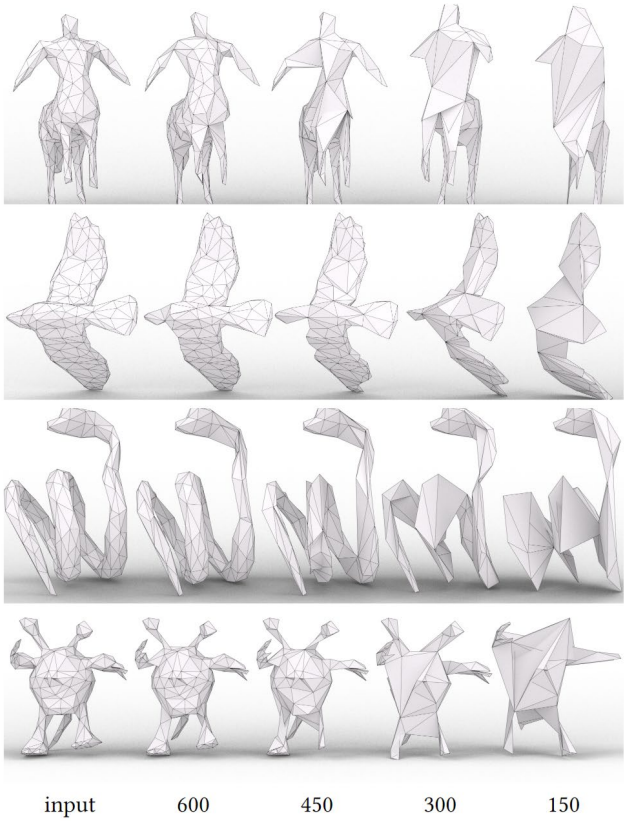


Fig. 6. Intermediate pooled meshes on the SHREC11 classification dataset. The input meshes are all simplified to roughly 750 edges (500 faces), and are sequentially pooled to 600, 450, 300 and 150 edges.

Classification SHREC

Method	Split 16	Split 10
MeshCNN	**98.6**	**91.0%**
GWCNN	96.6%	90.3%
GI	96.6%	88.6%
SN	48.4%	52.7%
SG	70.8%	62.6%

[Ezuz et al. 2017]

Table 1. SHREC 30 class classification (comparisons taken from [2017]). Split 16 and 10 are the training splits, trained up to 200 epochs.

COSEG Segmentation

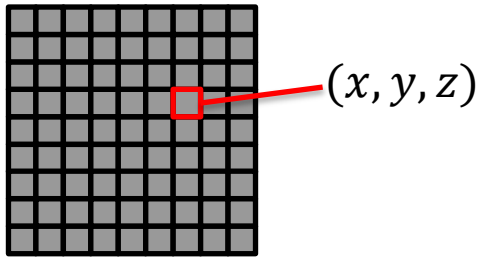
Method	Vases	Chairs	Telealiens
MeshCNN (*UNet*)	**97.27%**	**99.63%**	**97.56%**
MeshCNN (*rand. pool*)	96.64%	99.23%	97.01%
PointNet	91.5%	70.2%	54.4%
PointNet++	94.7%	98.9%	79.1%
PointCNN	96.37%	99.31%	97.40%

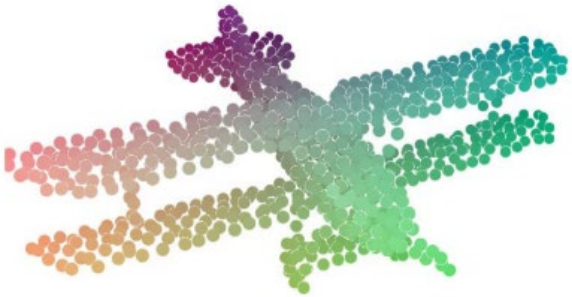
Table 3. MeshCNN evaluations on COSEG segmentation.

Hanocka, Rana, et al. “Meshcnn: a network with an edge.” TOG. 2019.

Explicit Representation: Geometry Img

- Definition
  - Geometry image is the re-organization of 3D data, such as point clouds or triangle meshes, where the points or vertices are arranged in a 2D matrix.
- Data Structure



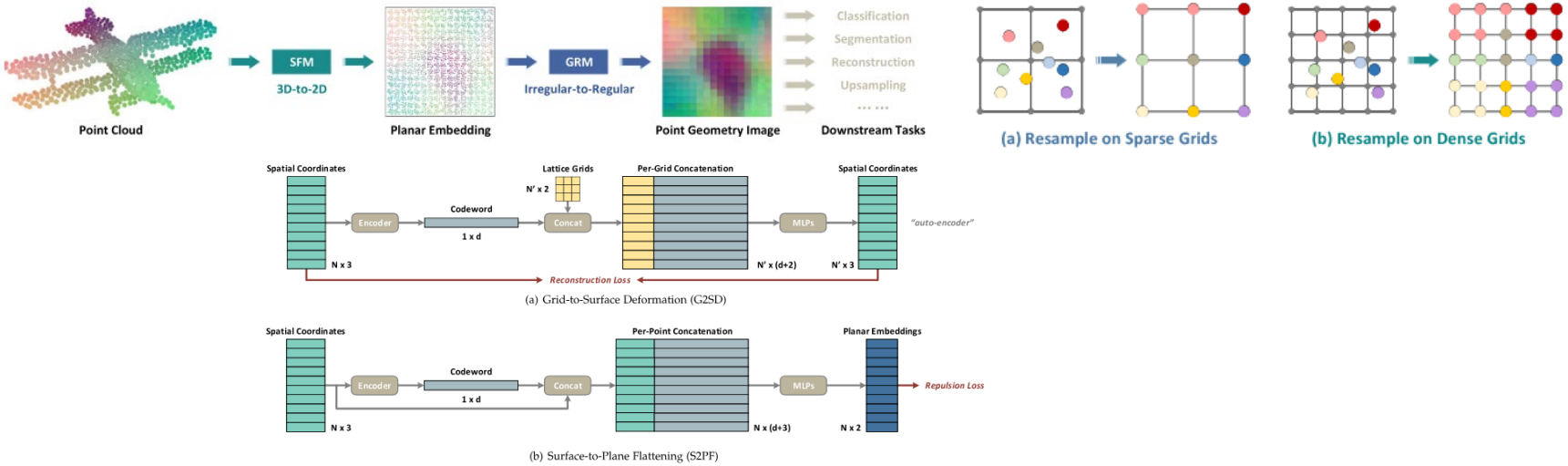


Re-organize

- Properties
  - Geometry images are 2D structured data, and the methods for 2D images could be directly applied on it;
  - The local neighbourhood relationships in 3D space remain unchanged with in the geometry image.

Explicit Representation: Geometry Img

- Flattening-Net (TPAMI)
  - Motivation: Unstructured nature of point cloud make it difficult to process, and KNN operation is time and memory-consuming.
  - Flattening-Net implicitly approximates a locally smooth 3D-to-2D flattening process to convert point clouds to geometry images, while preserving neighborhood consistency.
  - The proposed 3D representation is combined with various application.



Zhang, Qijian, et al. "Flattening-net: Deep regular 2d representation for 3d point cloud analysis." TPAMI.

Explicit Representation: Geometry Img

- Flattening-Net (TPAMI)
  - Performance

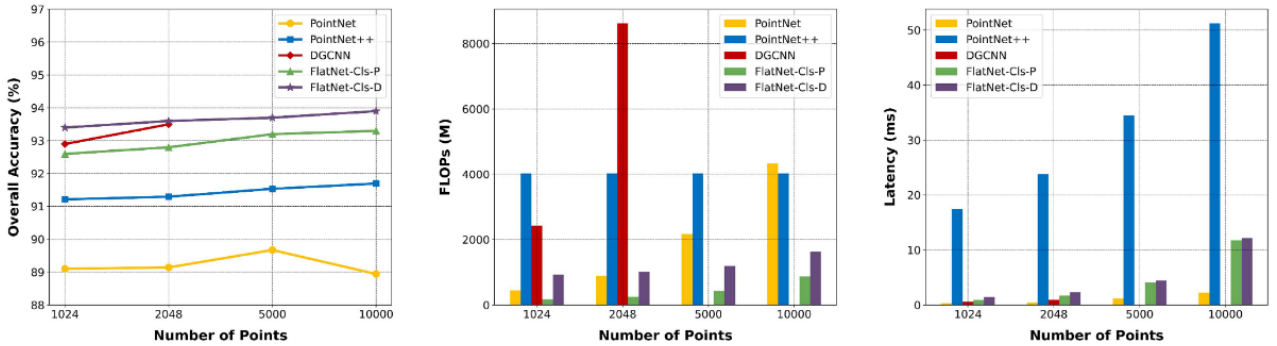


Fig. 7. Comparison of classification accuracy, FLOPs, and latency of different methods when dealing with increasing number of input points.

TABLE IV
OVERALL ACCURACY (OA) OF DIFFERENT DEEP SHAPE CLASSIFICATION METHODS ON MODELNET40. "∗" MEANS THAT POINT-WISE NORMALS ARE CONSUMED AS ADDITIONAL INPUT ATTRIBUTES

Method	# Points	OA (%)
PointNet-vanilla [22]	1024	87.1
PointNet [22]	1024	89.2
PointNet++ [23] ∗	5000	91.9
SpiderCNN [28] ∗	1024	92.4
SO-Net [26] ∗	5000	93.4
PointConv [30] ∗	1024	92.5
DGCNN [35]	1024	92.9
FlatNet-Cls-P	1024	92.6
FlatNet-Cls-D	1024	93.4

TABLE VII
PERFORMANCE OF PART SEGMENTATION ON SHAPENETPART MEASURED BY MEAN INTERSECTION-OVER-UNION (MIOU), WHERE "∗" MEANS THAT POINT-WISE NORMALS ARE CONSUMED AS ADDITIONAL INPUT ATTRIBUTES

Method	# Points	mIoU (%)
PointNet [22]	2048	83.7
PointNet++ [23] ∗	2048	85.1
SpiderCNN [28] ∗	2048	85.3
SO-Net [33] ∗	2048	84.9
PointConv [30] ∗	2048	85.7
DGCNN [35]	2048	85.1
FlatNet-Seg-P	2048	84.9
FlatNet-Seg-D	2048	85.8

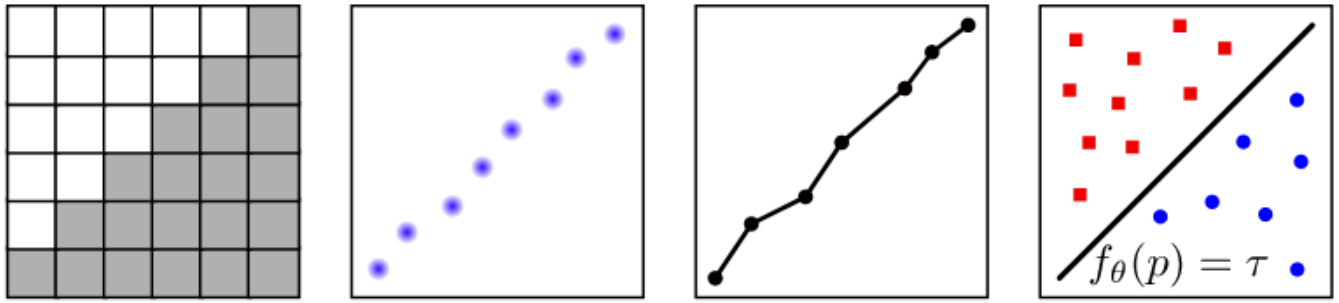
Zhang, Qijian, et al. "Flattening-net: Deep regular 2d representation for 3d point cloud analysis." TPAMI.

Implicit Representation

- Definition
  - Implicit representation utilizes the isosurface of a scalar field to represent the surface of the shape.
  - $f:\mathbb{R}^3\rightarrow\mathbb{R}$ is a scalar field, then the surface $\mathcal{S}$ is defined as

$$\mathcal{S}=\{\mathbf{x}|f(\mathbf{x})=\tau\},$$

where τ is the value of the isosurface.



Voxel | Point Cloud | Triangle Mesh | Implicit Field

Implicit Representation: BOF

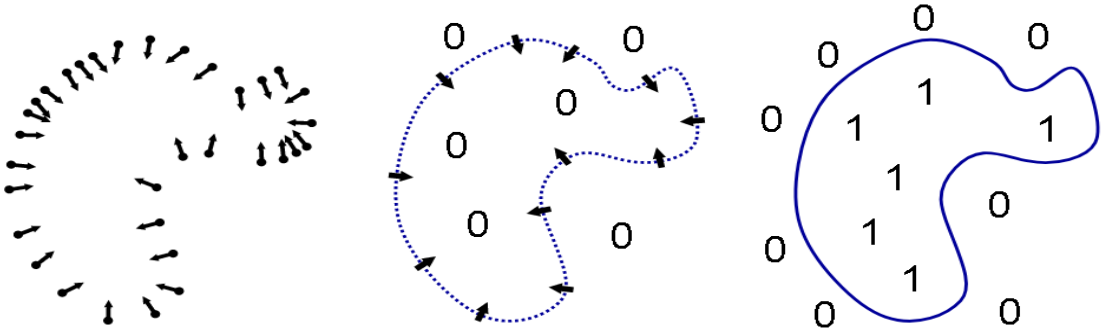
- Definition
  - The whole space is divided into two parts, inside and outside the shape. Given a point $\mathbf{x}\in\mathbb{R}$^3, its Binary Occupancy Field is defined as

$$f(\mathbf{x}) = \begin{cases} 1, \text{if } \mathbf{x} \text{ is outside the shape,} \\ 0, \text{else.} \end{cases}$$

  - In this condition, the isosurface with the value $\tau = 0.5$ represents the surface of the shape.
- Properties
  - BOF can only represent watertight shapes;
  - BOF represents the probability of the query points' location, thus it is less accurate than the distance fields when representing the surfaces.

Implicit Representation: BOF

- Poisson Surface Reconstruction (SGP 2006)
  - Input is a point cloud and its oriented normal vectors, and the output is the BOF of the shape.
  - The BOF only change from 0 to 1 on the surface, and it remains unchanged inside and outside the shape.
  - PSR builds the Poisson Equation about BOF according to the normal vectors,
  - $\nabla f(\mathbf{x}) = \mathbf{N}(\mathbf{x})$,
  - Then BOF could be calculated by solving Poisson Equation.

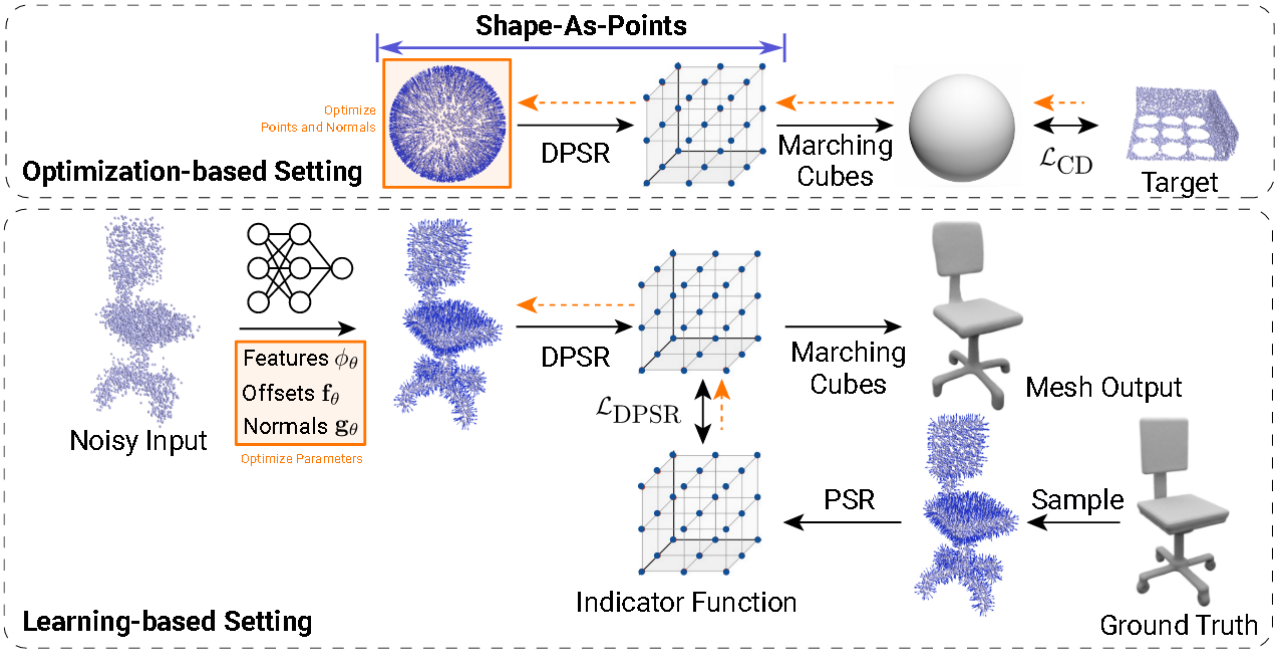


Oriented Point Cloud $\mathbf{P}, \mathbf{N}$ | Gradient of BOF, $\nabla f(\mathbf{x})$ | Solved BOF, f

Kazhdan, Michael, Matthew Bolitho, and Hugues Hoppe. "Poisson surface reconstruction." SGP. 2006.

Implicit Representation: BOF

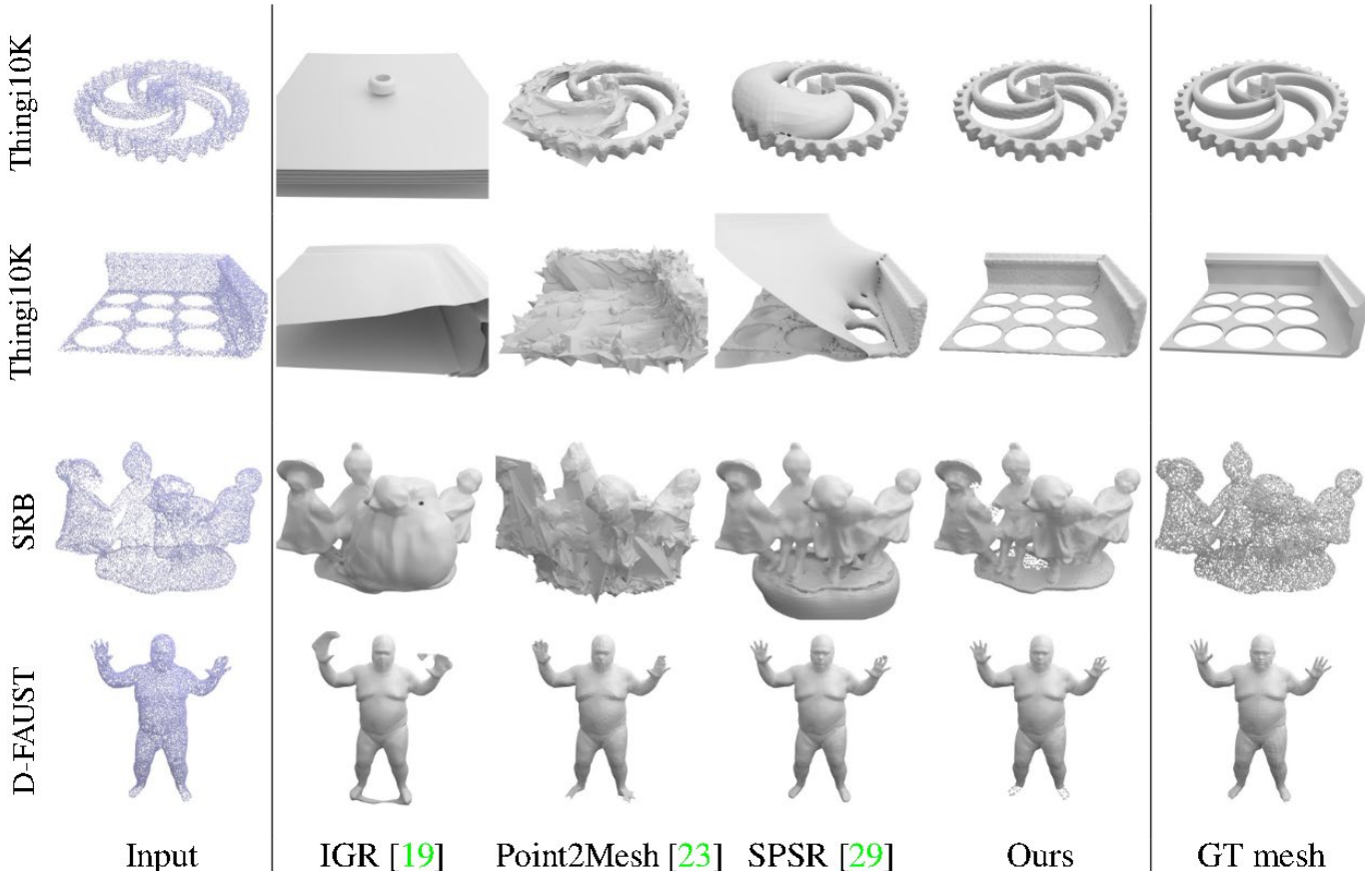
- Shape as Points (SAP, NIPS 2021)
  - Motivation: Previous PSR is not differentiable, limiting its application.
  - SAP introduces a differentiable point-to-mesh layer.
  - Solve Poisson Equation through spectral method, FFT and IFFT, improving the efficiency.



Peng, Songyou, et al. "Shape as points: A differentiable poisson solver." NIPS. 2021.

Implicit Representation: BOF

- Shape as Points (SAP, NIPS 2021)
  - Performance

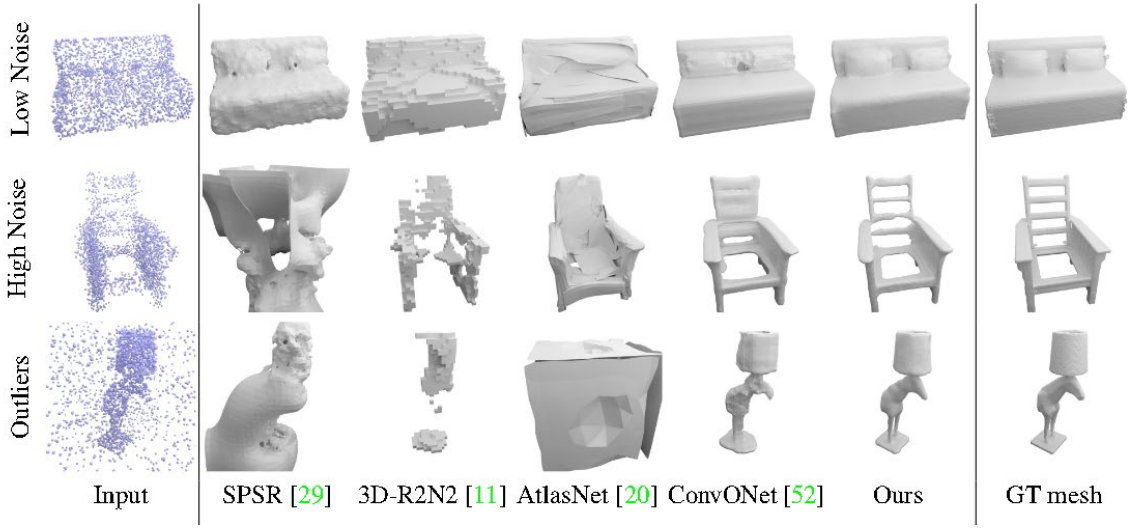


Peng, Songyou, et al. "Shape as points: A differentiable poisson solver." NIPS. 2021.

Implicit Representation: BOF

- Shape as Points (SAP, NIPS 2021)
  - Performance

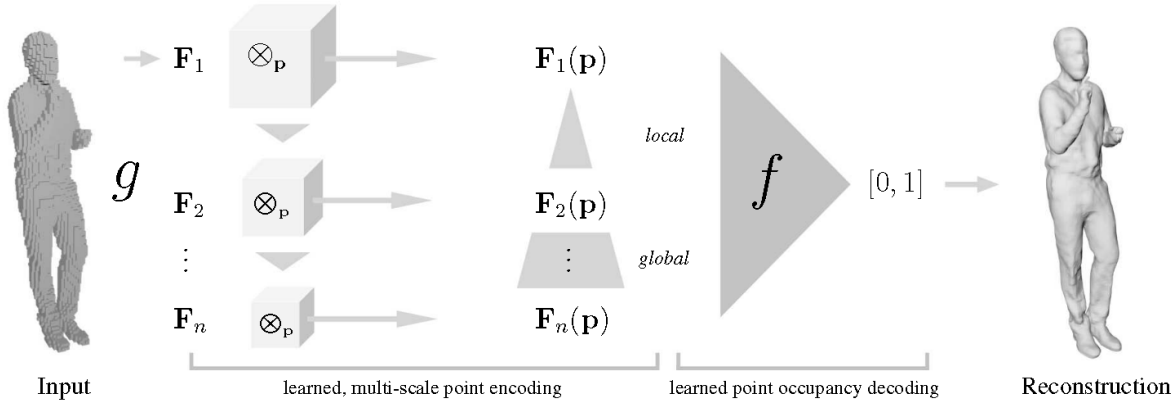
Dataset	Method	Chamfer-L_1 (↓)	F-Score (↑)	Normal C. (↑)	Time (s)
Thingi10K	IGR [19]	0.440	0.505	0.692	1842.3
	Point2Mesh [23]	0.109	0.656	0.806	3714.7
	SPSR [29]	0.223	0.787	0.896	9.3
	Ours	**0.054**	**0.940**	**0.947**	370.1
SRB	IGR [19]	0.178	0.755	–	1847.6
	Point2Mesh [23]	0.116	0.648	–	4707.9
	SPSR [29]	0.232	0.735	–	9.2
	Ours	**0.076**	**0.830**	–	326.0
D-FAUST	IGR [19]	0.235	0.805	0.911	1857.2
	Point2Mesh [23]	0.071	0.855	0.905	3678.7
	SPSR [29]	0.044	**0.966**	**0.965**	4.3
	Ours	**0.043**	**0.966**	0.959	379.9



Peng, Songyou, et al. "Shape as points: A differentiable poisson solver." NIPS. 2021.

Implicit Representation: BOF

- IF-Net (CVPR 2020)
  - IF-Net could handle different inputs, such as voxelization, partial point cloud.
  - It extract a learnable 3D multi-scale tensor of deep features, which is aligned with the original Euclidean space embedding the shape.
  - Given a query point, its feature at each scale are calculated by special interpolation.



$$f(\mathbf{F}_1(\mathbf{p}), \ldots, \mathbf{F}_n(\mathbf{p})) \colon \mathcal{F}_1 \times \ldots \times \mathcal{F}_n \mapsto [0, 1]$$

Chibane, Julian, Thiemo Alldieck, and Gerard Pons-Moll. "Implicit functions in feature space for 3d shape reconstruction and completion." CVPR. 2020.

Implicit Representation: BOF

- IF-Net (CVPR 2020)
  - Performance of various input type

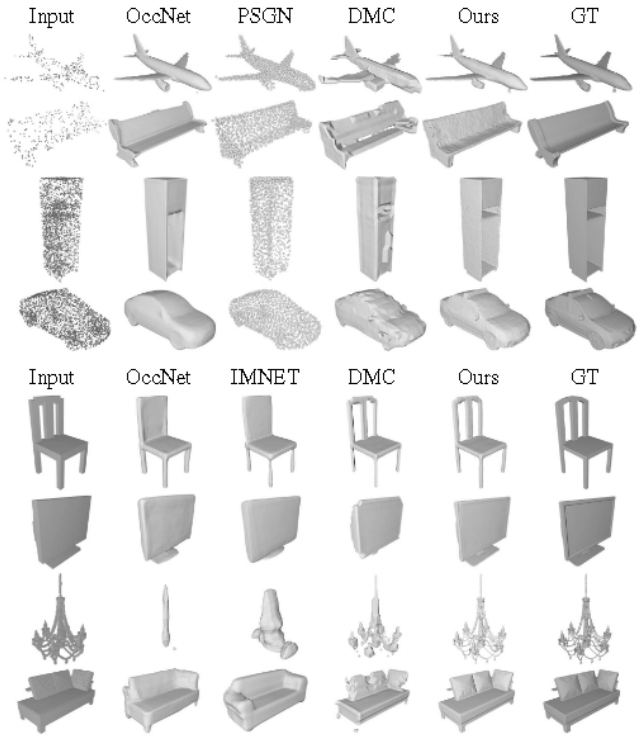


Figure 3. Qualitative results for two input types: point cloud (top) and voxels (bottom) on ShapeNet dataset. Each type is further subdivided into sparse (top two rows) and dense (bottom two rows).

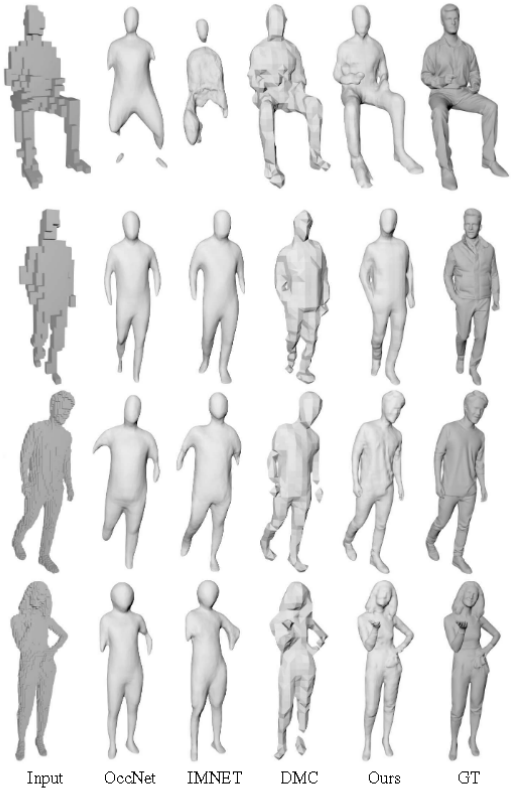


Figure 4. Qualitative results of sparse (32^3, upper) and dense (128^3, lower) 3D voxel super-resolution on the Humans dataset.

Chibane, Julian, Thiemo Alldieck, and Gerard Pons-Moll. "Implicit functions in feature space for 3d shape reconstruction and completion." CVPR. 2020.

Implicit Representation: SDF

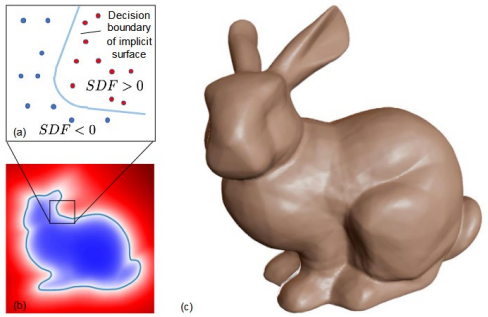
- Definition
  - The whole space is divided into two parts, inside and outside the shape. Given a point $\mathbf{x}\in\mathbb{R}^3$, its Signed Distance Field is defined as

$$f(\mathbf{x}) = \begin{cases} +d(\mathbf{x}), \text{if } \mathbf{x} \text{ is inside the shape,} \\ -d(\mathbf{x}), \text{else.} \end{cases}$$

  - In this condition, $d(\mathbf{x})$ represents $\mathbf{x}$'s closest distance to the surface, and the isosurface with the value τ=0 represents the surface of the shape.
- Properties
  - SDF can only represent watertight shapes;
  - SDF has the differentiable properties,

$$\nabla f(\mathbf{x}) = \mathbf{n}^* \text{ and } \|\nabla f(\mathbf{x})\| = 1$$

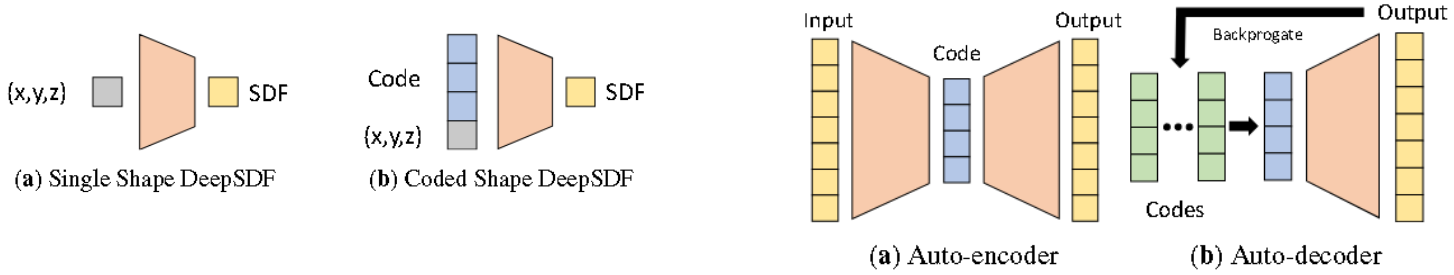
where $\mathbf{n}^*$ is the oriented normal of $\mathbf{x}$'s closest point on the surface



Chibane, Julian, Thiemo Alldieck, and Gerard Pons-Moll. "Implicit functions in feature space for 3d shape reconstruction and completion." CVPR. 2020.

Implicit Representation: SDF

- DeepSDF (CVPR 2019)
  - DeepSDF is a learned continuous SDF representation of a class of shapes.
  - Propose an auto-decoder architecture for shape representation.
  - Conditioning the network with a code vector allows DeepSDF to model a large space of shape.

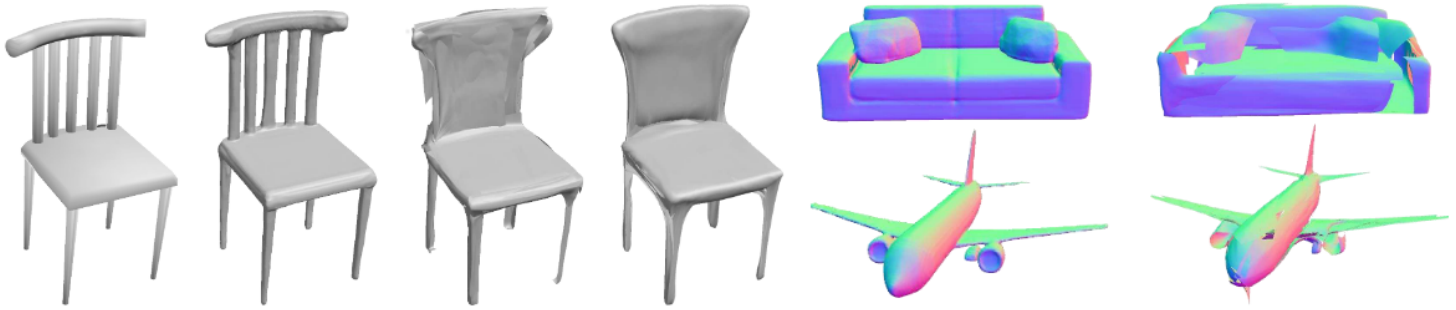


  - A randomly initialized latent vector is assigned to each shape in the beginning of training, and the latent vectors are optimized along with the decoder weights.

Park, Jeong Joon, et al. "Deepsdf: Learning continuous signed distance functions for shape representation." CVPR. 2019.

Implicit Representation: SDF

- DeepSDF (CVPR 2019)
  - Performance



(a) Ground-truth (b) Our Result (c) [19]-25 patch (d) [19]-sphere (e) Our Result (f) [19]-25 patch

Figure 6: Reconstruction comparison between DeepSDF and AtlasNet [19] (with 25-plane and sphere parameterization) for test shapes. Note that AtlasNet fails to capture the fine details of the chair, and that (f) shows holes on the surface of sofa and the plane.

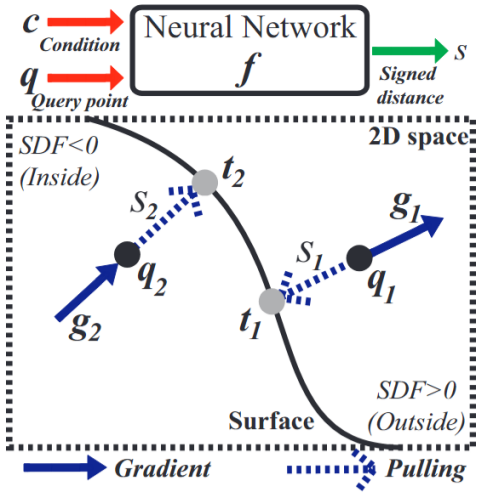
Figure 7: Reconstruction of test shapes. From left to right alternating: ground truth shape and our reconstruction. The two right most columns show failure modes of DeepSDF. These failures are likely due to lack of training data and failure of minimization convergence.

Park, Jeong Joon, et al. "Deepsdf: Learning continuous signed distance functions for shape representation." CVPR. 2019.

Implicit Representation: SDF

- Neural-Pull (ICML 2021)
  - Neural-Pull can learn SDFs directly from raw 3D point clouds without ground truth values.
  - SDFs are learned by updating the predicted signed distance values and the gradient to pull surrounding 3D space onto the surface.

$$\boldsymbol{t}_i' = \boldsymbol{q}_i - \boldsymbol{f}(\boldsymbol{c}, \boldsymbol{q}_i) \times \nabla \boldsymbol{f}(\boldsymbol{c}, \boldsymbol{q}_i) / ||\nabla \boldsymbol{f}(\boldsymbol{c}, \boldsymbol{q}_i)||_2,$$



Ma, Baorui, et al. "Neural-pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces." ICML. 2021

Implicit Representation: SDF

- Neural-Pull (ICML 2021)
  - Performance

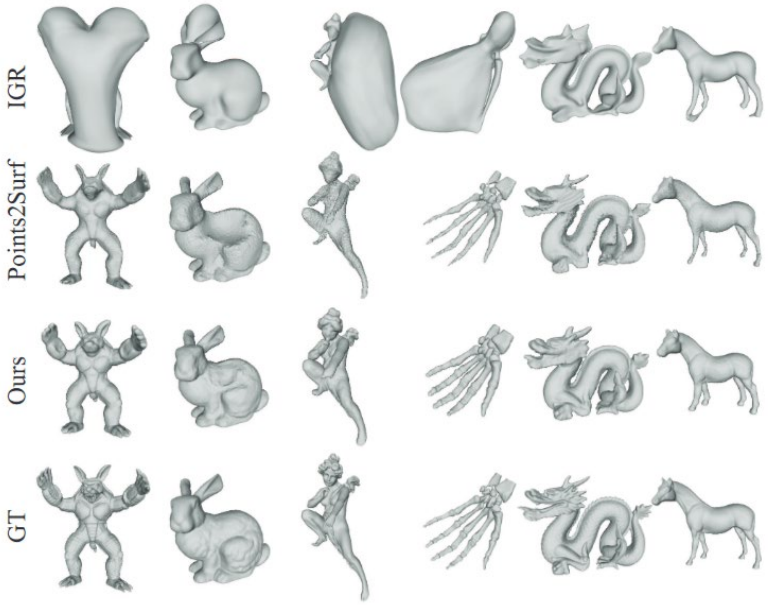


Figure 4. Comparison under FAMOUS in surface reconstruction.

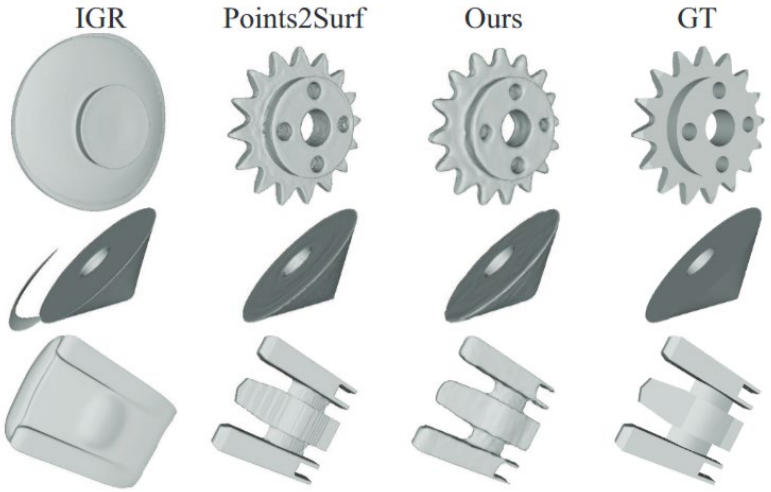


Figure 5. Comparison under ABC in surface reconstruction.

Ma, Baorui, et al. "Neural-pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces." ICML. 2021

Implicit Representation: UDF

- Definition
  - Unsigned Distance Field is the closest distance from the query point $\mathbf{x}$ to the surface

$$f(\mathbf{x})=d(\mathbf{x})$$

  - In this condition, $d(\mathbf{x})$ represents $\mathbf{x}$'s closest distance to the surface, and the isosurface with the value $\tau=0$ represents the surface of the shape.

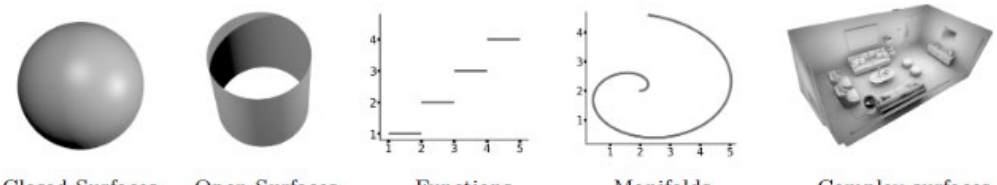
- Properties
  - UDF is the absolute value of SDF.
  - UDF can represent more general shapes, such as those with boundary and multiple layers;
  - UDF has the following differentiable properties,

$$\nabla f(\mathbf{x}) = \mathbf{n}^* \text{ and } \|\nabla f(\mathbf{x})\| = 1$$

  - where $\mathbf{n}\hat{}*$ is the aligned normal of $\mathbf{x}$'s
  - closest point on the surface

Implicit Representation: UDF

- NDF (NIPS 2020)
  - Motivation: Existing neural implicit representations are limited to closed surfaces, and they cannot handle more general shapes, such as clothing or a car with inner structures.



	Closed Surfaces	Open Surfaces	Functions	Manifolds	Complex surfaces
SDF/Occ	✓	✗	✗	✗	✗
NDF	✓	✓	✓	✓	✓

  - NDF can predict UDF from sparse point clouds.
  - The input point clouds are voxelized first and encoded by a 3D CNN (following IF-Net.)
  - The predicted UDF is differentiable, and based on this, any point in 3D space could be projected on the surface through

$$\mathbf{q} := \mathbf{p} - f(\mathbf{p}) \cdot \nabla_{\mathbf{p}} f(\mathbf{p}), \quad \mathbf{q} \in \mathcal{S} \subset \mathbb{R}^d, \quad \forall \mathbf{p} \in \mathbb{R}^d / \mathcal{C},$$

Chibane, Julian, and Gerard Pons-Moll. "Neural unsigned distance fields for implicit function learning." NIPS. 2020.

Implicit Representation: UDF

- NDF (NIPS 2020)
  - Performance

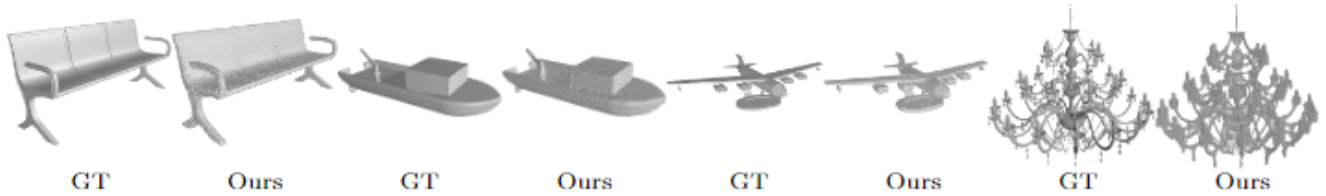


Figure 6: Reconstruction results on all classes of closed ShapeNet data from 3000 points trained with a single model. The quantitative Chamfer-L_2 ↓ results $\times 10^{-4}$ are: OccNet 4.0, PSGN 4.0, DMC 1.0, IF-Net 0.2 and NDF (ours) **0.05**.

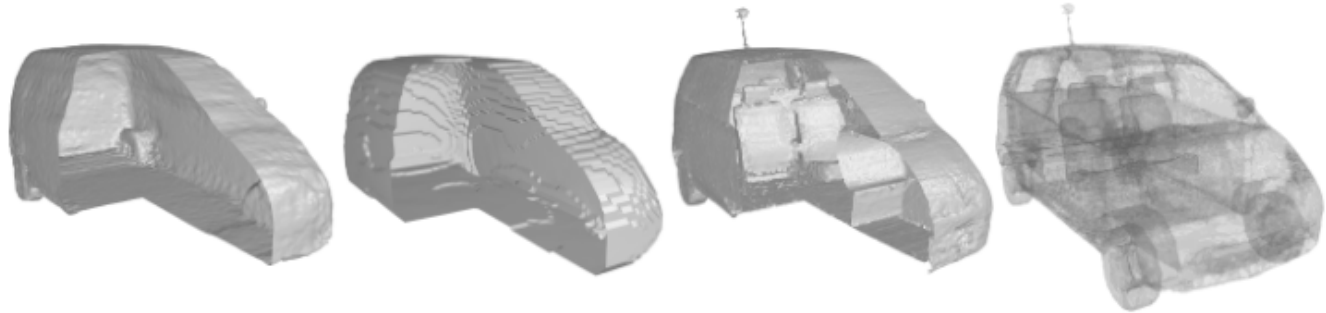


Figure 7: Point cloud reconstruction of inner-structures on a test set car. Ours is the only method to successfully reconstruct full inner-structure. SAL and ours directly trained on raw data. IF-Net trained on closed data (without inner structure) for reference.

Chibane, Julian, and Gerard Pons-Moll. "Neural unsigned distance fields for implicit function learning." NIPS. 2020.

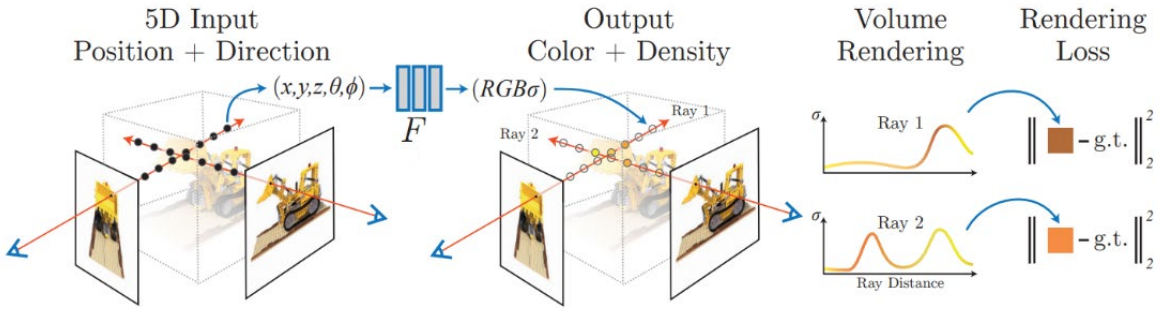
Implicit Representation: Density

- NeRF (ECCV 2020)

NeRF model represents 3D scenes as a **radiance field** approximated by a neural network. The radiance field describes **color** and **volume density** for every point and for every viewing direction in the scene,

$$F(\mathbf{x}, \theta, \phi) \rightarrow (\mathbf{c}, \sigma),$$

where $\mathbf{x} = (x, y, z)$ is the **in-scene coordinate**, (θ, ϕ) is the **azimuthal** and **polar view angles**, $\mathbf{c} = (r, g, b)$ represents **color**, and σ represents the **volume density**.



Given a camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ ($\mathbf{o}$ is the **camera position** and $\mathbf{d}$ is the viewing direction), the **color** $C(\mathbf{r})$ at the corresponding pixel is

$$C(r) = \int_{t_1}^{t_2} T(t) \cdot \sigma\big(\mathbf{r}(t)\big) \cdot \mathbf{c}(\mathbf{r}(t), \mathbf{d}) \cdot dt,$$

where $T(t)$ is the **accumulated transmittance**, representing the probability that the ray travels from t_1 to t without being intercepted,

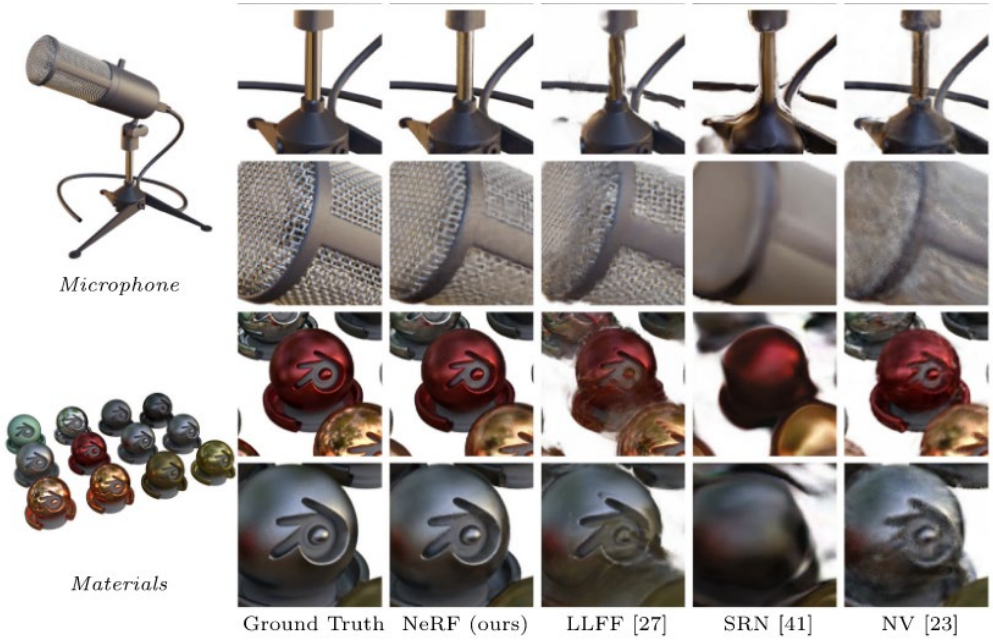
$$T(t) = \exp\left(-\int_{t_1}^{t} \sigma\big(\mathbf{r}(u)\big) \cdot du\right).$$

Mildenhall, Ben, et al. "Nerf: Representing scenes as neural radiance fields for view synthesis." ECCV. 2020.

Implicit Representation: Density

- NeRF (ECCV 2020)
  - Performance

Method	Diffuse Synthetic 360° [40]			Realistic Synthetic 360°			Real Forward-Facing [27]		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SRN [41]	33.20	0.963	0.073	22.26	0.846	0.170	22.84	0.668	0.378
NV [23]	29.62	0.929	0.099	26.05	0.893	0.160	–	–	–
LLFF [27]	34.38	0.985	0.048	24.88	0.911	0.114	24.13	0.798	**0.212**
Ours	**40.15**	**0.991**	**0.023**	**31.01**	**0.947**	**0.081**	**26.50**	**0.811**	0.250



Mildenhall, Ben, et al. "Nerf: Representing scenes as neural radiance fields for view synthesis." ECCV. 2020.

Implicit Representation: Density

- NeuS (NIPS 2021)
  - Motivation:
    - NeRF-based methods struggle to extract high-quality surfaces from learned density.
    - Conventional volume rendering method causes inherent geometric errors for surface reconstruction.
  - NeuS represent a surface as the zero-level set of a SDF and develop a new volume rendering method to train the neural SDF, decreasing the inherent geometric errors.
  - The S-density of the scene is defined as

$$\phi_s(f(\mathbf{x})),$$

where ϕ_s is the logistic density distribution, the derivative of the Sigmoid function.

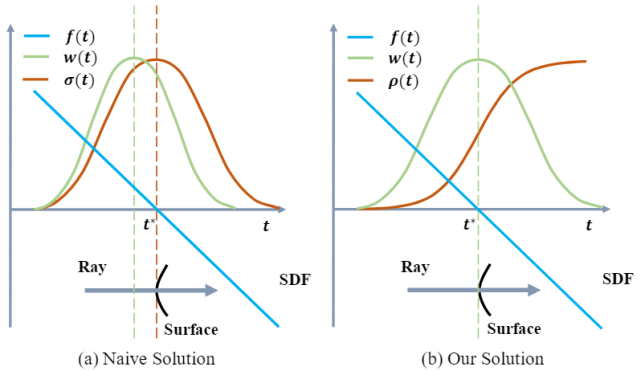


Figure 2: Illustration of (a) weight bias of naive solution, and (b) the weight function defined in our solution, which is unbiased in the first-order approximation of SDF.

Wang, Peng, et al. "Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction." NIPS. 2021.

Implicit Representation: Density

- NeuS (NIPS 2021)
  - Performance

	w/ mask			w/o mask			
ScanID	IDR	NeRF	Ours	COLMAP	NeRF	UNISURF	Ours
scan24	1.63	1.83	**0.83**	**0.81**	1.90	1.32	1.00
scan37	1.87	2.39	**0.98**	2.05	1.60	**1.36**	1.37
scan40	0.63	1.79	**0.56**	**0.73**	1.85	1.72	0.93
scan55	0.48	0.66	**0.37**	1.22	0.58	0.44	**0.43**
scan63	**1.04**	1.79	1.13	1.79	2.28	1.35	**1.10**
scan65	0.79	1.44	**0.59**	1.58	1.27	0.79	**0.65**
scan69	0.77	1.50	**0.60**	1.02	1.47	0.80	**0.57**
scan83	1.33	**1.20**	1.45	3.05	1.67	1.49	**1.48**
scan97	1.16	1.96	**0.95**	1.40	2.05	1.37	**1.09**
scan105	**0.76**	1.27	0.78	2.05	1.07	0.89	**0.83**
scan106	0.67	1.44	**0.52**	1.00	0.88	0.59	**0.52**
scan110	**0.90**	2.61	1.43	1.32	2.53	1.47	**1.20**
scan114	0.42	1.04	**0.36**	0.49	1.06	0.46	**0.35**
scan118	0.51	1.13	**0.45**	0.78	1.15	0.59	**0.49**
scan122	0.53	0.99	**0.45**	1.17	0.96	0.62	**0.54**
mean	0.90	1.54	**0.77**	1.36	1.49	1.02	**0.84**

Table 1: Quantitative evaluation on DTU dataset. COLMAP results are achieved by trim=0.

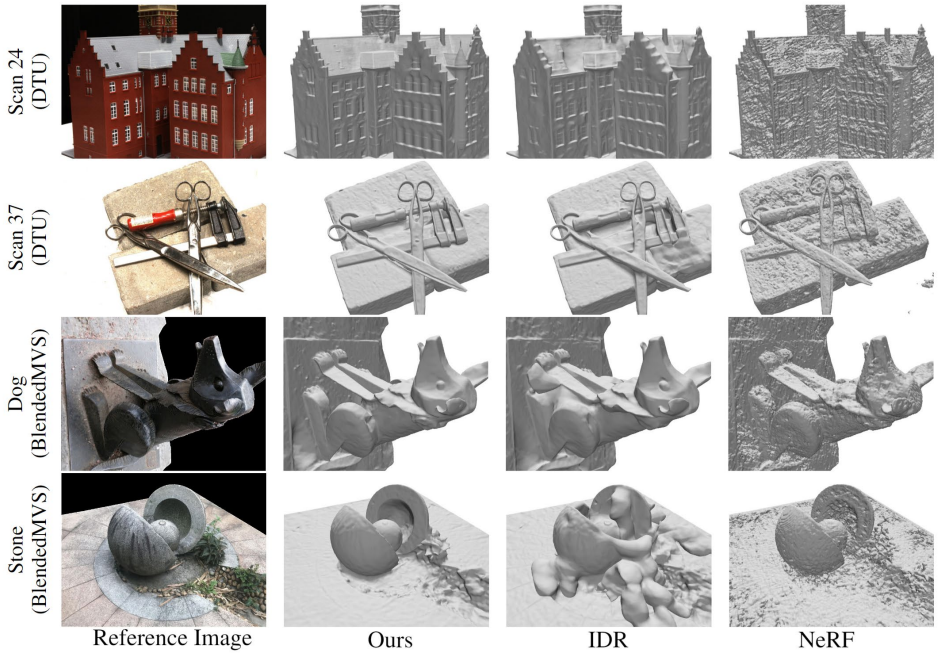
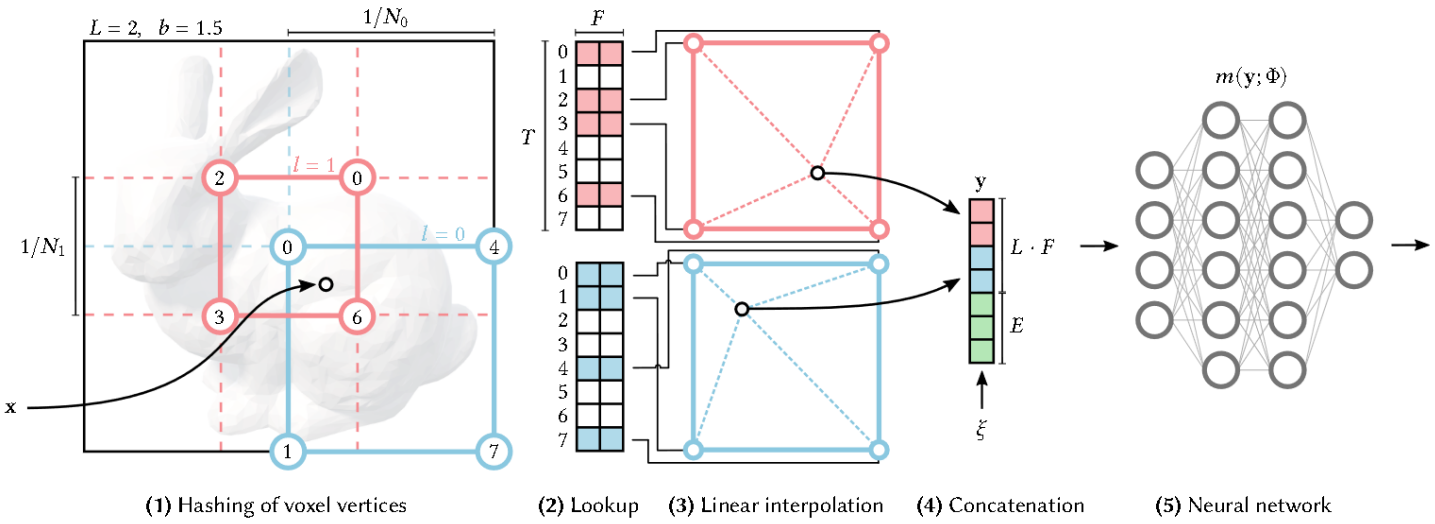


Figure 4: Comparions on surface reconstruction with mask supervision.

Wang, Peng, et al. "Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction." NIPS. 2021.

Implicit Representation: Density

- Instant-NGP (TOG 2022)
  - Motivation: Utilizing MLPs to simulate the density of a scene is costly to train and evaluation.
  - Instant-NGP utilizes multi-scale spatial feature volume and small network to accelerate the implementation.
  - Hash encoding is adaptive and efficient, boosting the whole processing.



Müller, Thomas, et al. "Instant neural graphics primitives with a multiresolution hash encoding." TOG. 2022.

Implicit Representation: Density

- Instant-NGP (TOG 2022)
  - Performance

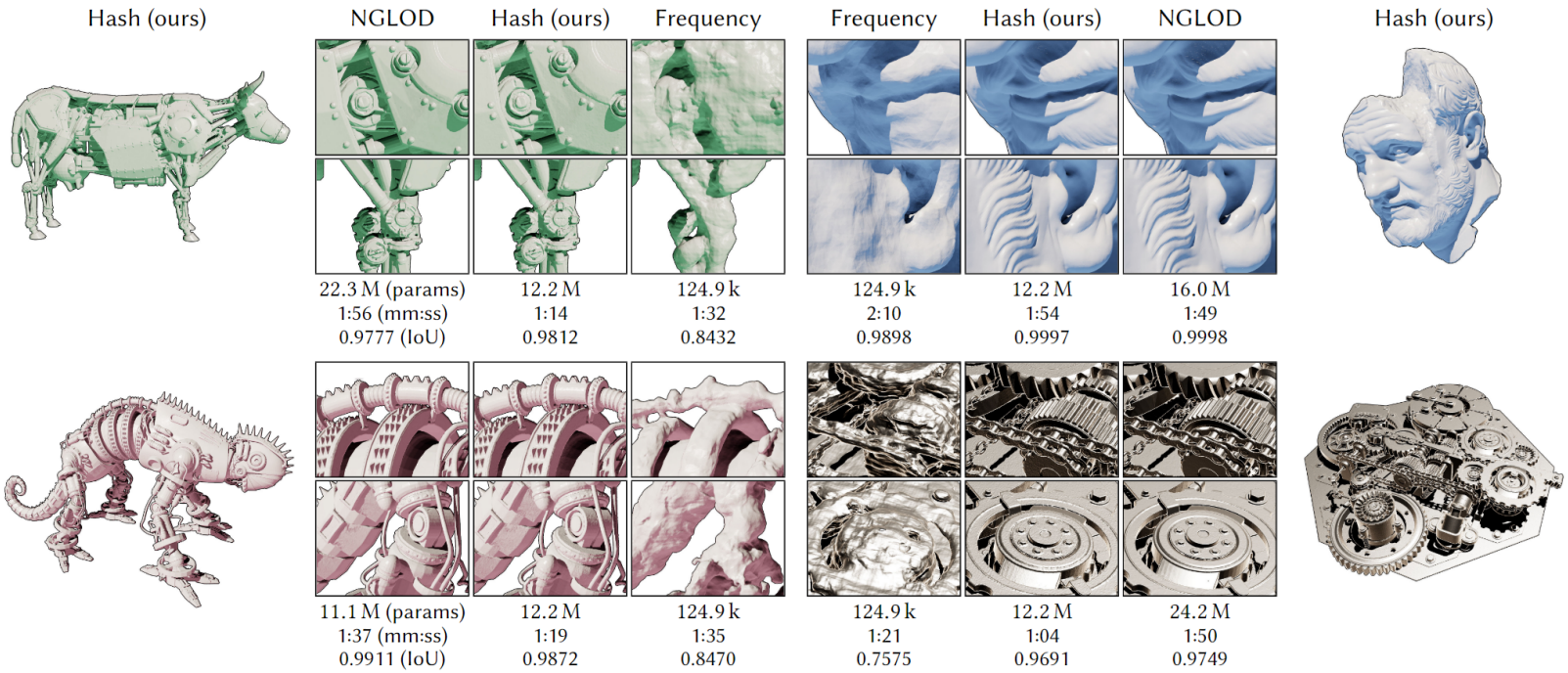
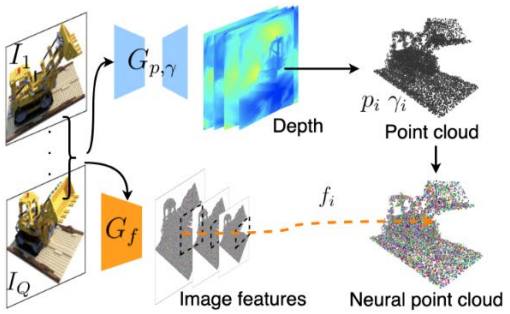


Fig. 7. Neural signed distance functions trained for 11 000 steps. The frequency encoding [Mildenhall et al. 2020] struggles to capture the sharp details on these intricate models. NGLOD [Takikawa et al. 2021] achieves the highest visual quality, at the cost of only training the SDF inside the cells of a close-fitting octree. Our hash encoding exhibits similar numeric quality in terms of intersection over union (IoU) and can be evaluated anywhere in the scene. However, it also exhibits visually undesirable surface roughness that we attribute to randomly distributed hash collisions. Bearded Man ©Oliver Laric (CC BY-NC-SA 2.0)

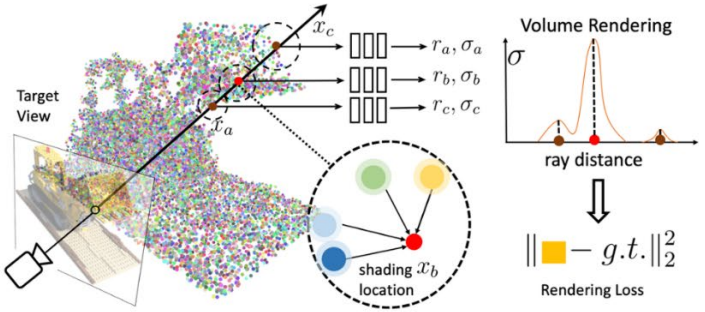
Müller, Thomas, et al. "Instant neural graphics primitives with a multiresolution hash encoding." TOG. 2022.

Implicit Representation: Density

- Point-NeRF (CVPR 2022)
  - Motivation:
    - NeRF could only be optimized per-scene, leading to prohibitive reconstruction time.
    - Deep multi-view stereo methods can quickly reconstruct scene geometry .
  - Point-NeRF combines the advantages of these two approaches by using neural point clouds with neural features to model a radiance field.
  - With the initialized point cloud, Point-NeRF could achieve more than 30 times faster training.



(a) Neural Point Generation.



(b) Point-NeRF Representation with Volume Rendering.

Xu, Qiangeng, et al. "Point-nerf: Point-based neural radiance fields." CVPR. 2022.

Implicit Representation: Density

- Point-NeRF (CVPR 2022)
  - Performance

	No Per-scene Optimization				Per-scene Optimization				
	PixelNeRF [63]	MVSNeRF [8]	IBRNet [53]	Ours	Ours_{1K}	Ours_{10K}	MVSNeRF_{10K}	IBRNet_{10K}	NeRF_{200k}
PSNR ↑	19.31	26.63	26.04	23.89	28.43	30.12	28.50	**31.35**	27.01
SSIM ↑	0.789	0.931	0.917	0.874	0.929	**0.957**	0.933	0.956	0.902
LPIPS_{Vgg} ↓	0.382	0.168	0.190	0.203	0.183	**0.117**	0.179	0.131	0.263
Time↓	-	-	-	-	**2min**	20min	24min	1h	10h

Table 1. Comparisons of our Point-NeRF with radiance-based models [29, 32, 53] and a point-based rendering model [2] on the DTU dataset [18] with the novel view synthesis setting introduced in [8]. The subscripts indicate the number of iterations during optimization.

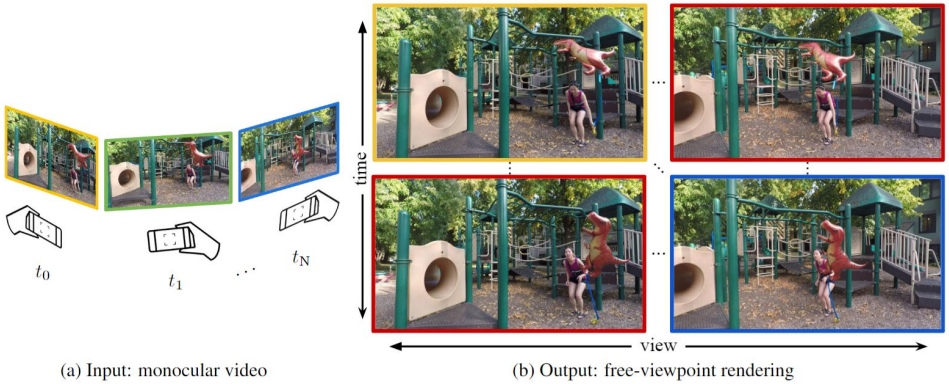
	NPBG [2]	NeRF [32]	IBRNet [53]	NSVF [29]	$\text{Point-NeRF}^{col}_{200K}$	Point-NeRF_{20K}	Point-NeRF_{200K}
PSNR ↑	24.56	31.01	28.14	31.75	31.77	30.71	**33.31**
SSIM ↑	0.923	0.947	0.942	0.964	0.973	0.967	**0.978**
LPIPS_{Vgg} ↓	0.109	0.081	0.072	-	0.062	0.081	**0.049**
LPIPS_{Alex} ↓	0.095	-	-	0.047	0.040	0.050	**0.027**

Table 2. Comparisons of Point-NeRF with radiance-based models [29,32,53] and a point-based rendering model [2] on the Synthetic-NeRF dataset [32]. The subscripts indicate the number of iterations. Our model not only surpasses other methods when converged after $200K$ steps (Point-NeRF_{200K}), but surpasses IBRNet [53] and is on par with NeRF [35] when optimized by only $20K$ steps (Point-NeRF_{20K}). Our methods can also initialize radiance fields based on point clouds reconstructed by methods such as COLMAP ($\text{Point-NeRF}^{col}_{200K}$).

Xu, Qiangeng, et al. "Point-nerf: Point-based neural radiance fields." CVPR. 2022.

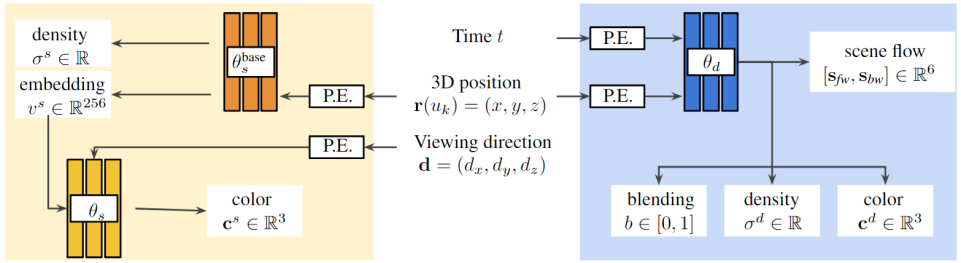
Implicit Representation: Density

- Dynamic NeRF (ICCV 2021)



(a) Input: monocular video (b) Output: free-viewpoint rendering

  - Dynamic NeRF jointly trains a static NeRF and a time-varying dynamic NeRF, and learn how to render the results in an unsupervised manner.
  - To solve the ill-posed issue of the implicit function from a single video, regularization losses are developed to encourage a more physically plausible solution.



(a) Static NeRF (Section 3.2) (b) Dynamic NeRF (Section 3.3)

Gao, Chen, et al. "Dynamic view synthesis from dynamic monocular video." ICCV. 2021.

Implicit Representation: Density

- Dynamic NeRF (ICCV 2021)
  - Spatial flow between different time frames

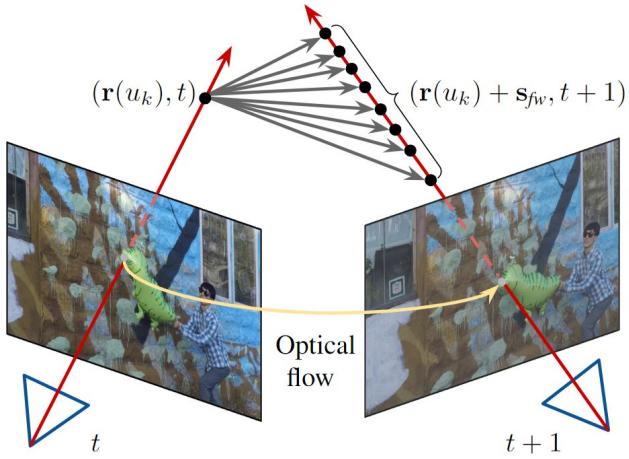
$$\left(\mathbf{s}_{fw}, \mathbf{s}_{bw}, \sigma_t^d, \mathbf{c}_t^d, b\right) = \mathrm{MLP}_{\theta_d}\left(\mathbf{r}(u_k), t\right)$$

Flow Density Color Blending Weight

$$\left(\sigma_{t+1}^d,\ \mathbf{c}_{t+1}^d\right) = \mathrm{MLP}_{\theta_d}(\mathbf{r}(u_k) + \mathbf{s}_{fw},\ t+1)$$

$$\left(\sigma_{t-1}^d,\ \mathbf{c}_{t-1}^d\right) = \mathrm{MLP}_{\theta_d}(\mathbf{r}(u_k) + \mathbf{s}_{bw},\ t-1)$$

$$\mathbf{C}_{t'}^d(\mathbf{r}) = \sum_{k=1}^{K} T_{t'}^d(u_k)\, \alpha^d(\sigma_{t'}^d(u_k)\, \delta_k)\, \mathbf{c}_{t'}^d(u_k)$$



  - Loss Functions

$$\mathcal{L}_{dyn} = \sum_{t' \in \{t, t-1, t+1\}} \sum_{ij} \left\| \left(\mathbf{C}_{t'}^d(\mathbf{r}_{ij}) - \mathbf{C}^{gt}(\mathbf{r}_{ij})\right) \right\|_2^2$$

$$\mathcal{L}_{slow} = \sum_{ij} \left\| \mathbf{s}_{fw}(\mathbf{r}_{ij}) \right\|_1 + \left\| \mathbf{s}_{bw}(\mathbf{r}_{ij}) \right\|_1$$

$$\mathcal{L}_{smooth} = \sum_{ij} \left\| \mathbf{s}_{fw}(\mathbf{r}_{ij}) + \mathbf{s}_{bw}(\mathbf{r}_{ij}) \right\|_2^2$$

$$\mathcal{L}_{cyc} = \sum \left\| \mathbf{s}_{fw}(\mathbf{r}, t) + \mathbf{s}_{bw}(\mathbf{r} + \mathbf{s}_{fw}(\mathbf{r}, t),\ t+1) \right\|_2^2 + \left\| \mathbf{s}_{bw}(\mathbf{r}, t) + \mathbf{s}_{fw}(\mathbf{r} + \mathbf{s}_{bw}(\mathbf{r}, t),\ t-1) \right\|_2^2$$

$$\mathcal{L}_{depth} = \sum_{ij} \left\| \overline{\mathbf{D}^d}(\mathbf{r}_{ij}) - \overline{\mathbf{D}^{gt}}(\mathbf{r}_{ij}) \right\|_2^2 + \left\| (\mathbf{D}^d(\mathbf{r}_{ij}) - \mathbf{D}^s(\mathbf{r}_{ij})) \cdot (1 - \mathbf{M}(\mathbf{r}_{ij})) \right\|_2^2,$$

Gao, Chen, et al. "Dynamic view synthesis from dynamic monocular video." ICCV. 2021.

Implicit Representation: Density

- Dynamic NeRF (ICCV 2021)
  - Performance

Table 1. **Novel view synthesis results.** We report the average PSNR and LPIPS results with comparisons to existing methods on Dynamic Scene dataset [62]. The best performance is in **bold** and the second best is underscored.

PSNR ↑ / LPIPS ↓	Jumping	Skating	Truck	Umbrella	Balloon1	Balloon2	Playground	Average
NeRF	20.58 / 0.305	23.05 / 0.316	22.61 / 0.225	21.08 / 0.441	19.07 / 0.214	24.08 / 0.098	20.86 / 0.164	21.62 / 0.252
NeRF + time	16.72 / 0.489	19.23 / 0.542	17.17 / 0.403	17.17 / 0.752	17.33 / 0.304	19.67 / 0.236	13.80 / 0.444	17.30 / 0.453
Yoon et al. [62]	20.16 / 0.148	21.75 / 0.135	23.93 / **0.109**	20.35 / 0.179	18.76 / 0.178	19.89 / 0.138	15.09 / 0.183	19.99 / 0.153
Tretschk et al. [55]	19.38 / 0.295	23.29 / 0.234	19.02 / 0.453	19.26 / 0.427	16.98 / 0.353	22.23 / 0.212	14.24 / 0.336	19.20 / 0.330
Li et al. [28]	24.12 / 0.156	**28.91** / 0.135	**25.94** / 0.171	22.58 / 0.302	21.40 / 0.225	24.09 / 0.228	20.91 / 0.220	23.99 / 0.205
Ours	**24.23** / **0.144**	28.90 / **0.124**	25.78 / 0.134	**23.15** / **0.146**	**21.47** / **0.125**	**25.97** / **0.059**	**23.65** / **0.093**	**24.74** / **0.118**

NeRF + time | Yoon et al. [62] | Tretschk et al. [55] | Li et al. [28] | Ours | Ground truth

Figure 8. **Novel view synthesis.** Our model enables the free-viewpoint synthesis of a dynamic scene. Compared with Yoon et al. [62], our results appear slightly blurry (because we reconstruct the entire frame as opposed to warp and blend input images), but align with the ground truth image better and create smoother view-interpolation results. When compared to other NeRF-based methods, our results are sharper and closer to the ground truth. Please refer to the supplementary material for video results.

Gao, Chen, et al. "Dynamic view synthesis from dynamic monocular video." ICCV. 2021.

Implicit Representation: Density

- 3DGS (TOG 2023)

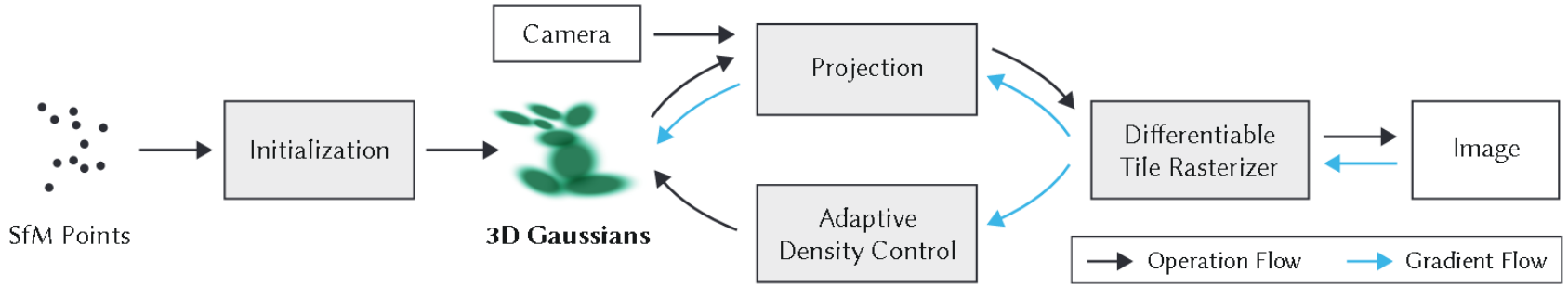
The geometry is modelled as a set of 3D Gaussians, which are defined by a 3D **covariance** matrix Σ at center point μ in world space,

$$G(\mathbf{x}) = \exp\left(-\frac{1}{2}\mathbf{x}^T\Sigma^{-1}\mathbf{x}\right),\ \Sigma = \mathbf{RSS}^T\mathbf{R}^T,$$

where $\mathbf{S}$ is **scaling** matrix and $\mathbf{R}$ is **rotation** matrix. In the rendering processing, $G(\mathbf{x})$ is multiplied by α.

The **color** of each Gaussian is represented via **spherical harmonics** (**SH**).

In summary, each Gaussian has the following parameters, position μ, covariance Σ, factor α, and SH coefficients.



The color C along a ray is given by

$$C = \sum_{i=1}^{N} T_i \alpha_i \mathbf{c}_i, \qquad \alpha_i = (1 - \exp(-\sigma_i \delta_i)) \text{ and } T_i = \prod_{j=1}^{i-1}(1 - \alpha_i).$$

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." TOG. 2023.

Implicit Representation: Density

- 3DGS (TOG 2023)
  - Performance

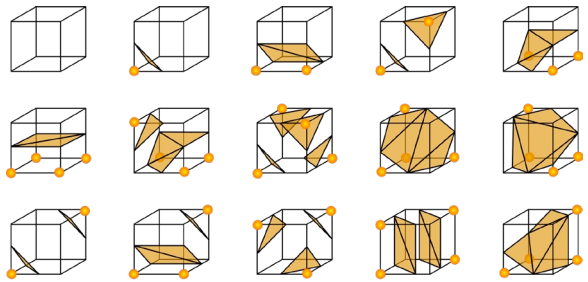
Table 1. Quantitative evaluation of our method compared to previous work, computed over three datasets. Results marked with dagger † have been directly adopted from the original paper, all others were obtained in our own experiments.

Dataset	Mip-NeRF360						Tanks&Temples						Deep Blending					
Method|Metric	*SSIM*↑	*PSNR*↑	*LPIPS*↓	Train	FPS	Mem	*SSIM*↑	*PSNR*↑	*LPIPS*↓	Train	FPS	Mem	*SSIM*↑	*PSNR*↑	*LPIPS*↓	Train	FPS	Mem
Plenoxels	0.626	23.08	0.463	25m49s	6.79	2.1GB	0.719	21.08	0.379	25m5s	13.0	2.3GB	0.795	23.06	0.510	27m49s	11.2	2.7GB
INGP-Base	0.671	25.30	0.371	5m37s	11.7	13MB	0.723	21.72	0.330	5m26s	17.1	13MB	0.797	23.62	0.423	6m31s	3.26	13MB
INGP-Big	0.699	25.59	0.331	7m30s	9.43	48MB	0.745	21.92	0.305	6m59s	14.4	48MB	0.817	24.96	0.390	8m	2.79	48MB
M-NeRF360	0.792†	27.69†	0.237†	48h	0.06	8.6MB	0.759	22.22	0.257	48h	0.14	8.6MB	0.901	29.40	0.245	48h	0.09	8.6MB
Ours-7K	0.770	25.60	0.279	6m25s	160	523MB	0.767	21.20	0.280	6m55s	197	270MB	0.875	27.78	0.317	4m35s	172	386MB
Ours-30K	0.815	27.21	0.214	41m33s	134	734MB	0.841	23.14	0.183	26m54s	154	411MB	0.903	29.41	0.243	36m2s	137	676MB

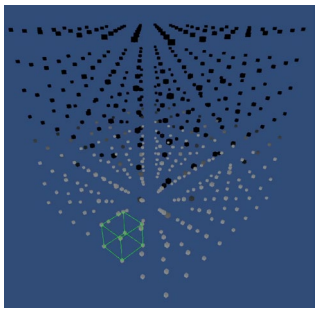
Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." TOG. 2023.

Surface Extraction

- Marching Cubes
  - Marching Cubes (MC) is the most widely used method to extract the polygonal mesh, which is represented as an isosurface of a 3D scalar field.
  - The whole space is divided into non-overlapping cubes, and the polygons in each cube has 256 conditions according to the occupancy of 8 corners (2^8=256). These conditions can be summarized into 15 key conditions.
  - During the implementation, each cube is processed separately by determining which condition the cube belongs to according to the occupancy of its corners.
  - Iterate over all cubes to extract the whole polygons.



The 15 key conditions of Marching Cubes.

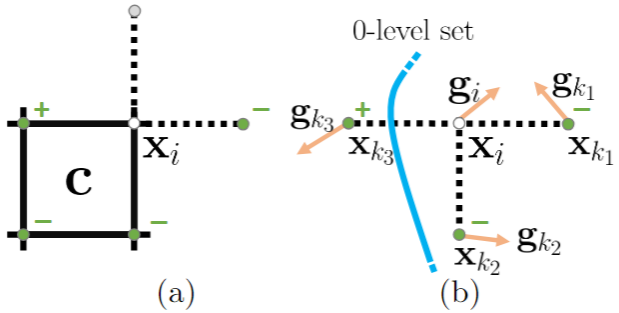


Process each cube literately.

Lorensen, William E., and Harvey E. Cline. "Marching cubes: A high resolution 3D surface construction algorithm." Seminal graphics: pioneering efforts that shaped the field. 1998.

Surface Extraction

- MeshUDF (ECCV 2022)
  - Motivation: Traditional MC can only extract watertight surfaces from BOFs and SDFs, it cannot recover more general surfaces from UDF.
  - MeshUDF is a fast and differentiable method to extract surfaces from UDFs. It utilizes the UDF gradient to judge the occupancy of each cube corner, making it could be fed into Marching Cubes.



  - $\mathbf{x}$ is the corner of cube and $\mathbf{g}$ is the corresponding UDF grad. In this example, $\mathbf{g}\_i \cdot \mathbf{g}\_k1 > 0$, so $\mathbf{x}\_i$ and $\mathbf{x}\_k1$ has the same signs. On the contrary, $\mathbf{g}\_i \cdot \mathbf{g}\_k3 < 0$, so $\mathbf{x}\_i$ and $\mathbf{x}\_k3$ has the difference signs and the surface interact with edge $\mathbf{x}\_i$ $\mathbf{x}\_k3$.

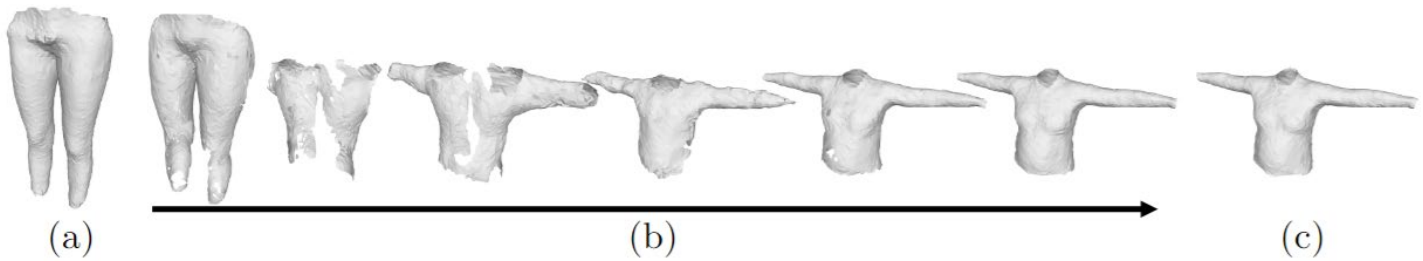
Guillard, Benoit, Federico Stella, and Pascal Fua. "Meshudf: Fast and differentiable meshing of unsigned distance field networks." ECCV. 2022.

Surface Extraction

- MeshUDF (ECCV 2022)
  - Performance

Table 1: **Comparing UDF meshing methods.** Average Chamfer (CHD), image consistency (IC), normal consistency (NC) and processing time for 300 garments (left) and 300 ShapeNet cars (right). We use a single UDF network in each case and only change the meshing procedure. For BP, we decompose the time into sampling and meshing times.

	Garments, ϕ_θ network			**Cars, NDF network [11]**		
	BP	*Inflation*	*Ours*	*BP*	*Inflation*	*Ours*
CHD (↓)	1.62	3.00	**1.51**	6.84	11.24	**6.63**
IC (%, ↑)	92.51	88.48	**92.80**	90.50	87.09	**90.87**
NC (%, ↑)	89.50	94.16	**95.50**	61.50	**73.19**	70.38
Time (↓)	16.5s + 3000s	**1.0 sec.**	1.2 sec.	24.7s + 8400s	**4.8 sec.**	7.1 sec.

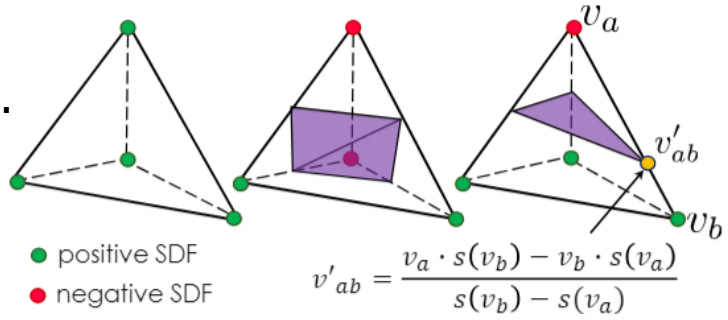


Shape Deformation.

Guillard, Benoit, Federico Stella, and Pascal Fua. "Meshudf: Fast and differentiable meshing of unsigned distance field networks." ECCV. 2022.

Surface Extraction

- DMTet (NIPS 2021)
  - Motivation:
    - Tetrahedron is much simpler than cube.
    - Fixed cube corners cannot handle detailed structures.
  - DMTet divides the whole space into tetrahedrons.
  - The corners of tetrahedrons are deformable to adopt the detailed structures.



Triangle conditions in a tetrahedron.

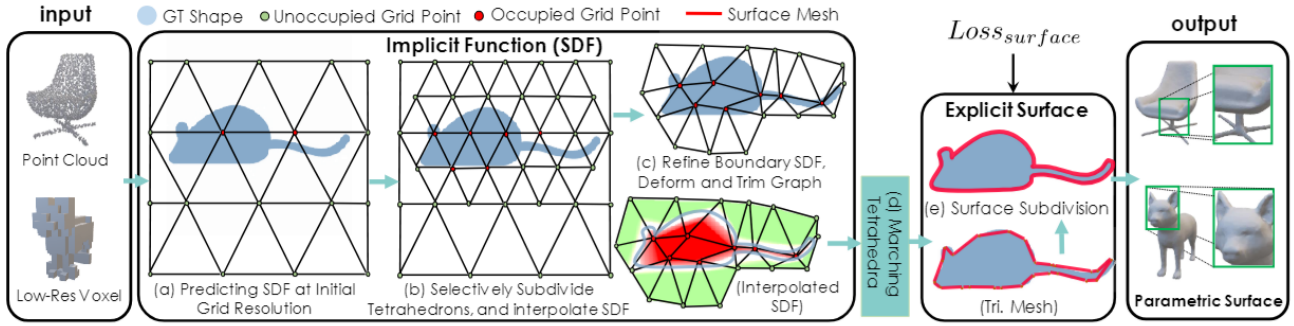


Figure 1: **DMTET** reconstructs the shape implicitly in a coarse-to-fine manner by predicting the SDF defined on a deformable tetrahedral grid. It then converts the SDF to a surface mesh by a differentiable Marching Tetrahedra layer. **DMTET is trained by optimizing the objective function defined on the final surface.**

Shen, Tianchang, et al. "Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis." NIPS. 2021.

Surface Extraction

- DMTet (NIPS 2021)
  - Performance

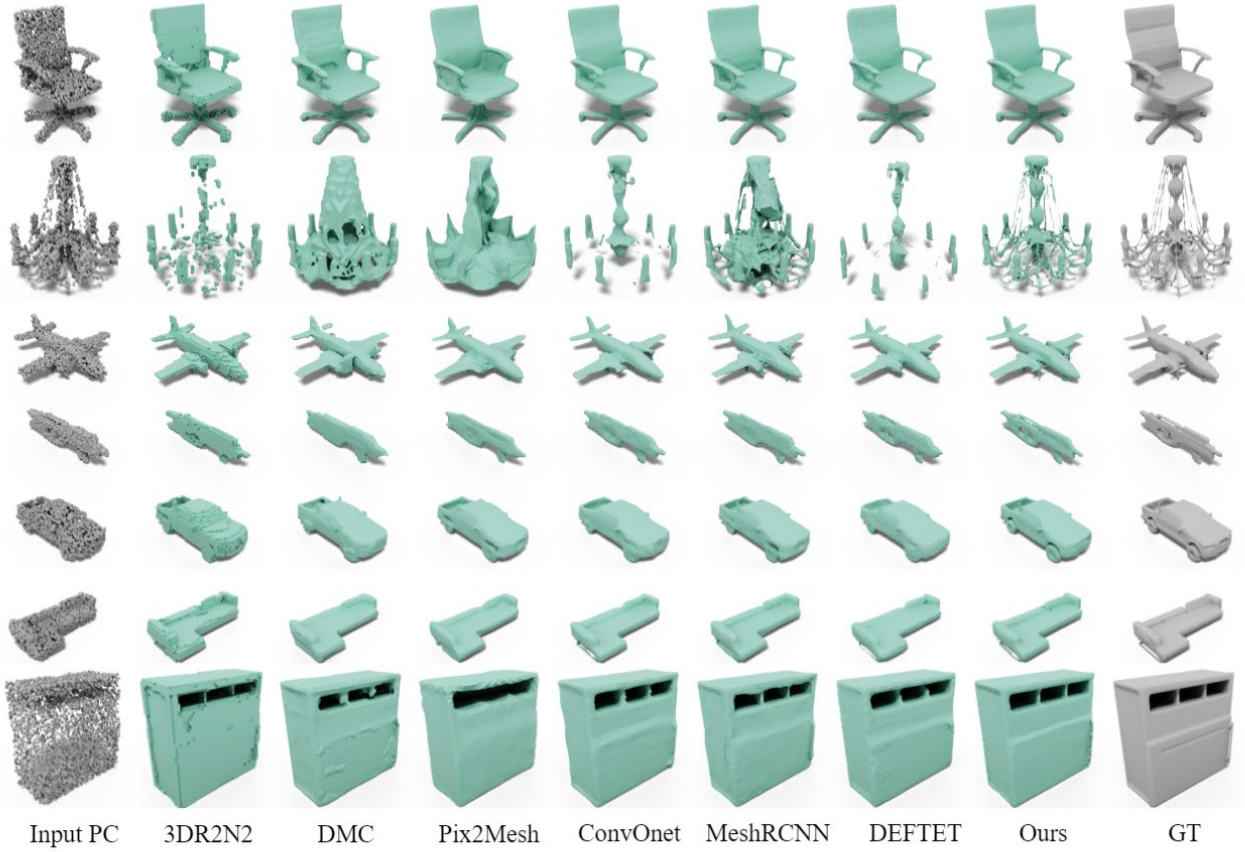


Figure 7: **Qualitative results on 3D Reconstruction from Point Clouds:** Our model reconstructs shapes with more geometric details compared to baselines.

Shen, Tianchang, et al. "Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis." NIPS. 2021.

Surface Extraction

- DMTet (NIPS 2021)
  - Performance

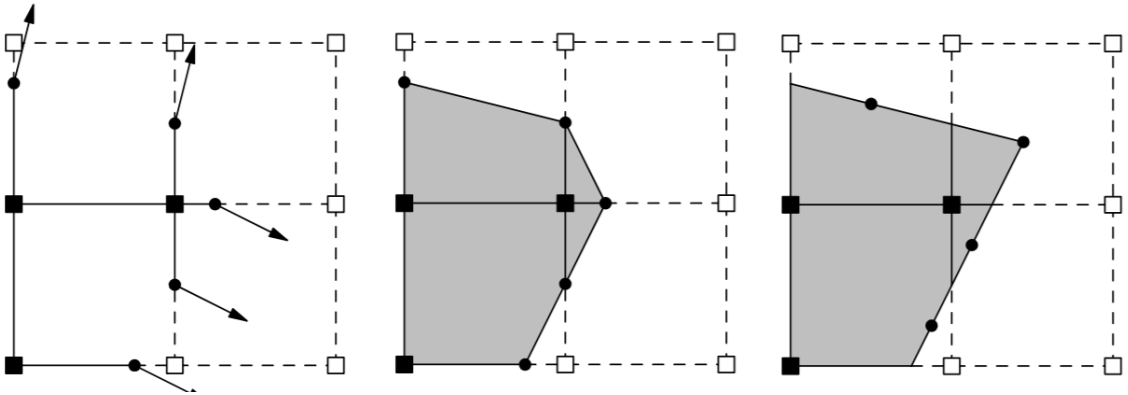
Category	Airplane	Bench	Dresser	Car	Chair	Display	Lamp	Speaker	Rifle	Sofa	Table	Phone	Vessel	**Mean↓**	**Time(ms)↓**
3D-R2N2 [10]	1.48	1.59	1.64	1.62	1.70	1.66	1.74	1.74	1.37	1.60	1.78	1.55	1.51	1.61	174
DMC [31]	1.57	1.47	1.29	1.67	1.44	1.25	2.15	1.49	1.45	1.19	1.33	0.88	1.70	1.45	349
Pixel2mesh [54]	0.98	1.28	1.44	1.19	1.91	1.25	2.07	1.61	0.91	1.15	1.82	0.83	1.12	1.35	**30**
ConvOnet [44]	0.82	0.95	0.96	1.12	1.03	0.93	1.22	1.12	0.79	0.91	0.94	0.67	0.99	0.95	866
MeshRCNN [22]	0.88	1.01	1.05	1.14	1.10	0.99	1.20	1.21	0.83	0.96	1.00	0.71	1.03	1.01	228
DEFTET [18]	0.85	0.94	0.97	1.13	1.04	0.92	1.28	1.17	0.85	0.90	0.93	0.65	0.99	0.97	61
DMTET wo (Def, Vol., Surf.)	0.82	0.96	0.94	0.98	0.99	0.90	1.04	1.03	0.80	0.86	0.93	0.65	0.89	0.91	52
DMTET wo (Vol., Surf.)	0.69	0.82	0.88	0.92	0.92	0.82	0.89	0.97	0.65	0.81	0.84	0.61	0.80	0.81	52
DMTET wo Vol.	0.65	0.78	0.84	0.89	0.89	0.79	0.86	0.95	0.61	0.78	0.79	0.60	0.78	0.79	67
DMTET wo Surf.	0.63	0.77	0.84	0.88	**0.88**	0.79	**0.84**	**0.94**	0.60	0.78	0.79	0.59	**0.76**	0.78	108
DMTET	**0.62**	**0.76**	**0.83**	**0.87**	**0.88**	**0.78**	**0.84**	**0.94**	**0.59**	**0.77**	**0.78**	**0.57**	**0.76**	**0.77**	129

Table 3: Quantitative Results on **Point Cloud Reconstruction** (Chamfer L1). Note that all the networks in the baselines are not designed for this task, and thus we use the same encoder and their decoder for a fair comparison. We also ablate ourselves by operating on fixed grid (DMTET wo (Def, Vol., Surf.)), removing volume subdivision (DMTET wo Vol.), or surface subdivision (DMTET wo Surf.), or the both (DMTET wo (Vol., Surf.)).

Shen, Tianchang, et al. "Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis." NIPS. 2021.

Surface Extraction

- Dual Contouring
  - Motivation: Traditional MC cannot handle sharp structures of the surface.



Implicit Field and its Gradient | Marching Cubes | Dual Contouring

  - Dual Contouring (DC) utilizes gradient information to build Quadratic Error Function (QEF) in each cube to optimize the intersection point of the surface.

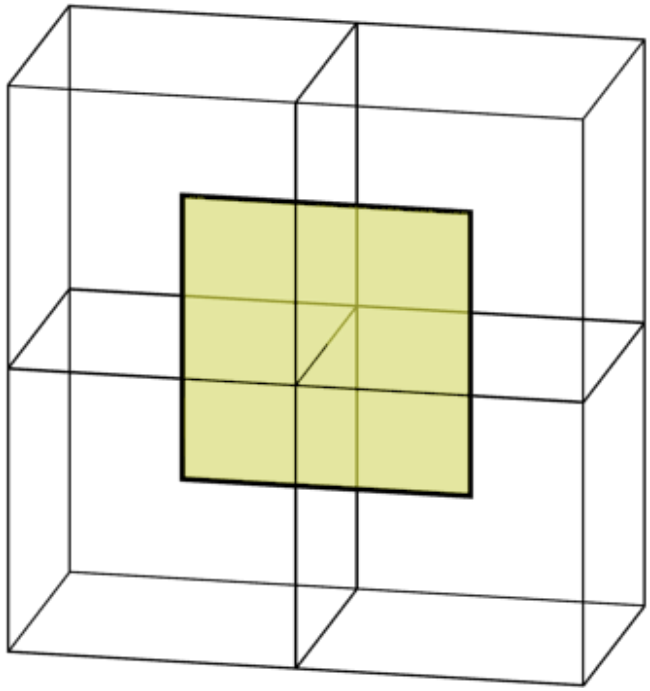
$$E(\mathbf{x}) = \sum_i (\mathbf{n}_i \cdot (\mathbf{x} - \mathbf{p}_i)),$$

where i is the index of edges interacting with the surface, $\mathbf{p}\_i$ and $\mathbf{n}\_i$ represent the interaction point and gradient, respectively.

Ju, Tao, et al. "Dual contouring of hermite data." Proceedings of the 29th annual conference on Computer graphics and interactive techniques. 2002.

Surface Extraction

- Dual Contouring
  - For each edge that exhibits the intersection with the surface, generate a quad connecting the minimizing vertices of the four cubes containing the edge.

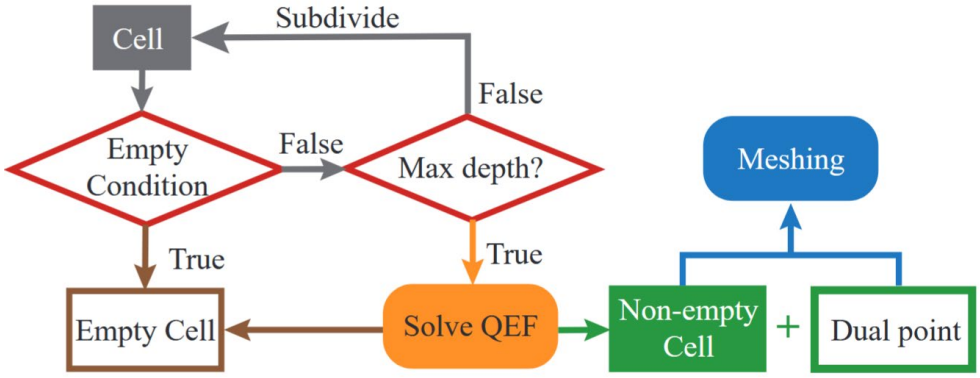


Ju, Tao, et al. "Dual contouring of hermite data." Proceedings of the 29th annual conference on Computer graphics and interactive techniques. 2002.

Surface Extraction

- DualUDF (ICCV 2023)
  - Motivation:
    - Dual Contouring can only extract watertight surfaces.
    - A neural UDF typically has substantial approximation errors, especially near the surfaces.
  - DualUDF utilizes an adaptive Octree to accelerate the implementation.
  - QEF is modified be compatible with UDF.

$$\mathbf{v} = \arg\min_{\mathbf{x}} \sum_{i=1}^{m} \left(\mathbf{n}_i \cdot (\mathbf{p}_i - \mathbf{x}) - d_i\right)^2$$



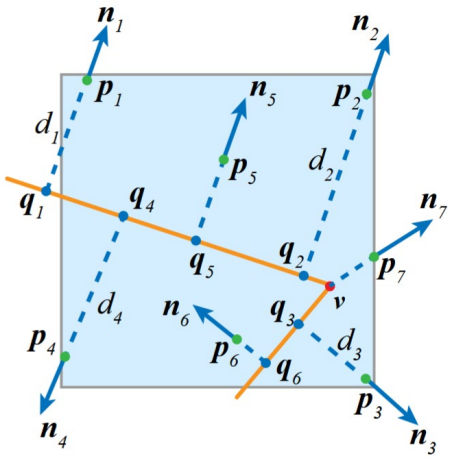


Figure 3: The pipeline of our mesh extraction method.

Zhang, Congyi, et al. "Surface extraction from neural unsigned distance fields." ICCV. 2023.

Surface Extraction

- DualUDF (ICCV 2023)
  - Performance

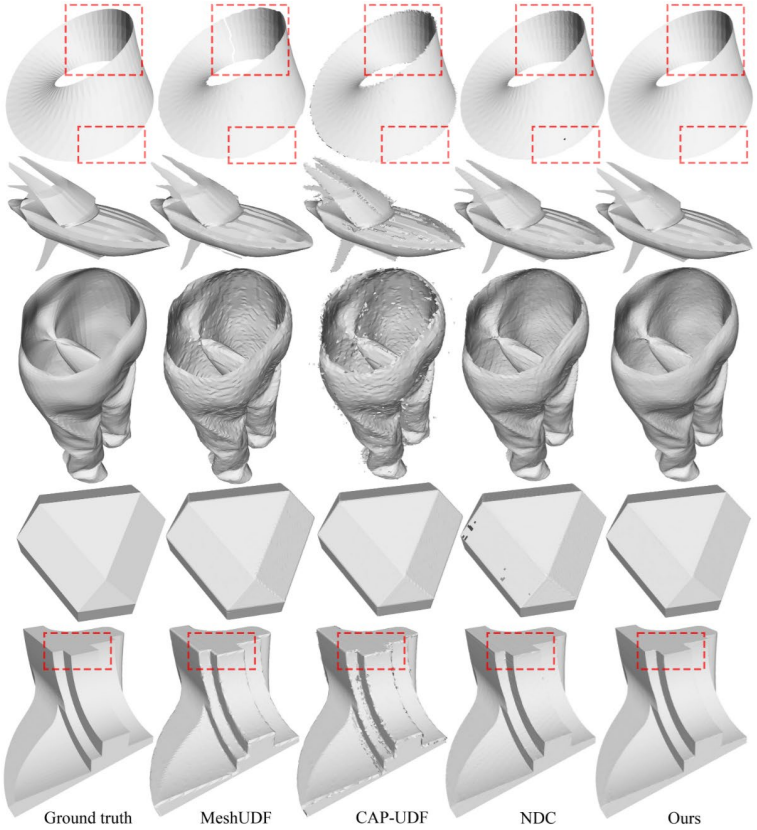


Figure 4: Meshes extracted from neural UDFs. We show 6 examples and compare our results to the mesh extraction results in *MeshUDF* [7], *CAP-UDF* [21] and *NDC* [3]. Our method preserves sharp geometric features better, yields results without undesirable holes, and reconstructs the original smooth boundaries of open surfaces.

Zhang, Congyi, et al. "Surface extraction from neural unsigned distance fields." ICCV. 2023.