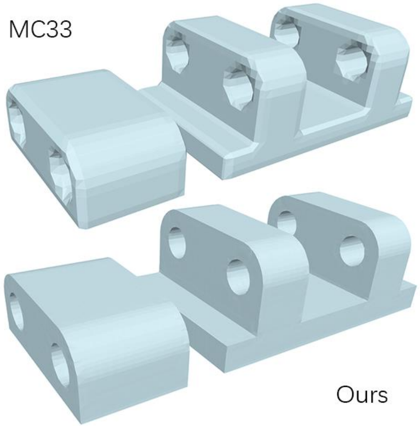
Neural Mesh Reconstruction

Zhiqin Chen

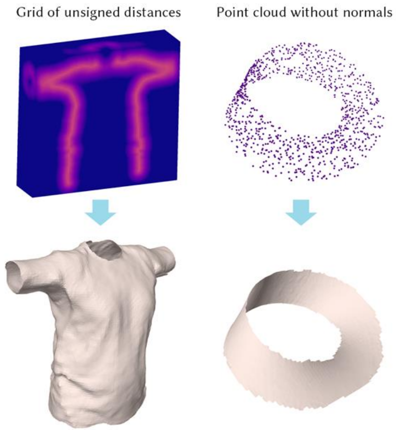
Simon Fraser University

Overview



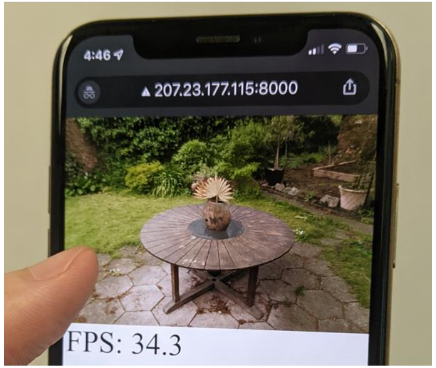
Neural Marching Cubes
(SIGGRAPH Asia 2021)

→



Neural Dual Contouring
(SIGGRAPH 2022)

→



MobileNeRF
(Arxiv 2022)

The inspiration

Neural implicit field:
The output shape is always smooth.

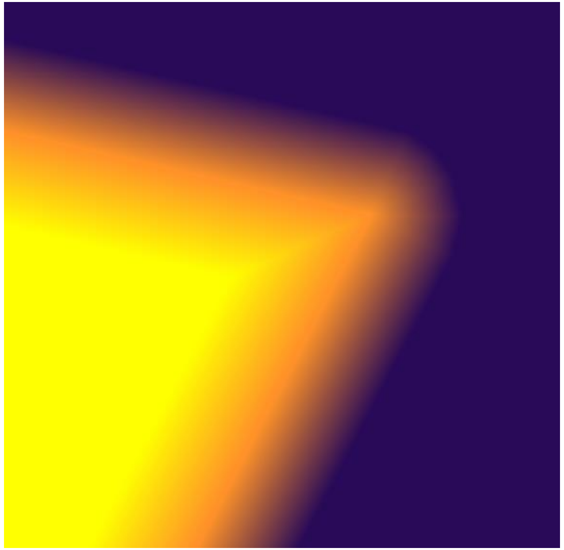
Reasons:

1. Properties of MLPs.

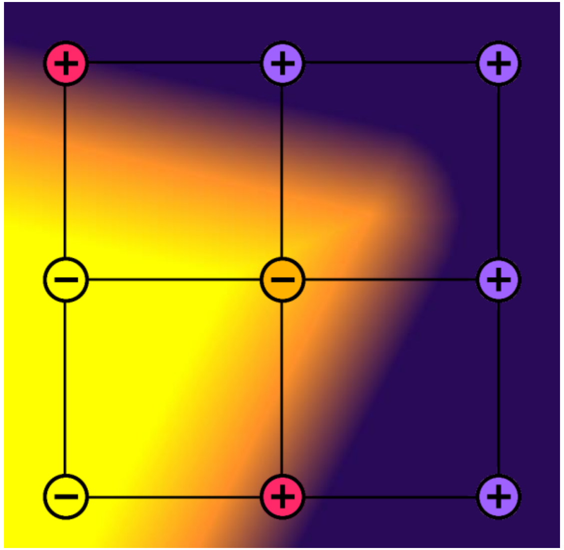
2. Marching Cubes cannot reconstruct sharp features.

[2] Marching cubes: A high resolution 3D surface construction algorithm. William E. Lorensen and Harvey E. Cline. SIGGRAPH 1987.

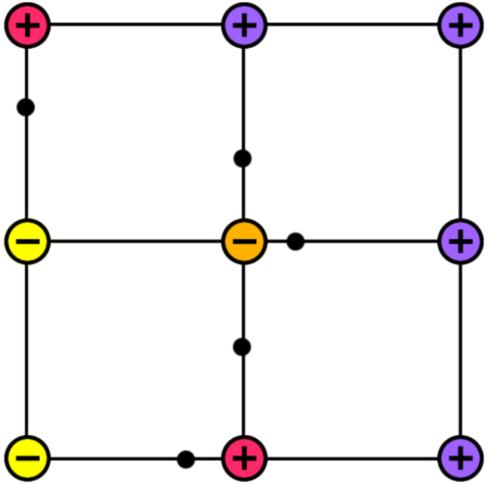
Marching Cubes



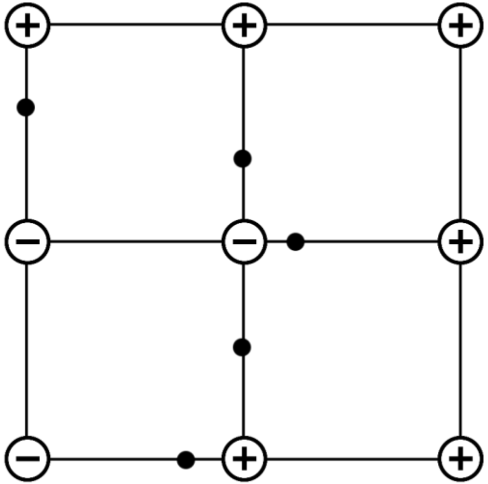
Marching Cubes



Marching Cubes

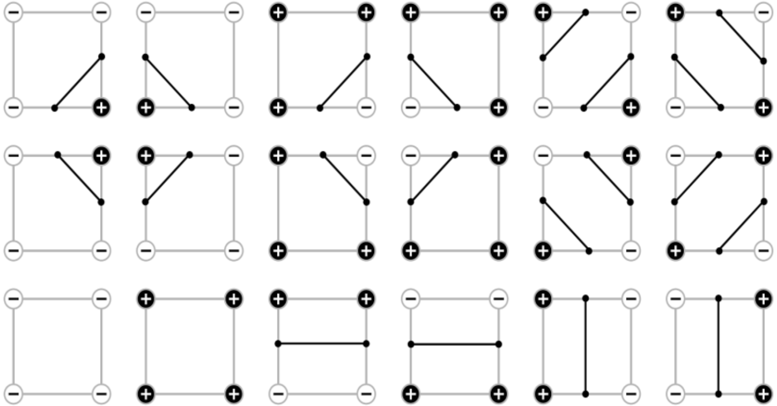


Marching Cubes

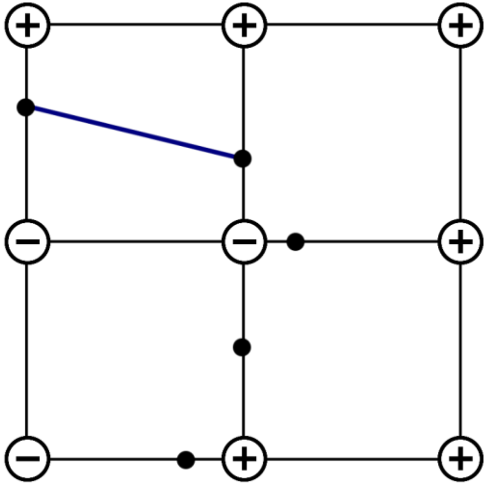


Look-up
table

(a) The face tessellations of Marching Cubes

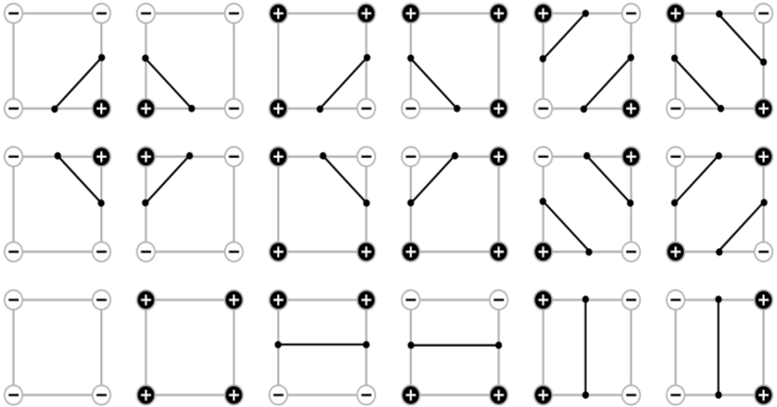


Marching Cubes

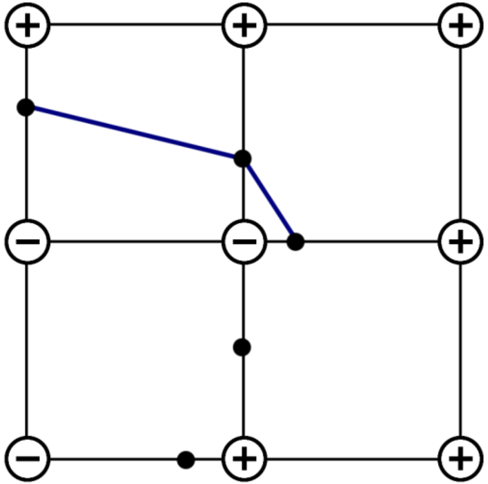


Look-up
table

(a) The face tessellations of Marching Cubes

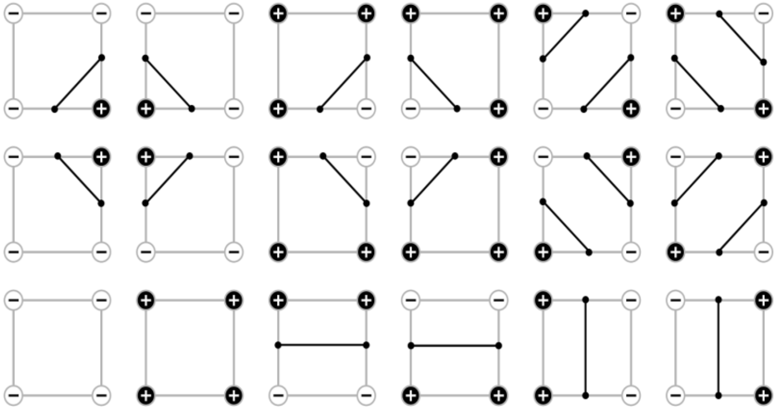


Marching Cubes

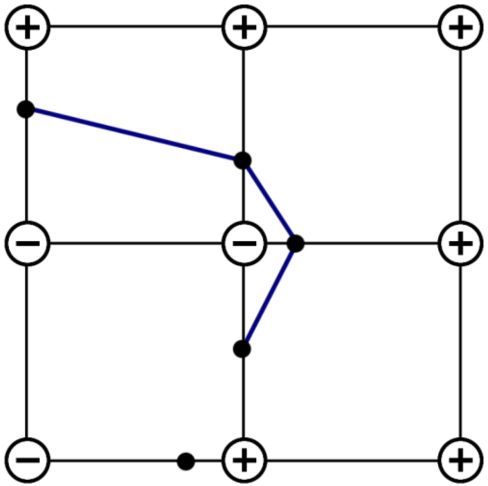


Look-up
table

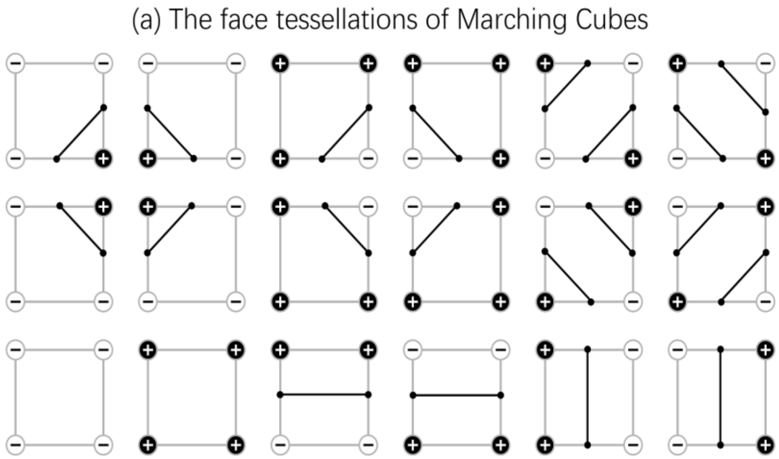
(a) The face tessellations of Marching Cubes



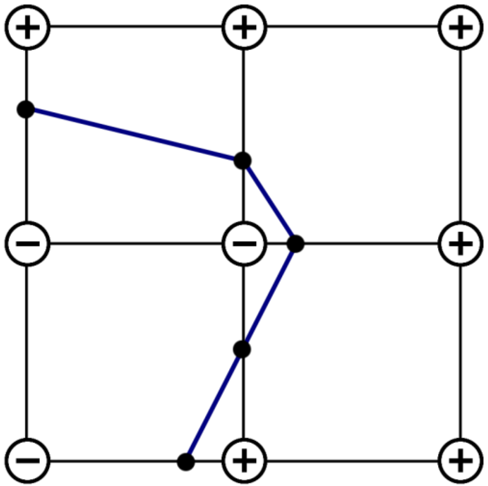
Marching Cubes



Look-up
table

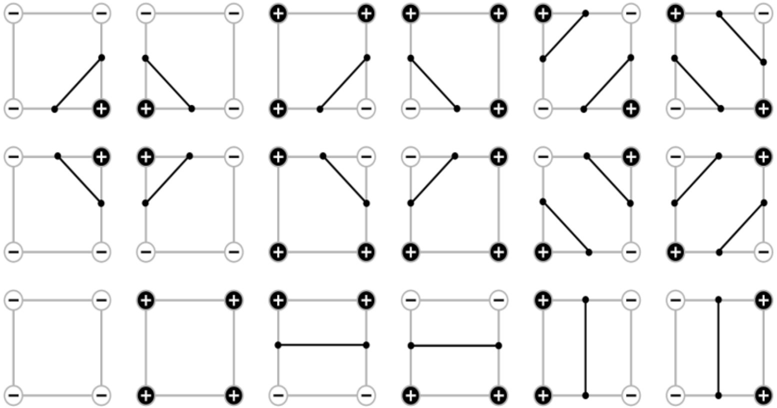


Marching Cubes

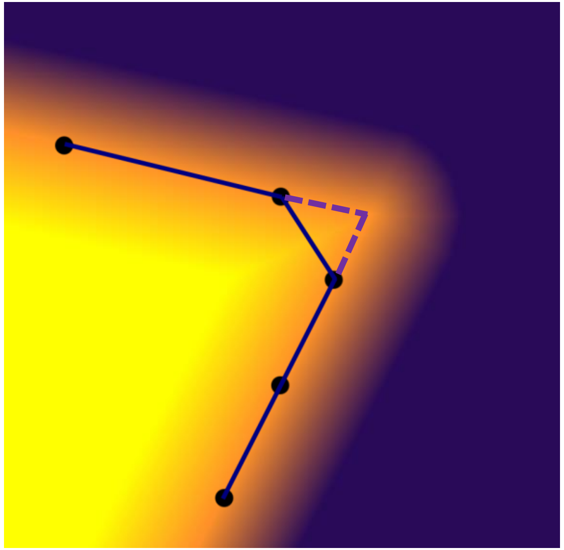


Look-up
table

(a) The face tessellations of Marching Cubes



Marching Cubes



The inspiration

Marching Cubes cannot reconstruct sharp features.

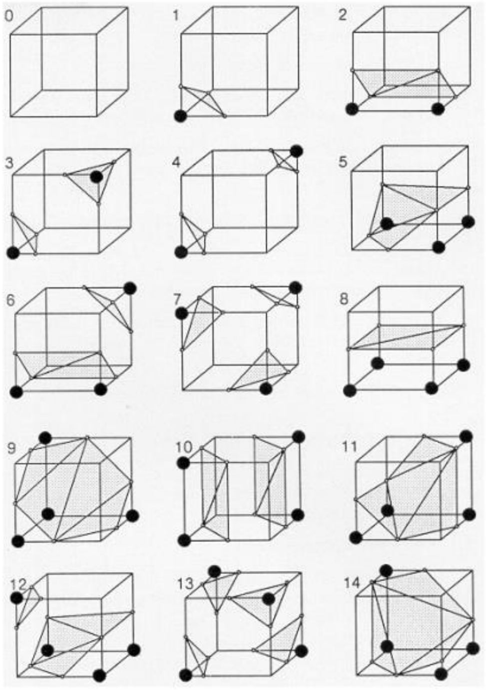
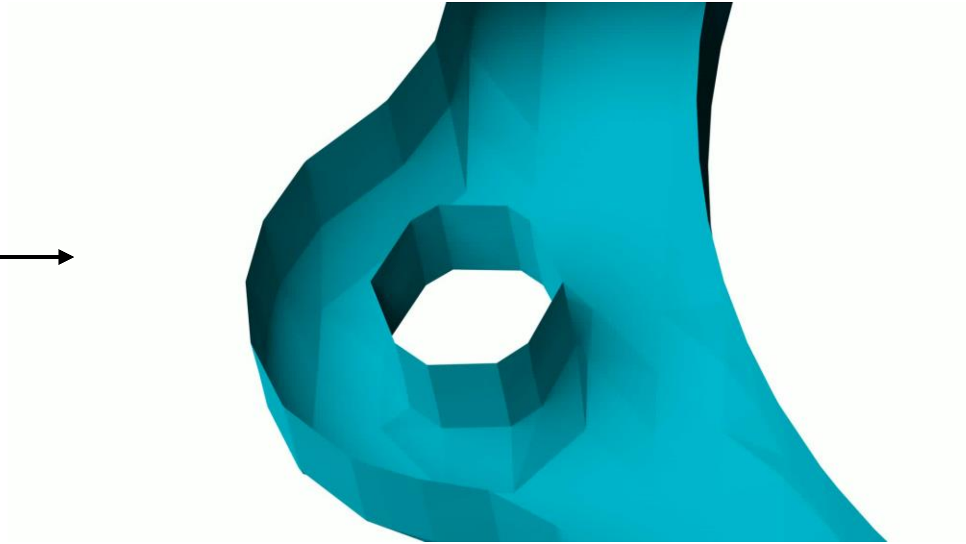
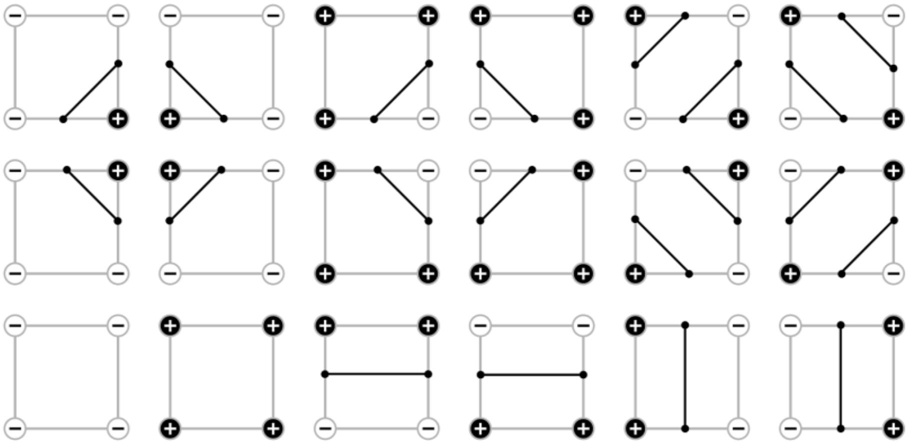


Figure 3. Triangulated Cubes.

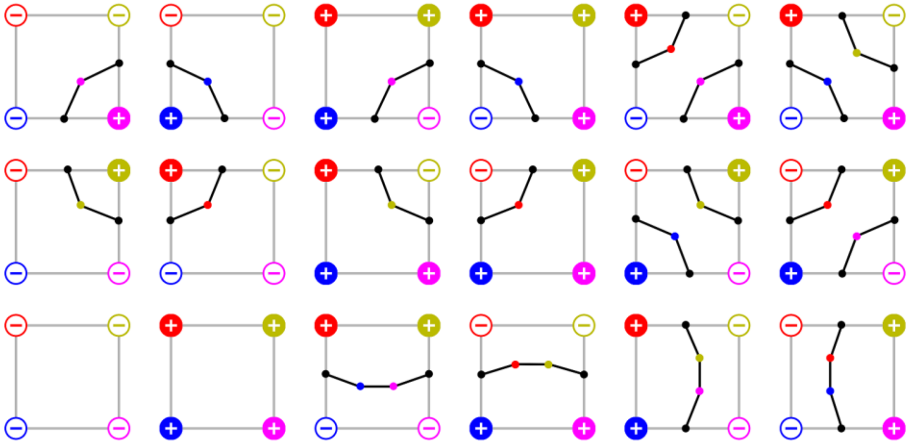


The inspiration

(a) The face tessellations of Marching Cubes



(b) Our face tessellations

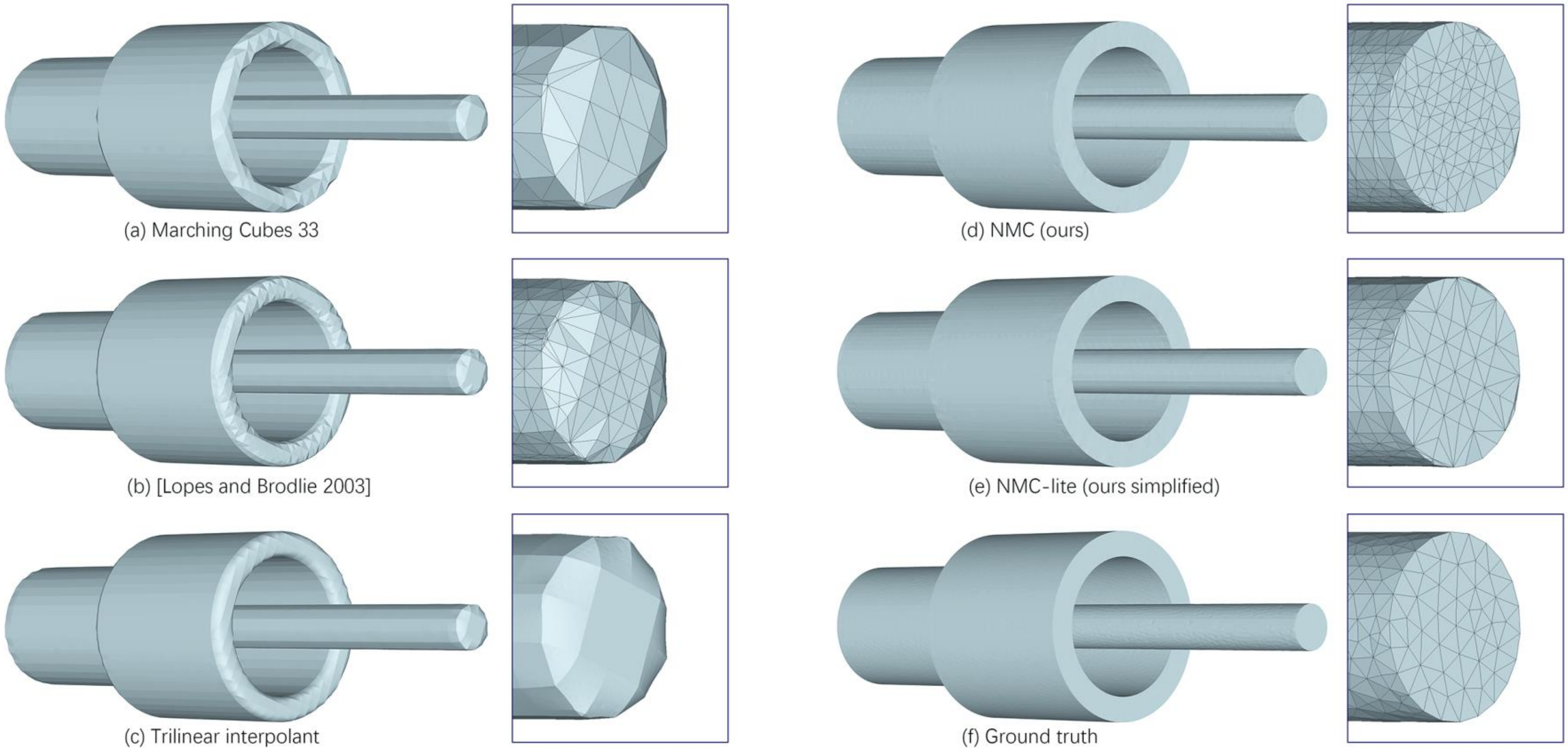


1. The MC templates cannot represent sharp features.

2. Additional vertices have to be added to represent sharp features. Now where to put those vertices?

Neural Marching Cubes

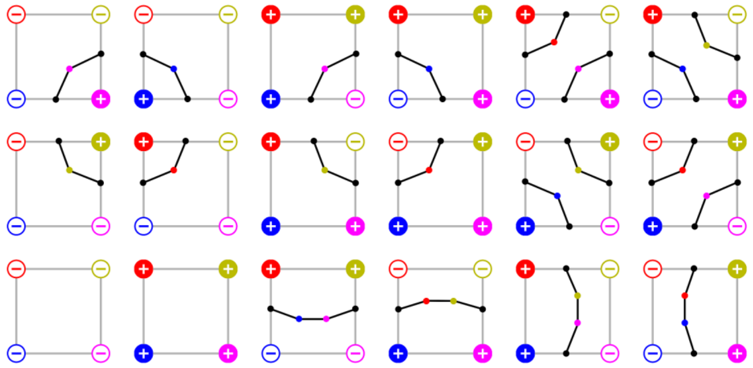
Zhiqin Chen, Hao Zhang



Summary of Neural Marching Cubes

1. Design templates.

(b) Our face tessellations

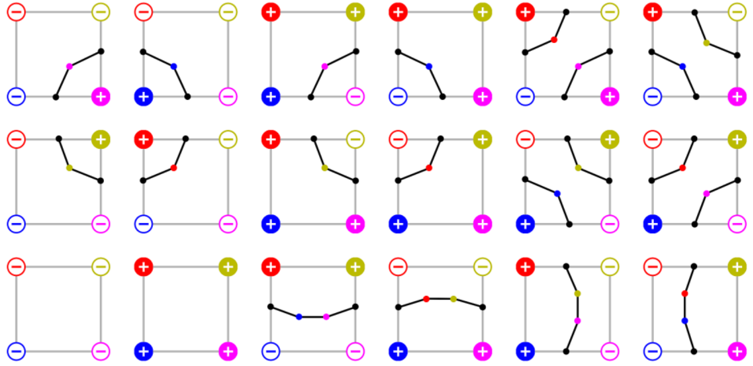


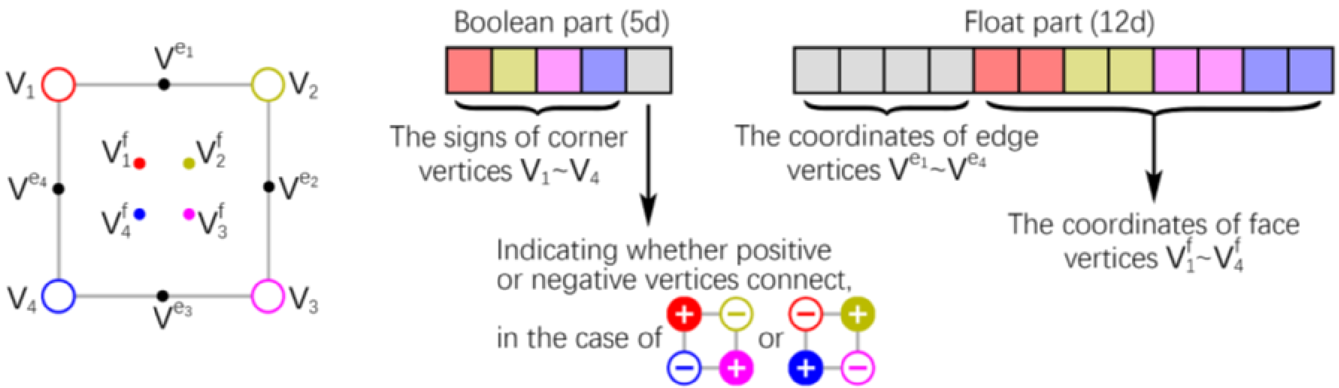
Summary of Neural Marching Cubes

1. Design templates.

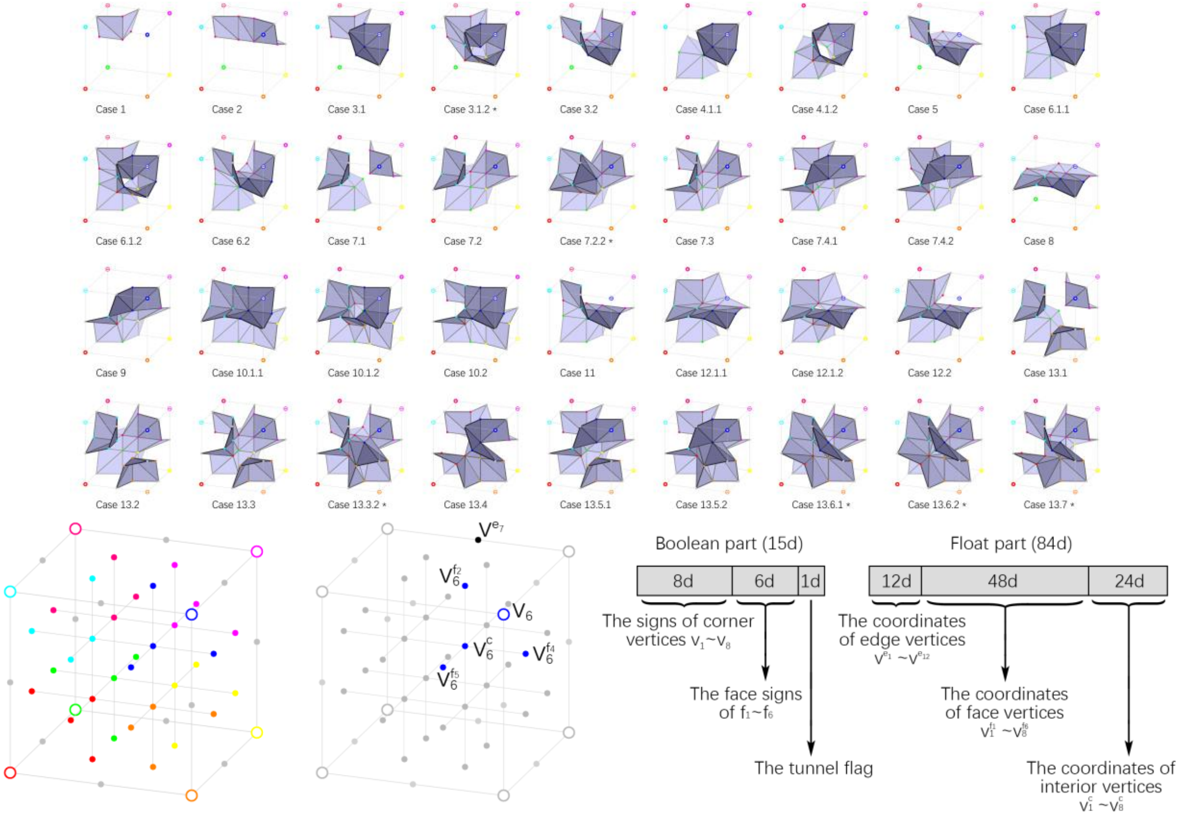
2. Parameterize templates.

(b) Our face tessellations





Summary of Neural Marching Cubes

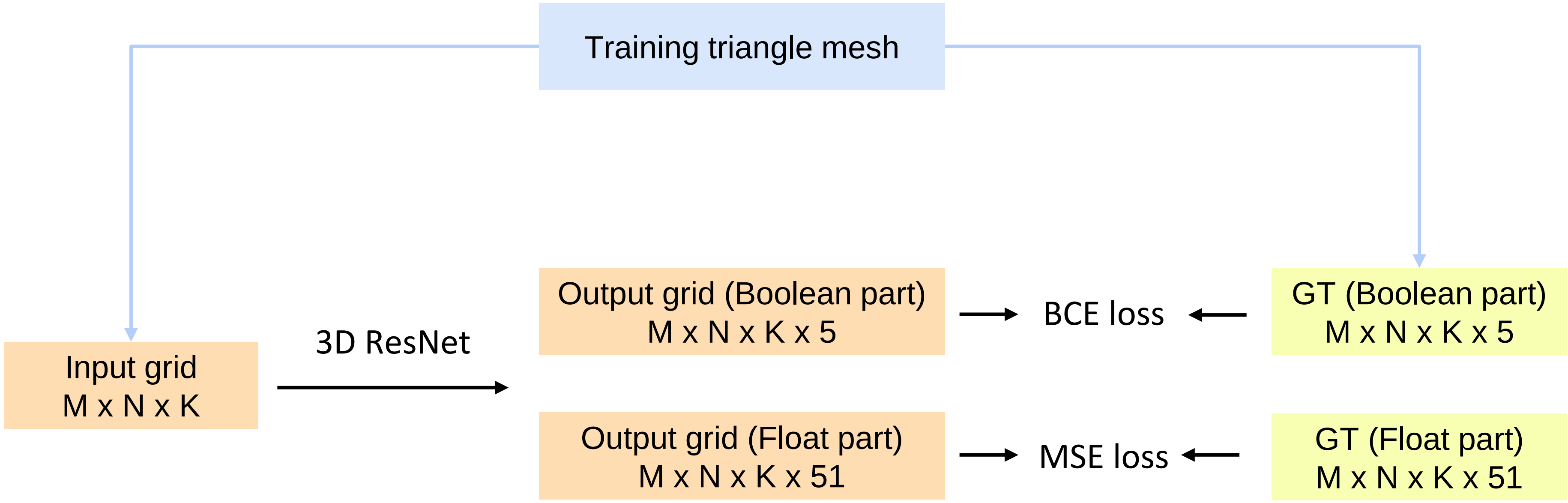


(a) All added vertices in a cube

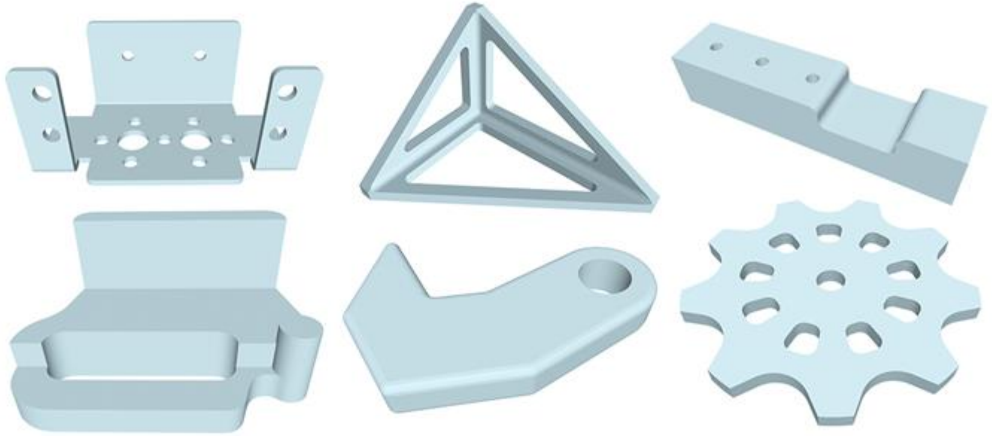
(b) all vertices corresponding to v_6 and an edge vertex on e_7

(c) Our representation to store each cube

Network and loss functions



Datasets



Training on:

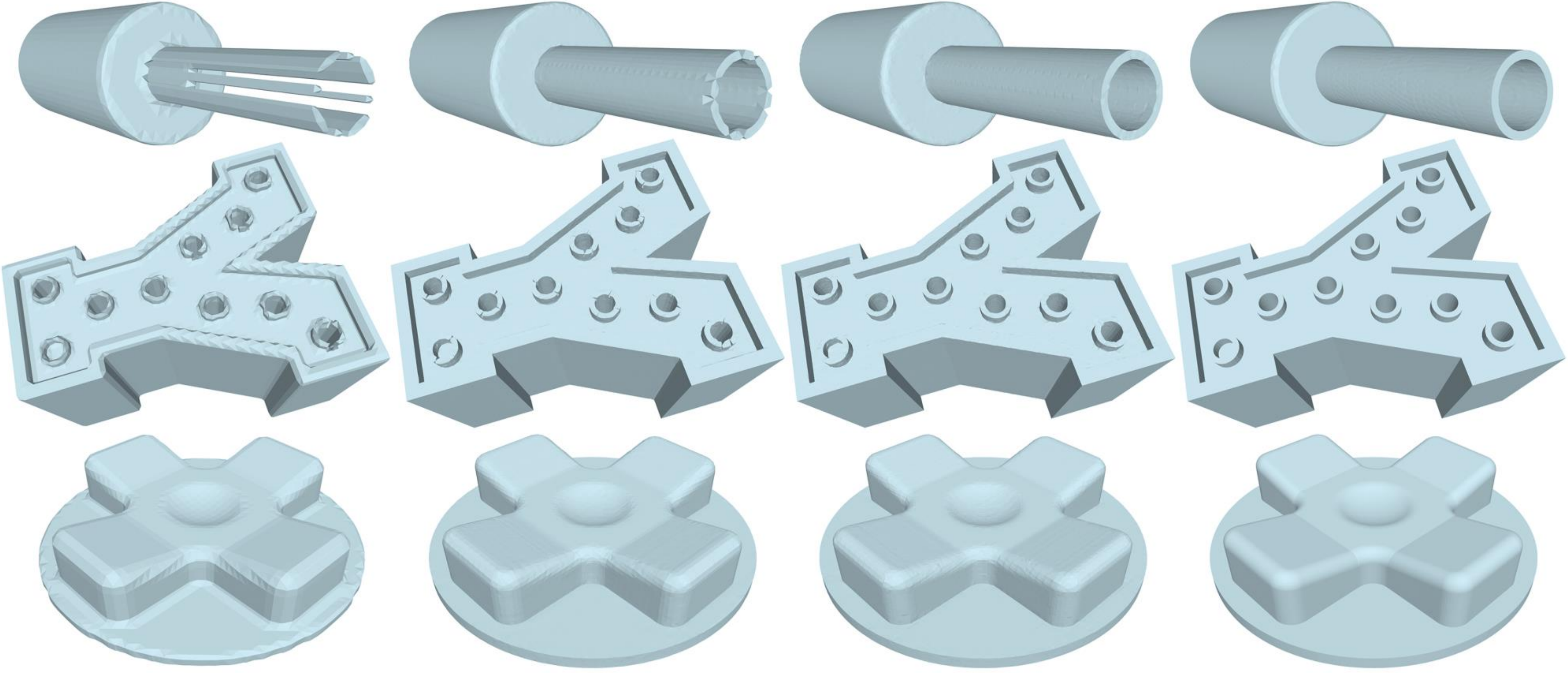
CAD shapes from the ABC dataset.

Testing on:

ABC, Thingi10k, FAUST.

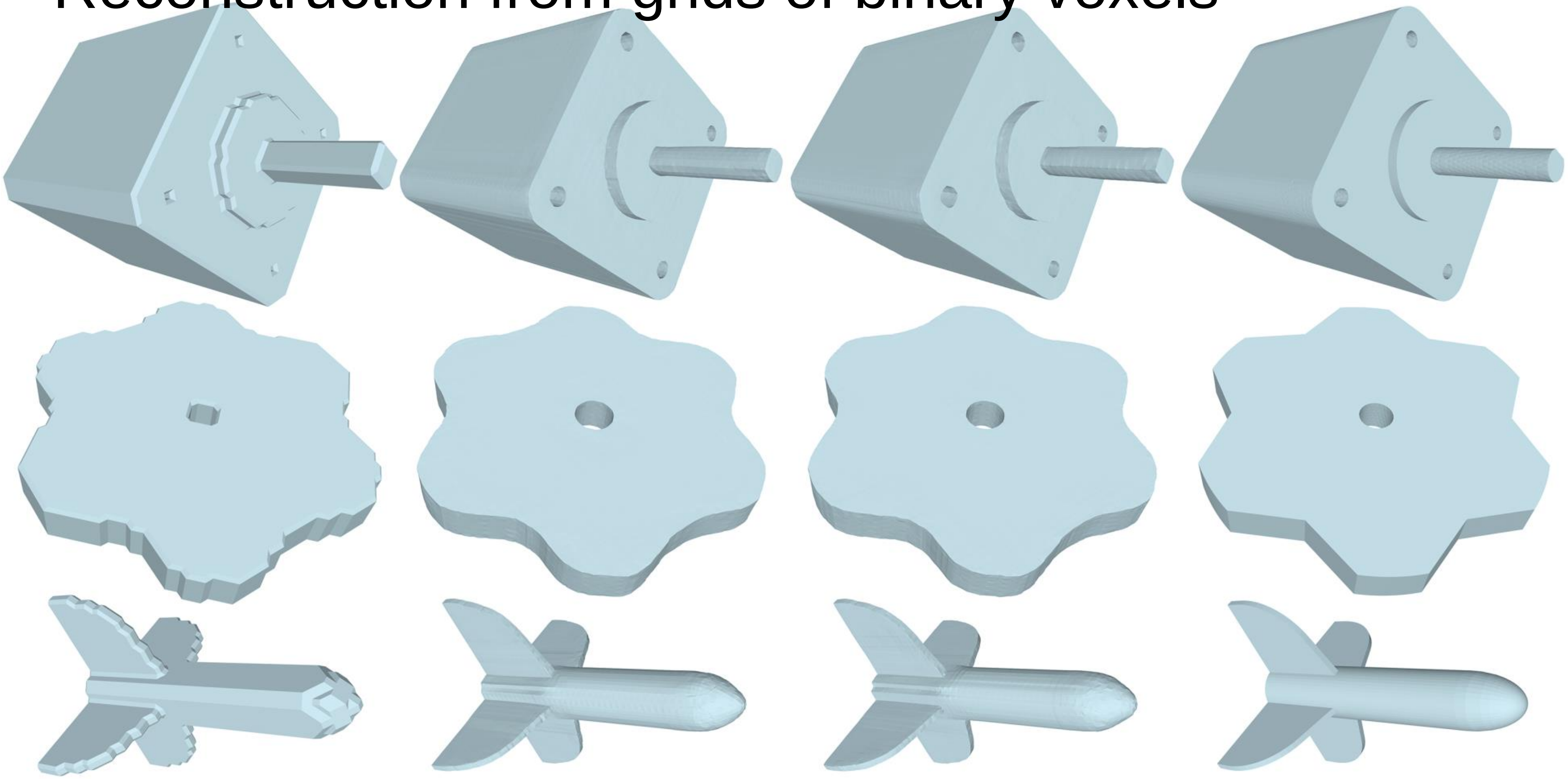
[3] ABC: a big CAD model dataset for geometric deep learning. Sebastian Koch et al. CVPR, 2019.
[4] Thingi10k: a dataset of 10,000 3d-printing models. Qingnan Zhou and Alec Jacobson. ArXiv, 2016.
[5] FAUST: Dataset and evaluation for 3D mesh registration. Federica Bogo et al. CVPR 2014.
[6] Marching cubes 33: construction of topologically correct isosurfaces. Evgeni Chernyaev. Technical Report CN/95-17, CERN, 1995.
[7] Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing. Adriano Lopes and Ken Brodlie. TVCG 2003.

Reconstruction from grids of signed distances



(a) Marching Cubes 33 (c) NMC-lite (d) NMC (e) Ground truth

Reconstruction from grids of binary voxels



(a) Marching Cubes 33 (c) NMC-lite (d) NMC (e) Ground truth

Organic shapes - FAUST

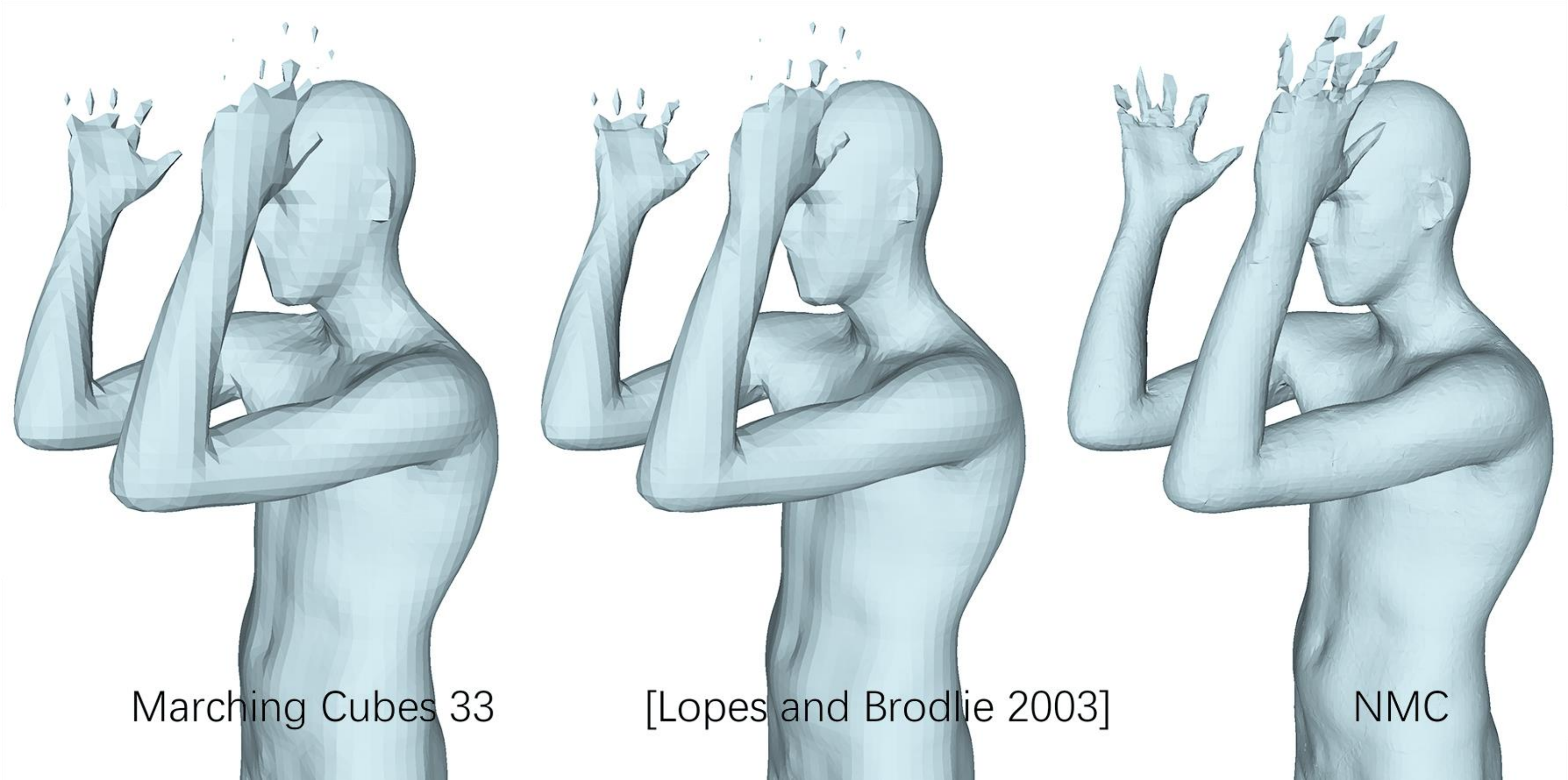


Table 1. Quantitative comparison results on ABC test set with SDF input.

64^3 resolution	CD($\times 10^5$)↓	F1↑	NC↑	ECD($\times 10^2$)↓	EF1↑	#V	#T
MC33	4.850	0.788	0.950	5.736	0.105	5,472.51	10,953.67
Lopes2003	4.803	0.798	0.958	6.841	0.100	21,979.95	43,892.05
Trilinear	4.733	0.803	0.960	7.275	0.098	-	-
NMC-lite	4.341	**0.877**	**0.975**	**0.382**	**0.759**	22,710.56	43,876.87
NMC	**4.323**	**0.877**	**0.975**	0.390	0.758	42,766.54	85,543.83
32^3 resolution	CD($\times 10^4$)↓	F1↑	NC↑	ECD($\times 10^2$)↓	EF1↑	#V	#T
MC33	5.239	0.570	0.900	5.504	0.048	1,297.38	2,595.47
Lopes2003	5.343	0.577	0.911	6.213	0.047	5,215.12	10,397.68
Trilinear	5.161	0.585	0.915	7.217	0.045	-	-
NMC-lite	3.922	0.823	**0.950**	**0.532**	0.631	5,464.48	10,389.43
NMC	**3.919**	**0.824**	0.949	0.598	**0.634**	9,728.20	19,460.09

Table 2. Quantitative comparisons on ABC test set with binary voxel input.

64^3 resolution	CD($\times 10^5$)↓	F1↑	NC↑	ECD($\times 10^2$)↓	EF1↑	#V	#T
MC33	26.860	0.085	0.921	11.196	0.018	5,826.08	11,655.52
Lopes2003	26.829	0.084	0.921	14.601	0.017	23,302.73	46,608.90
Trilinear	26.826	0.084	0.921	14.866	0.017	-	-
NMC-lite	**9.302**	**0.443**	0.930	0.559	**0.365**	22,185.94	42,915.64
NMC	9.341	0.438	**0.931**	**0.528**	0.356	42,043.03	84,087.85
32^3 resolution	CD($\times 10^4$)↓	F1↑	NC↑	ECD($\times 10^2$)↓	EF1↑	#V	#T
MC33	9.636	0.036	0.882	11.764	0.018	1,532.70	3,065.30
Lopes2003	9.632	0.036	0.883	14.723	0.017	6,130.84	12,261.58
Trilinear	9.641	0.035	**0.884**	14.820	0.017	-	-
NMC-lite	**5.909**	**0.237**	0.871	**0.901**	**0.112**	5,236.79	9,975.67
NMC	6.029	0.232	0.871	0.910	0.109	9,469.84	18,933.65

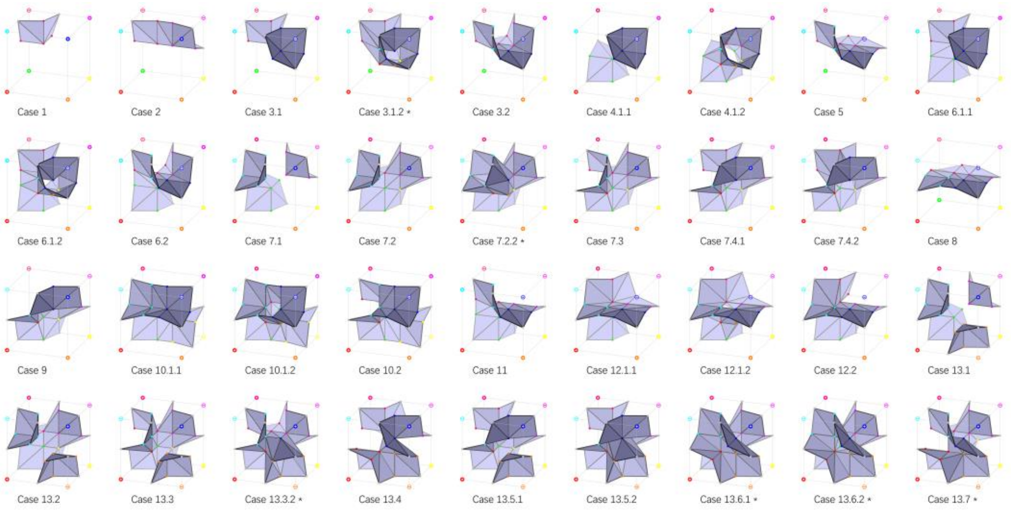
Table 3. Quantitative comparison results on Thingi10K with SDF inputs.

64^3 resolution	CD($\times 10^5$)↓	F1↑	NC↑	ECD($\times 10^2$)↓	EF1↑	#V	#T
MC33	3.195	0.795	0.945	3.763	0.099	5,517.51	11,044.35
Lopes2003	3.084	0.805	0.953	4.567	0.087	22,224.23	44,135.98
Trilinear	3.076	0.811	0.956	5.211	0.084	-	-
NMC-lite	**2.470**	**0.893**	**0.972**	0.330	**0.722**	22,991.80	44,109.17
NMC	2.477	**0.893**	**0.972**	**0.312**	**0.722**	40,951.73	81,910.41
32^3 resolution	CD($\times 10^4$)↓	F1↑	NC↑	ECD($\times 10^2$)↓	EF1↑	#V	#T
MC33	10.519	0.540	0.882	4.046	0.040	1,284.98	2,569.73
Lopes2003	10.473	0.547	0.893	4.596	0.038	5,163.28	10,281.15
Trilinear	10.431	0.555	0.897	5.180	0.037	-	-
NMC-lite	**8.425**	0.807	**0.935**	0.600	**0.542**	5,423.92	10,263.13
NMC	8.454	**0.808**	0.933	**0.596**	0.539	9,161.94	18,327.88

Table 4. Quantitative comparisons on Thingi10K with binary voxel inputs.

64^3 resolution	CD($\times 10^5$)↓	F1↑	NC↑	ECD($\times 10^2$)↓	EF1↑	#V	#T
MC33	25.538	0.069	0.907	7.411	0.017	5,939.62	11,881.67
Lopes2003	25.526	0.068	0.908	11.948	0.015	23,757.44	47,517.48
Trilinear	25.510	0.068	0.909	12.598	0.015	-	-
NMC-lite	**6.055**	**0.495**	**0.923**	0.606	**0.328**	22,540.88	43,272.05
NMC	6.108	0.493	**0.923**	**0.602**	0.314	40,430.06	80,861.75
32^3 resolution	CD($\times 10^4$)↓	F1↑	NC↑	ECD($\times 10^2$)↓	EF1↑	#V	#T
MC33	9.247	0.028	0.865	8.632	0.017	1,553.93	3,107.50
Lopes2003	**9.246**	0.028	**0.867**	12.344	0.015	6,215.99	12,431.69
Trilinear	9.256	0.028	**0.867**	12.709	0.015	-	-
NMC-lite	9.998	**0.258**	0.852	**0.946**	**0.096**	5,261.82	9,971.62
NMC	10.177	0.256	0.852	0.957	0.093	9,043.78	18,083.90

Issues of NMC

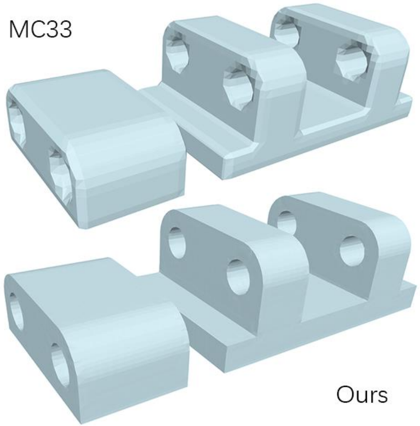


1. Complex

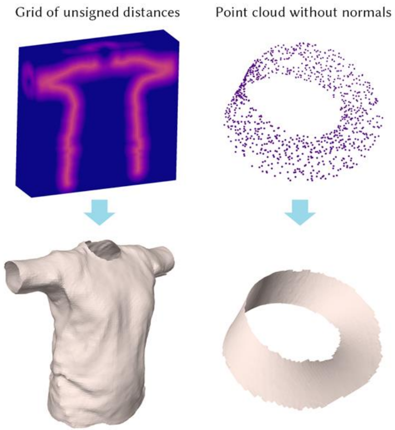
2. Slow

3. Producing a lot more vertices and triangles (4x or 8x) compared to MC

Overview



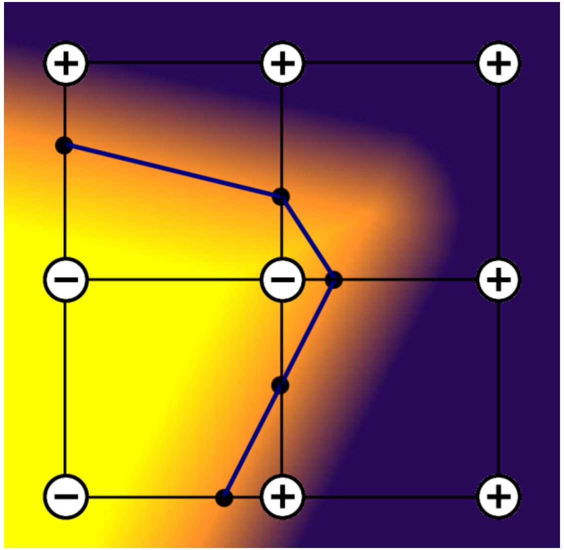
Neural Marching Cubes
(SIGGRAPH Asia 2021)



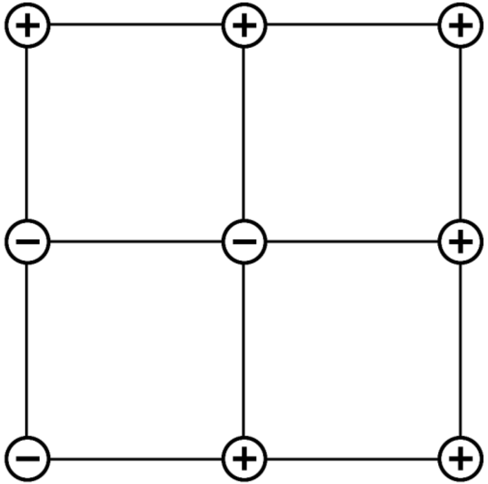
Neural Dual Contouring
(SIGGRAPH 2022)

MobileNeRF
(Arxiv 2022)

Marching Cubes

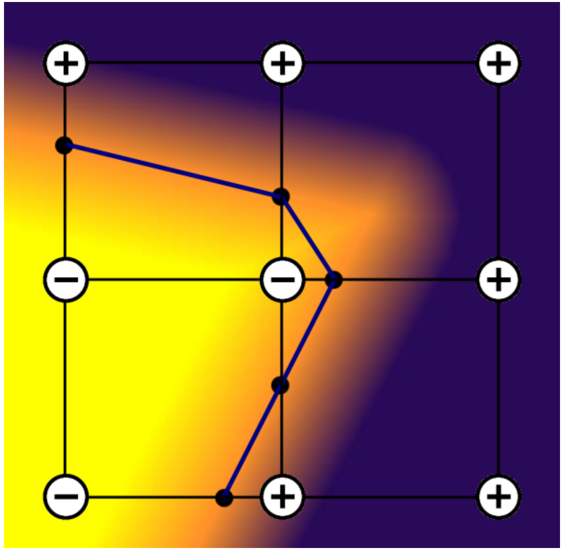


Dual Contouring

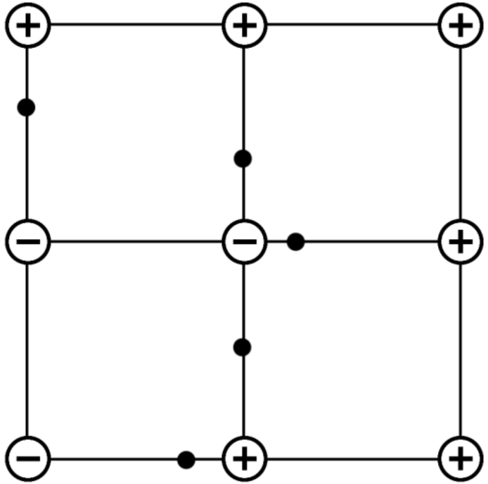


[8] Dual Contouring of Hermite Data. Ju et al. ACM Transactions on graphics, 2002.

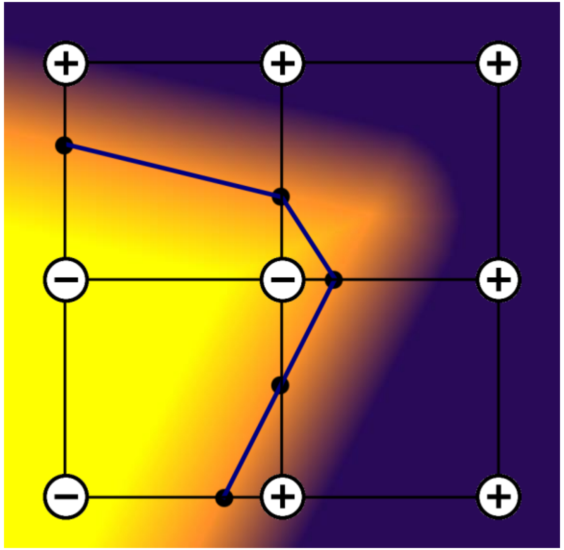
Marching Cubes



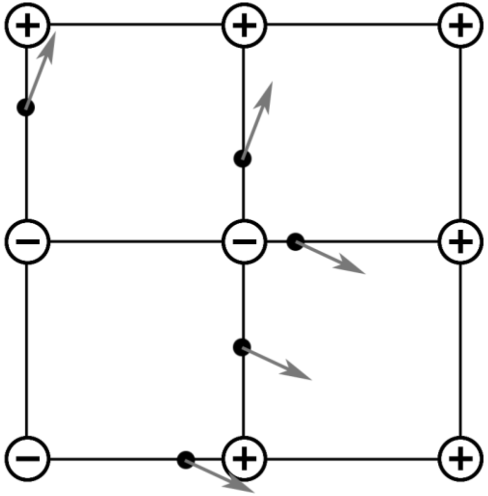
Dual Contouring



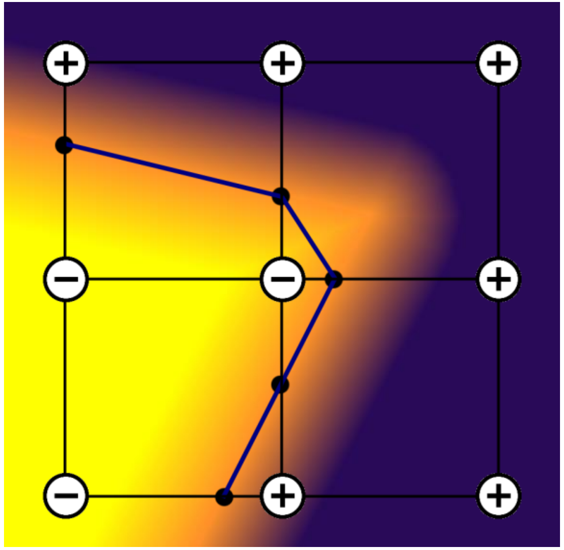
Marching Cubes



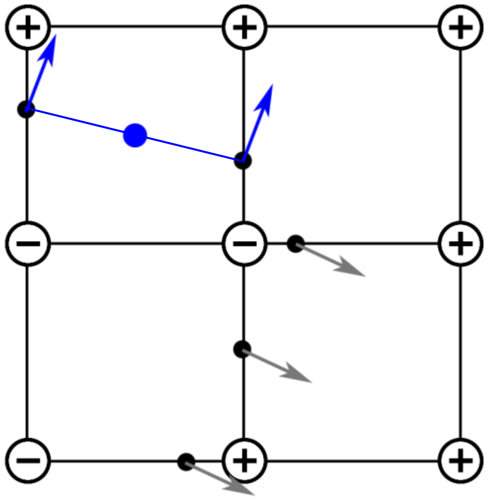
Dual Contouring



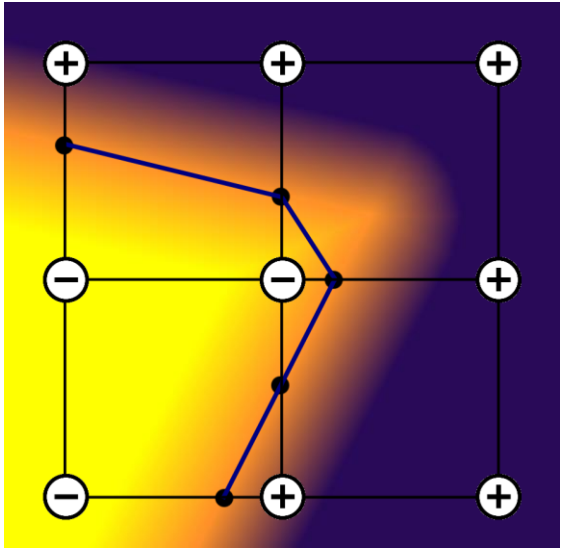
Marching Cubes



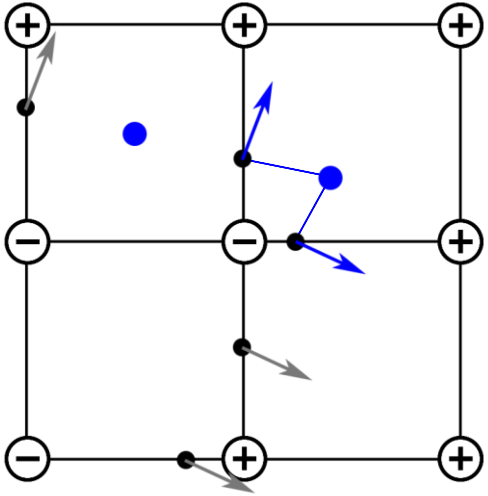
Dual Contouring



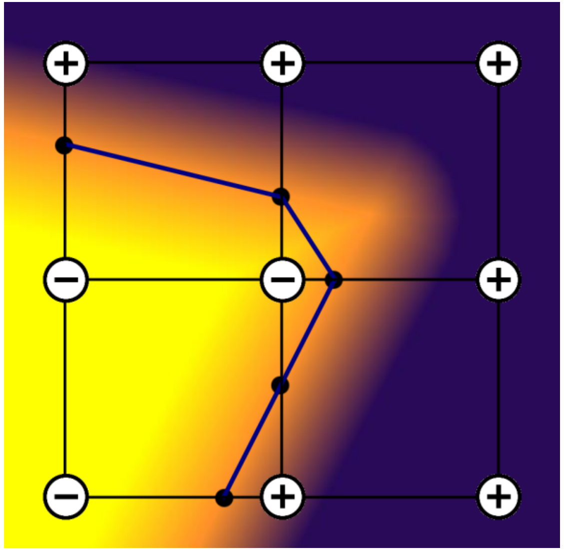
Marching Cubes



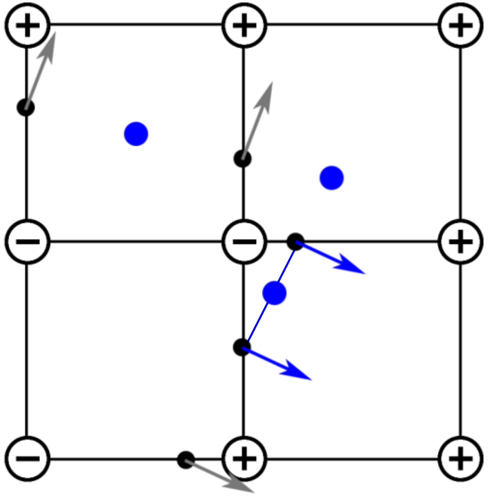
Dual Contouring



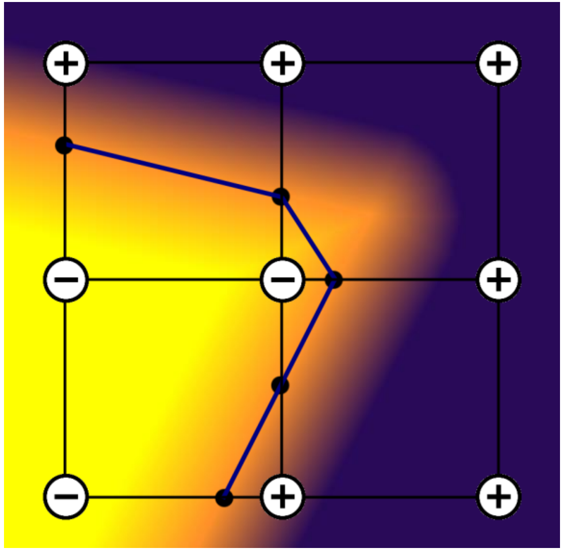
Marching Cubes



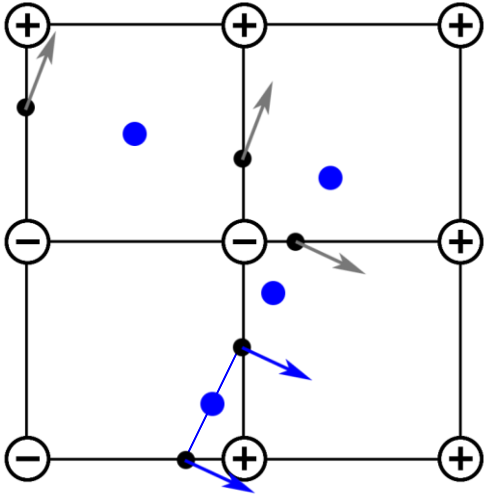
Dual Contouring



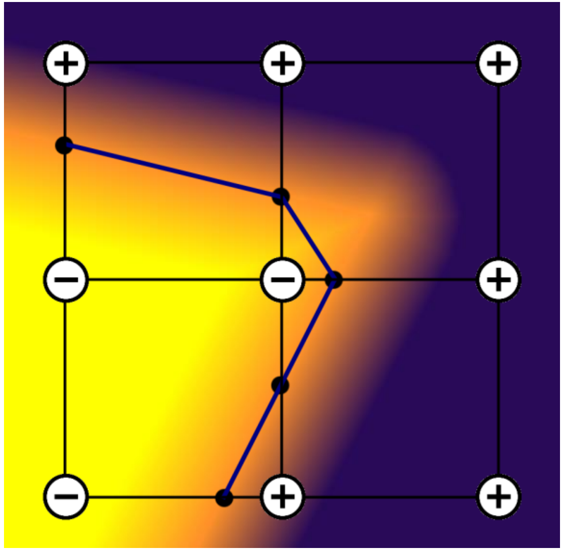
Marching Cubes



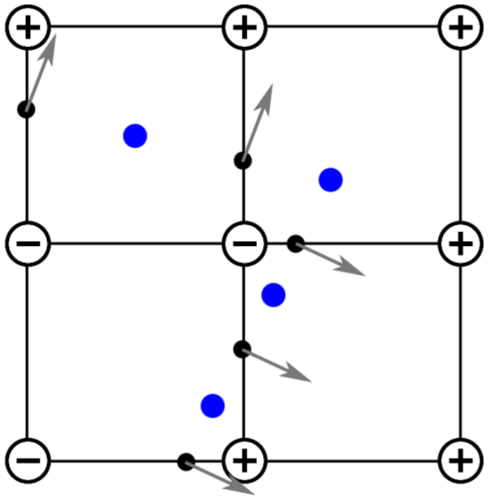
Dual Contouring



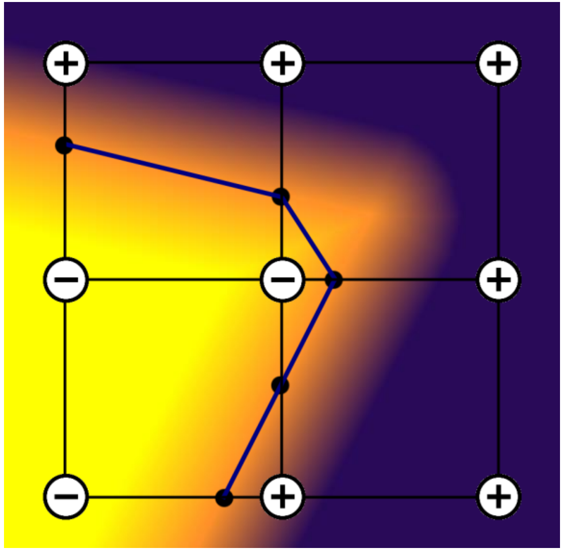
Marching Cubes



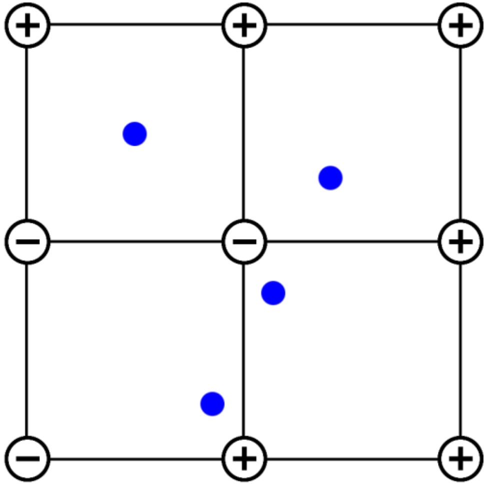
Dual Contouring



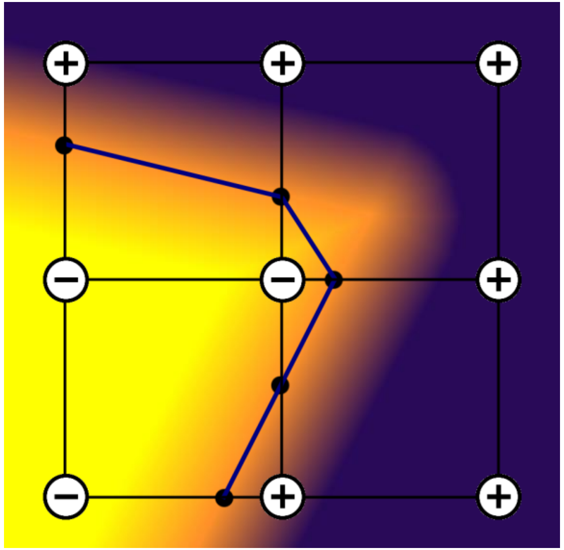
Marching Cubes



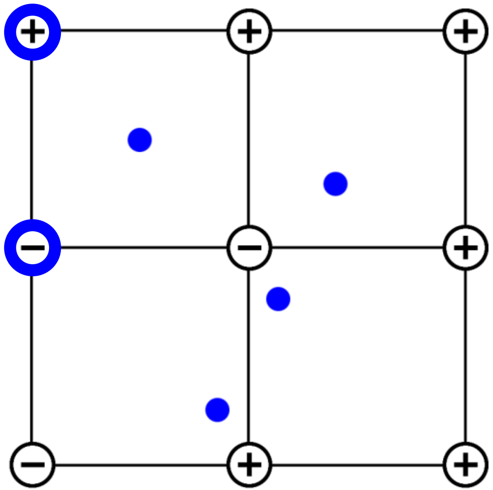
Dual Contouring



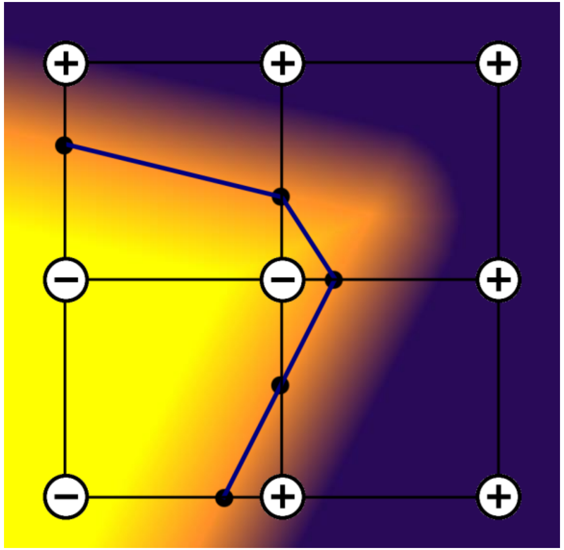
Marching Cubes



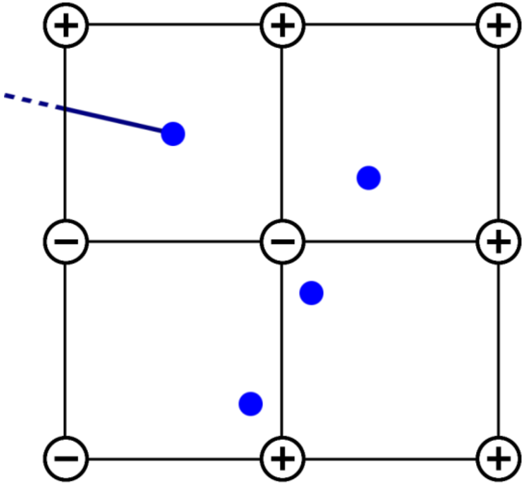
Dual Contouring



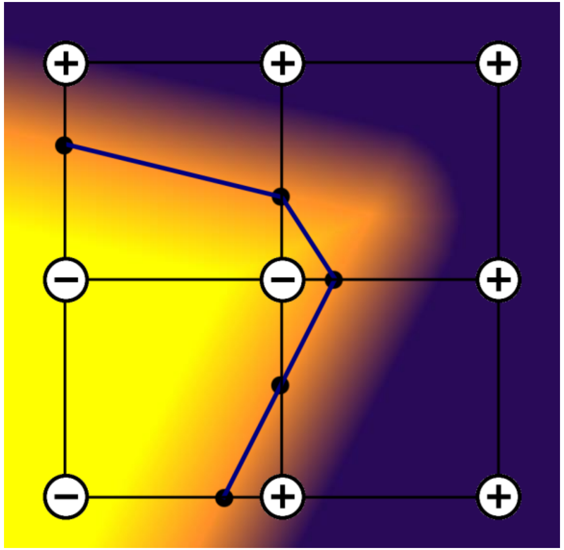
Marching Cubes



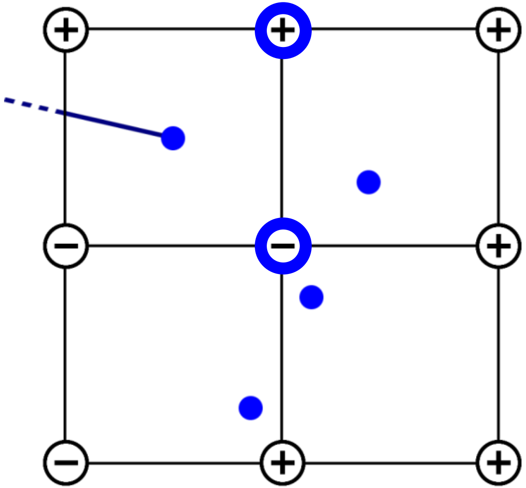
Dual Contouring



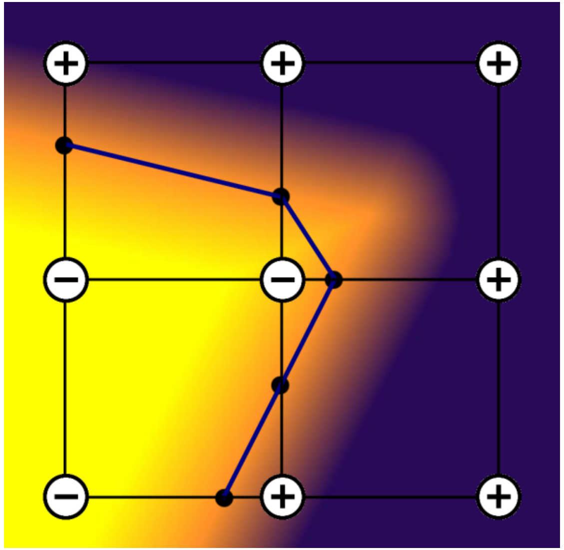
Marching Cubes



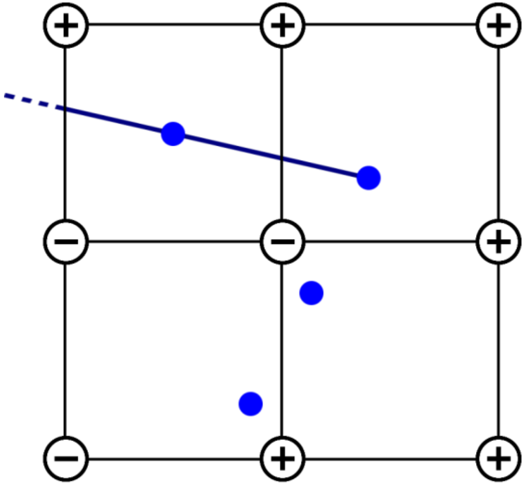
Dual Contouring



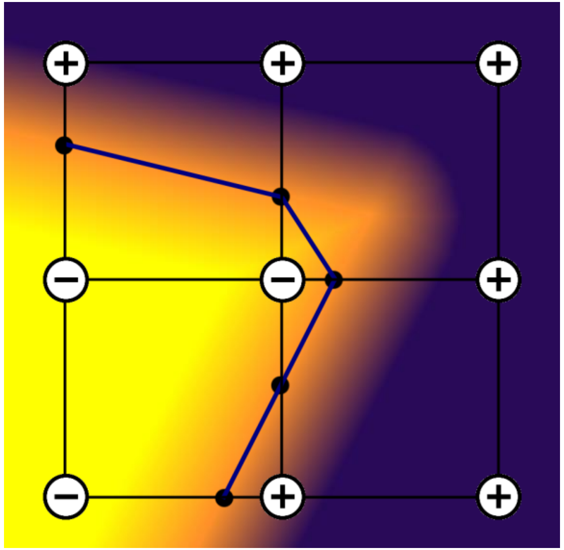
Marching Cubes



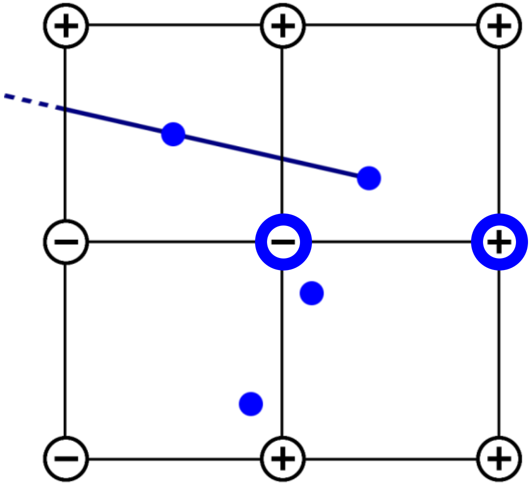
Dual Contouring



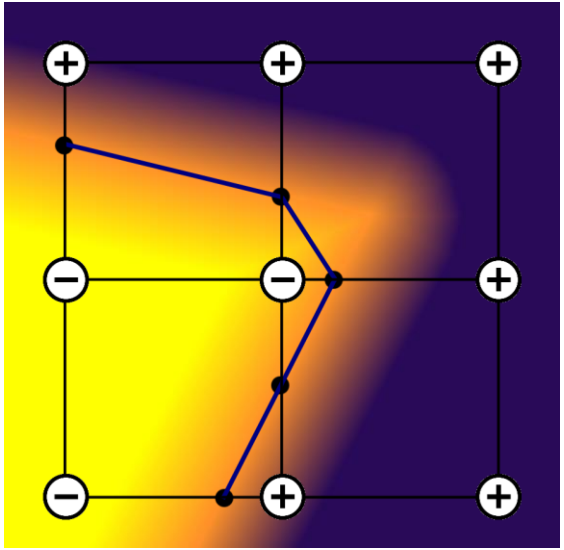
Marching Cubes



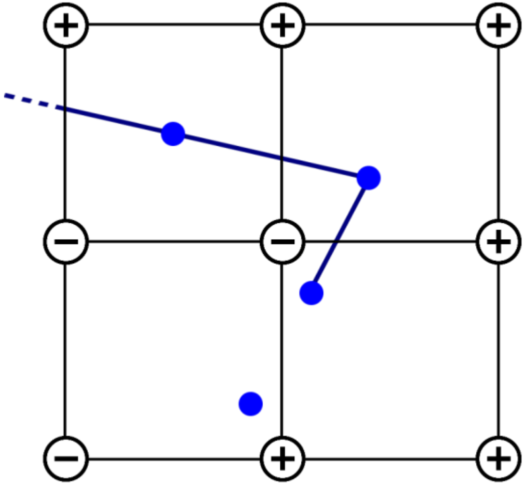
Dual Contouring



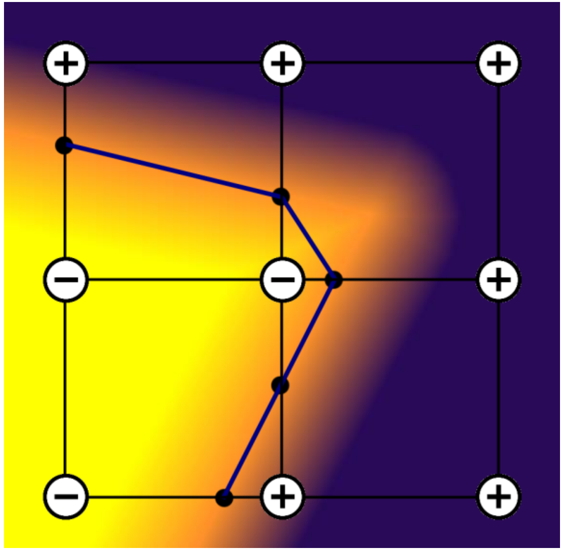
Marching Cubes



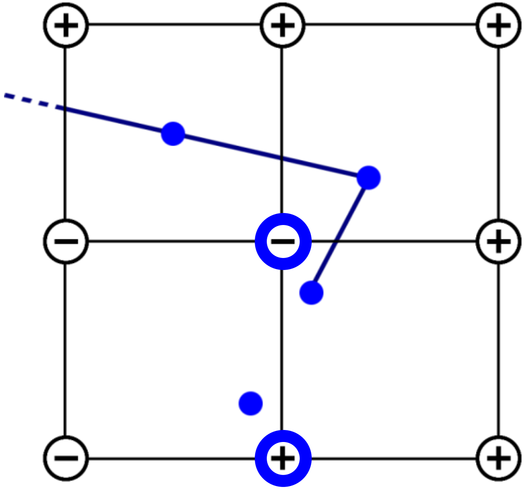
Dual Contouring



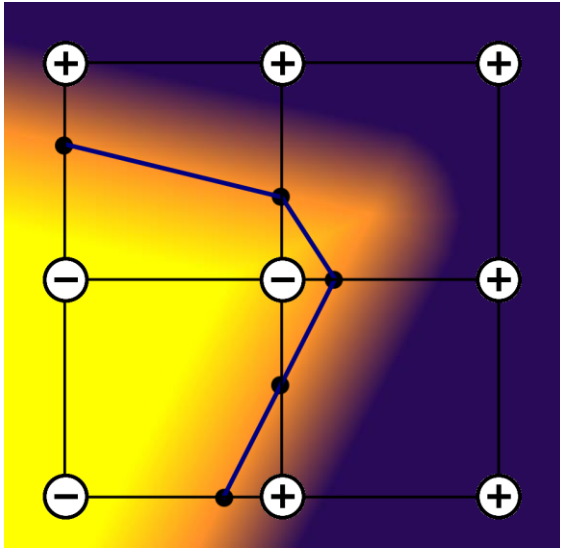
Marching Cubes



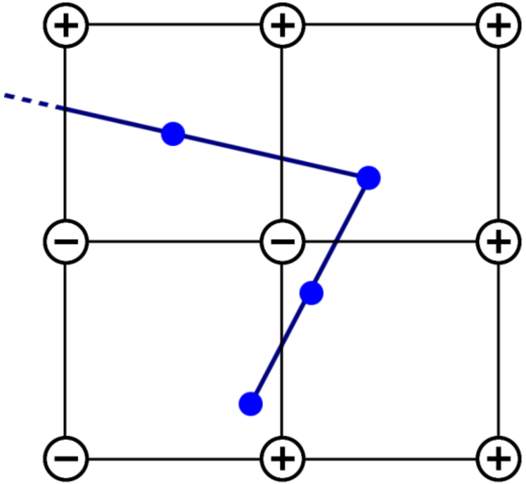
Dual Contouring



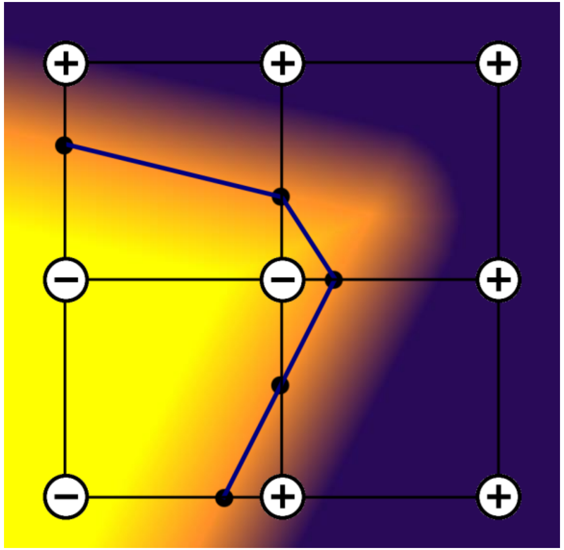
Marching Cubes



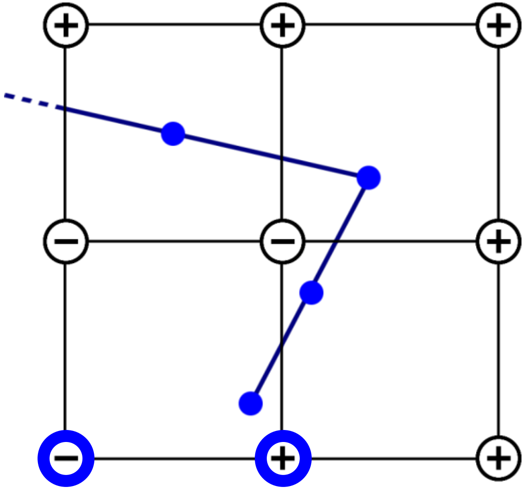
Dual Contouring



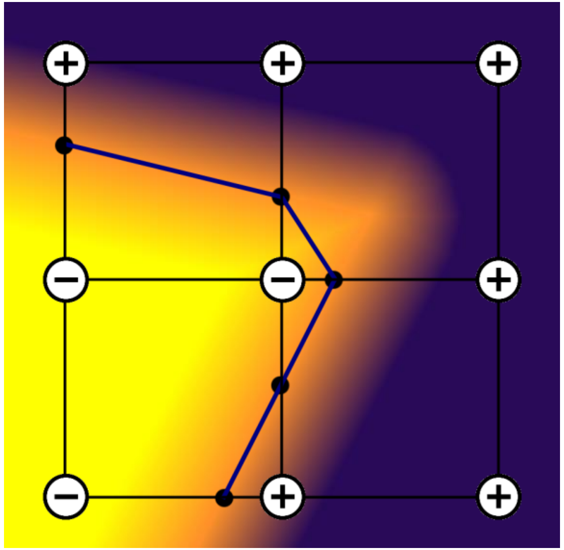
Marching Cubes



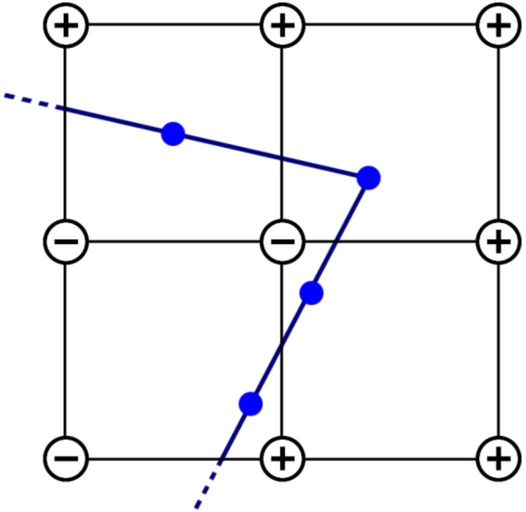
Dual Contouring



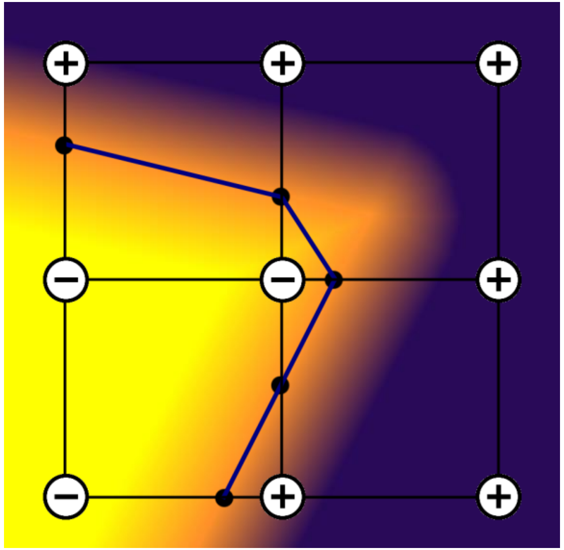
Marching Cubes



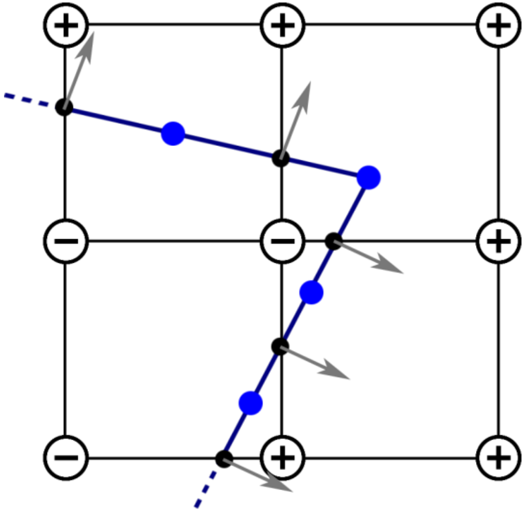
Dual Contouring



Marching Cubes



Dual Contouring

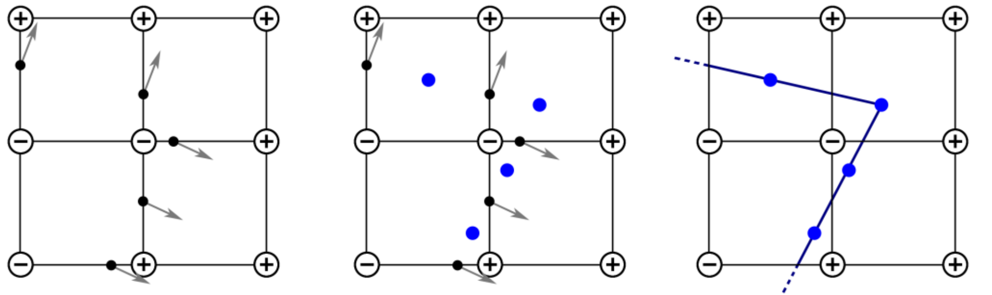


Issues of NMC

1. Complex

2. Slow

3. Producing a lot more vertices and triangles (4x or 8x) compared to MC



DC input: corner signs, intersection points on cell edges and their gradients (the arrows).

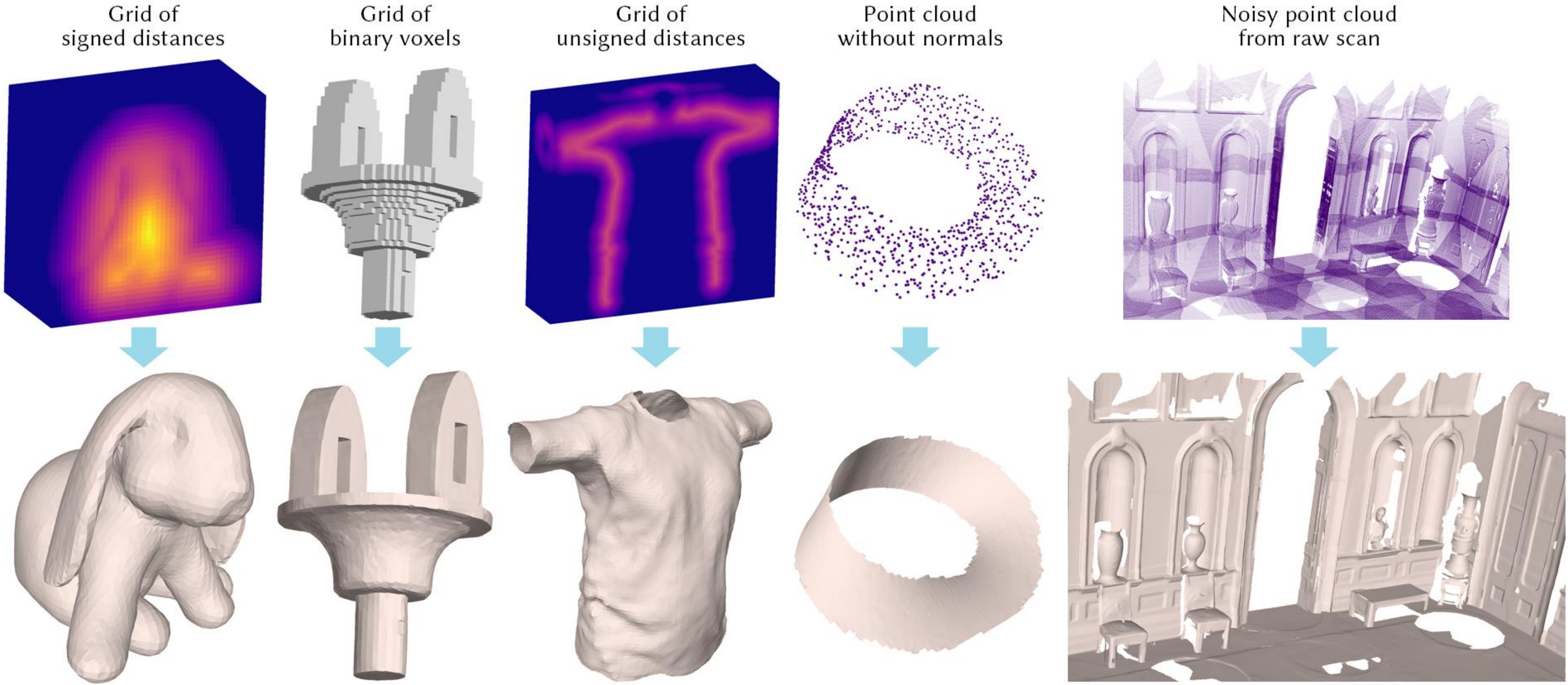
For each cell with a sign change, generate a vertex according to the QEFs.

For each edge with a sign change, generate a quad face connecting the vertices of the four adjacent cells.

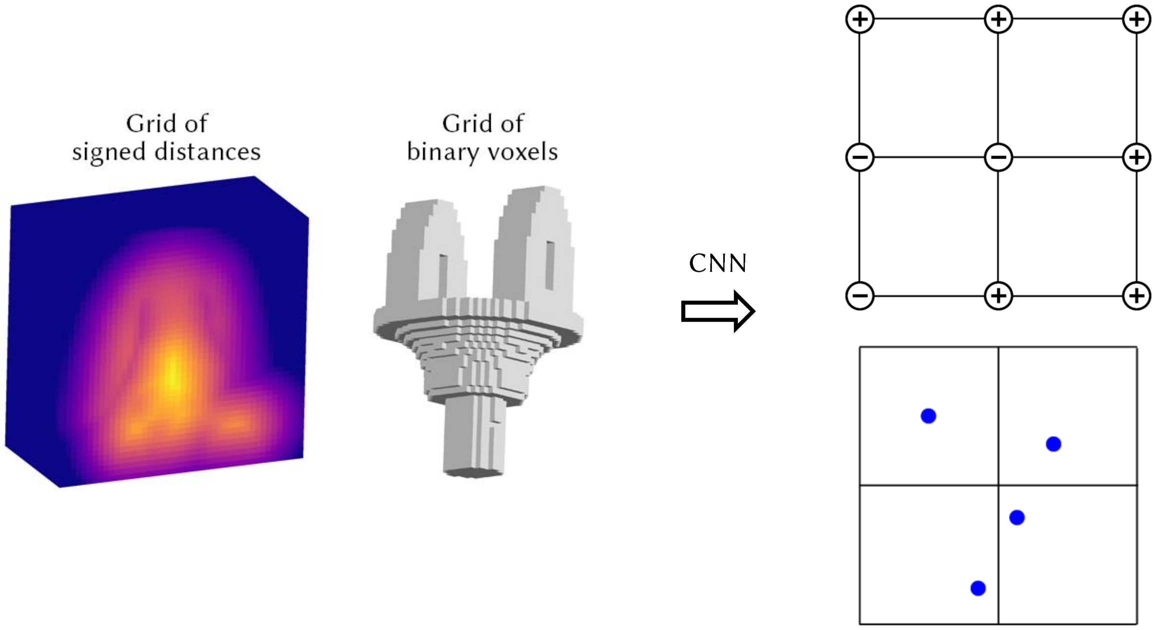
Dual Contouring

Neural Dual Contouring

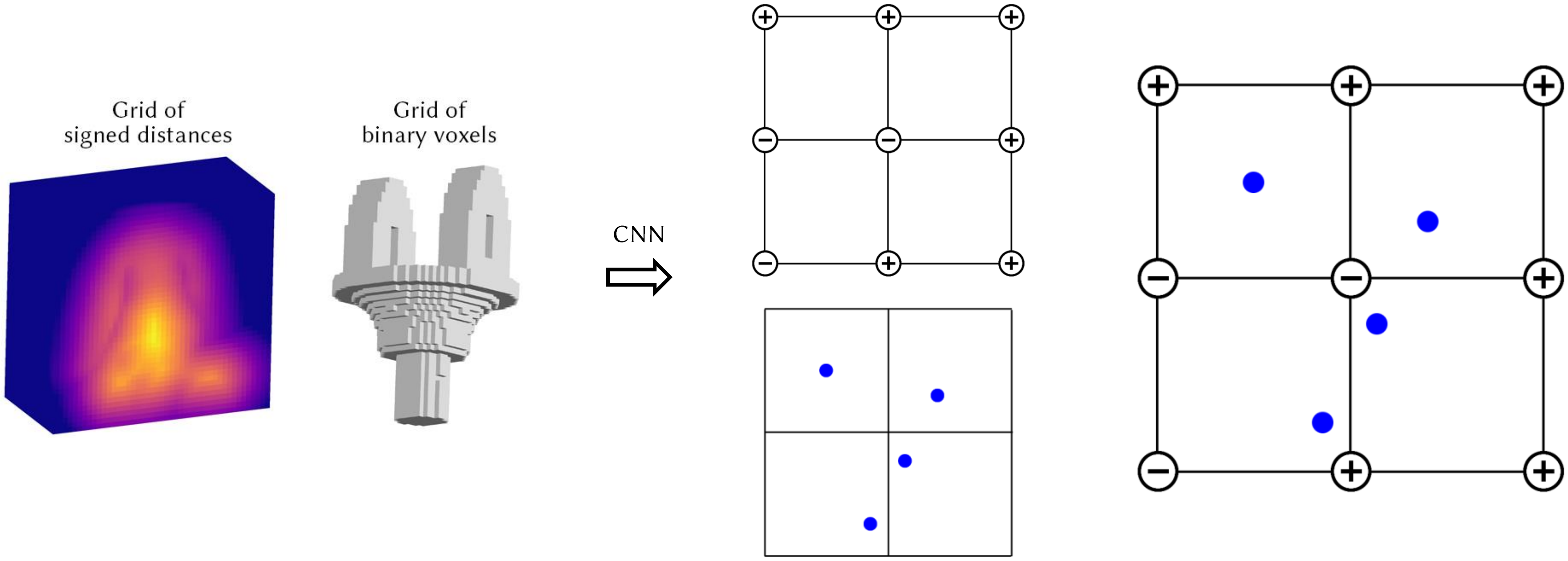
Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, Hao Zhang



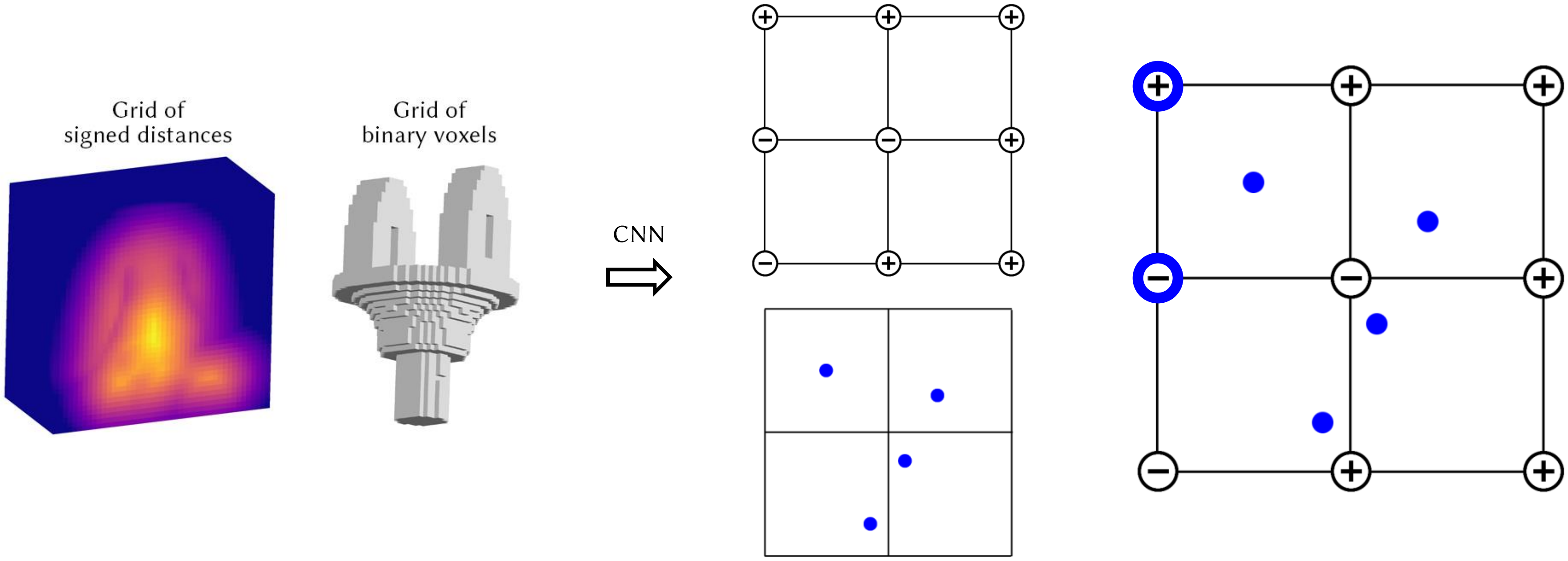
Neural Dual Contouring (NDC)



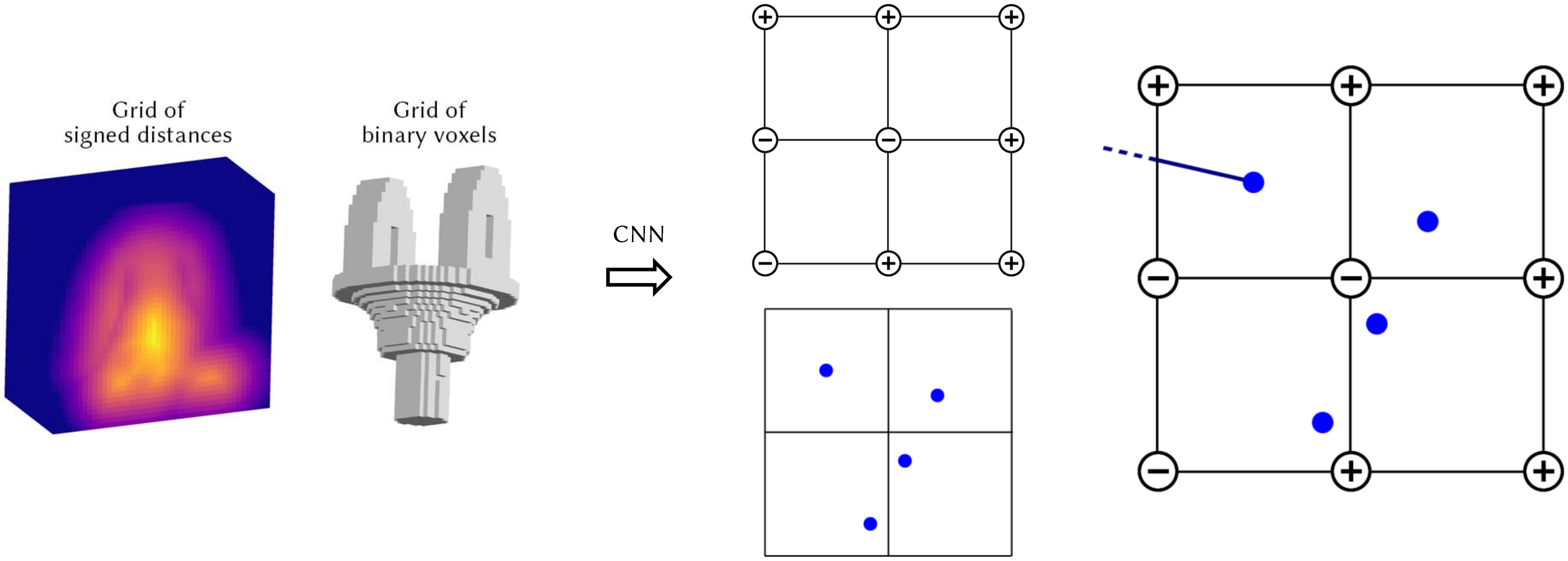
Neural Dual Contouring (NDC)



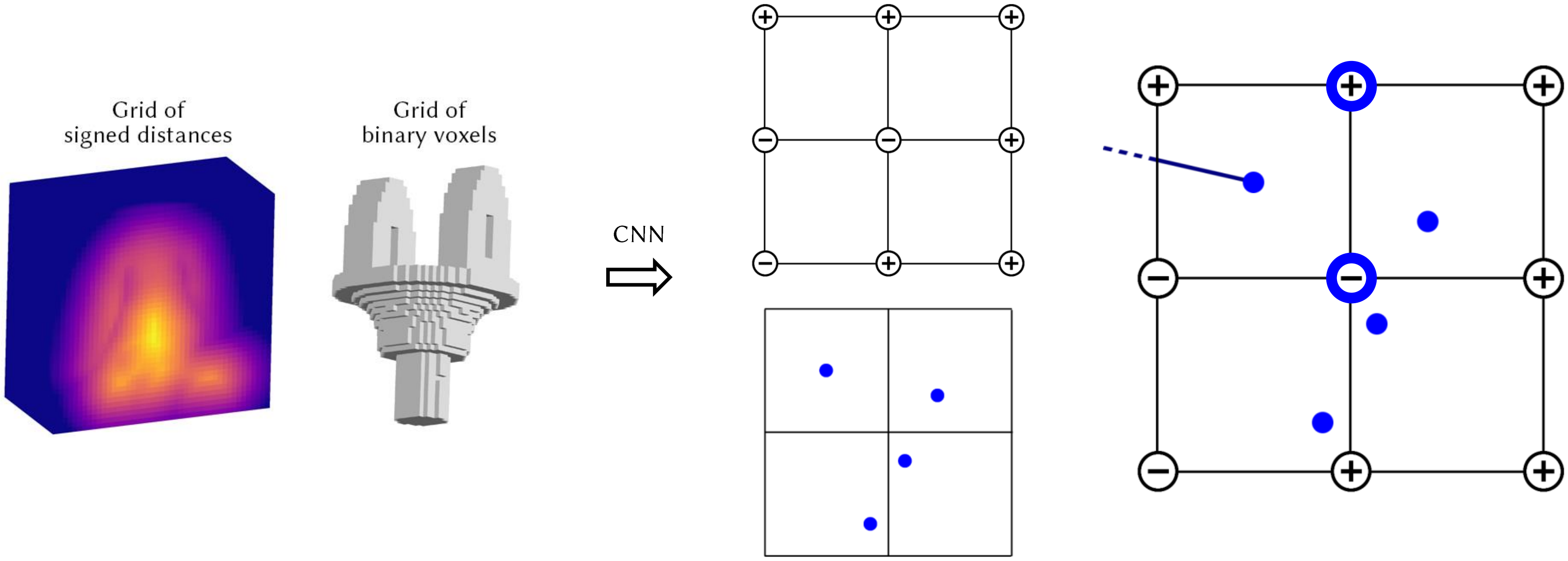
Neural Dual Contouring (NDC)



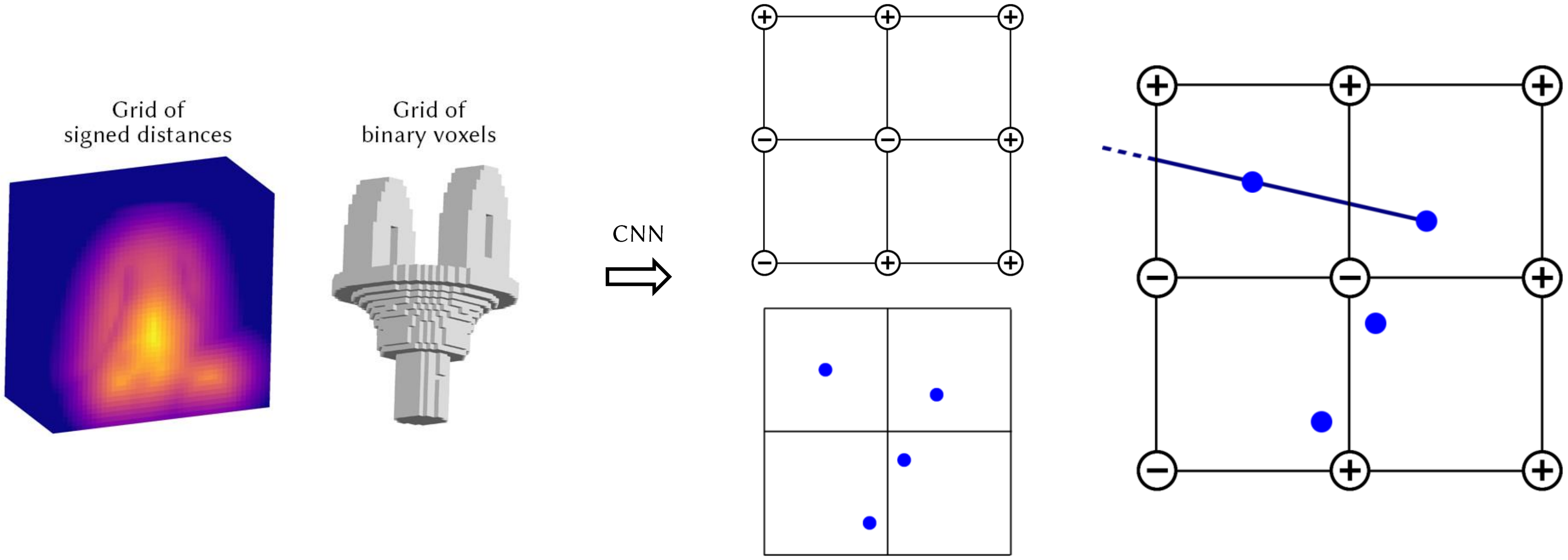
Neural Dual Contouring (NDC)



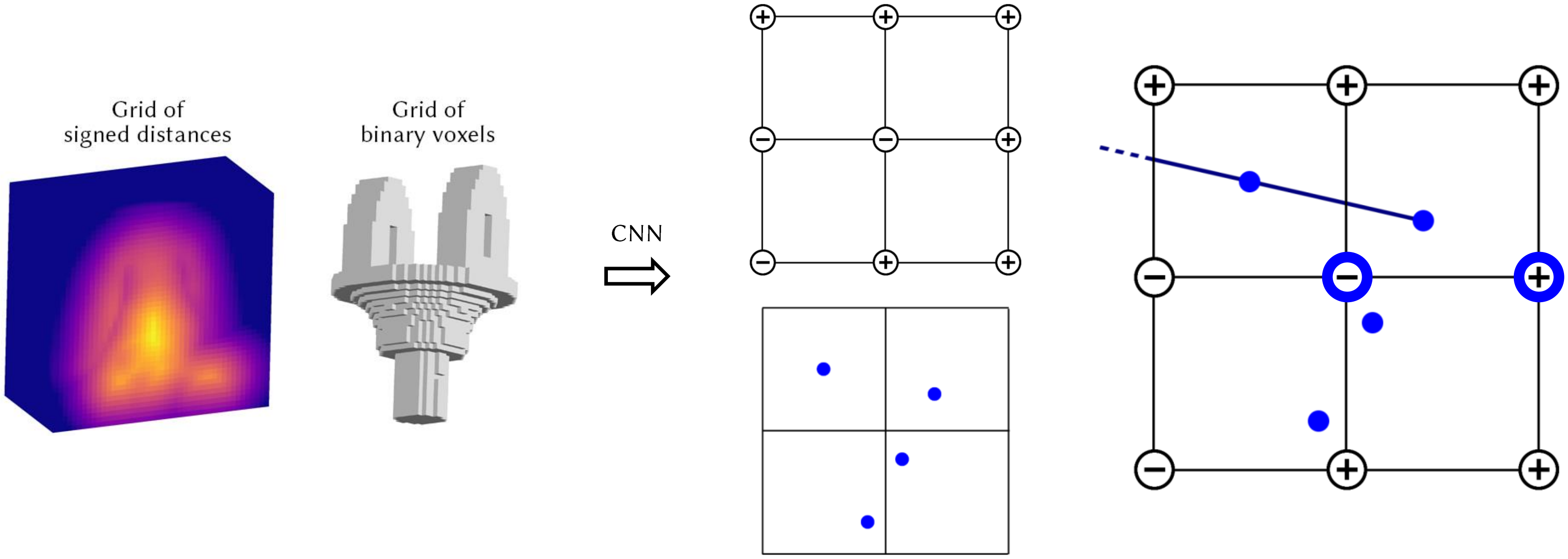
Neural Dual Contouring (NDC)



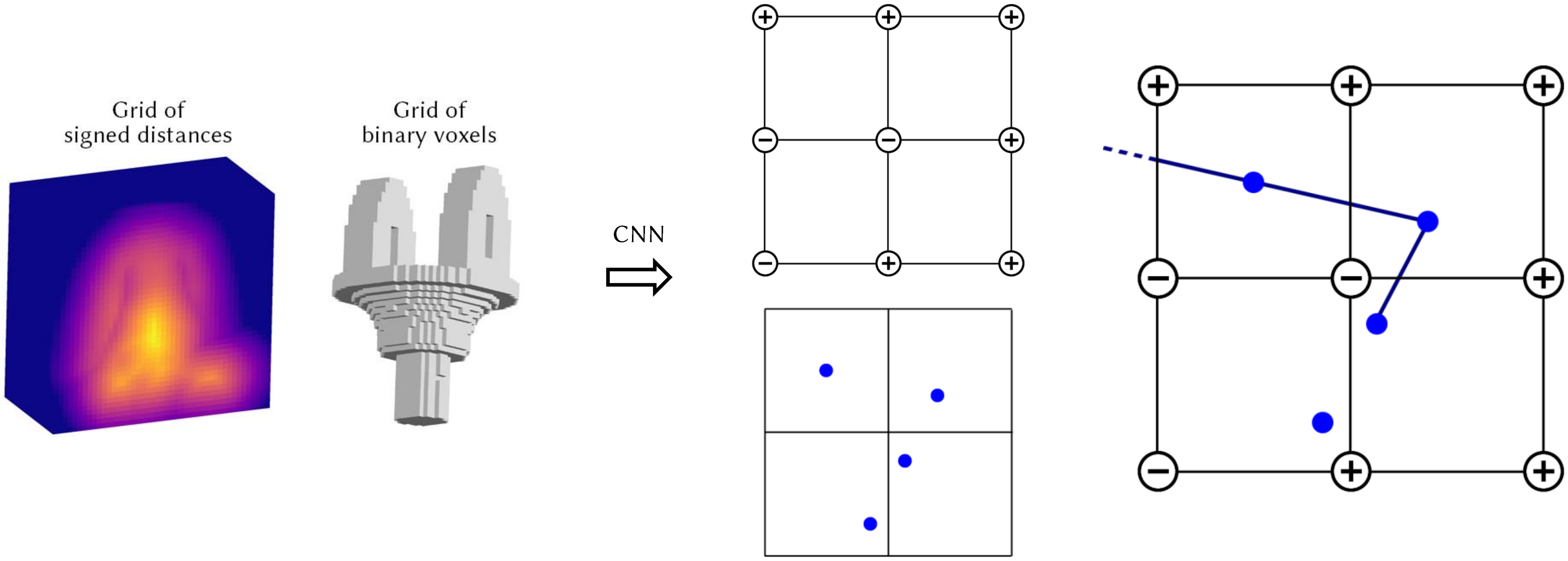
Neural Dual Contouring (NDC)



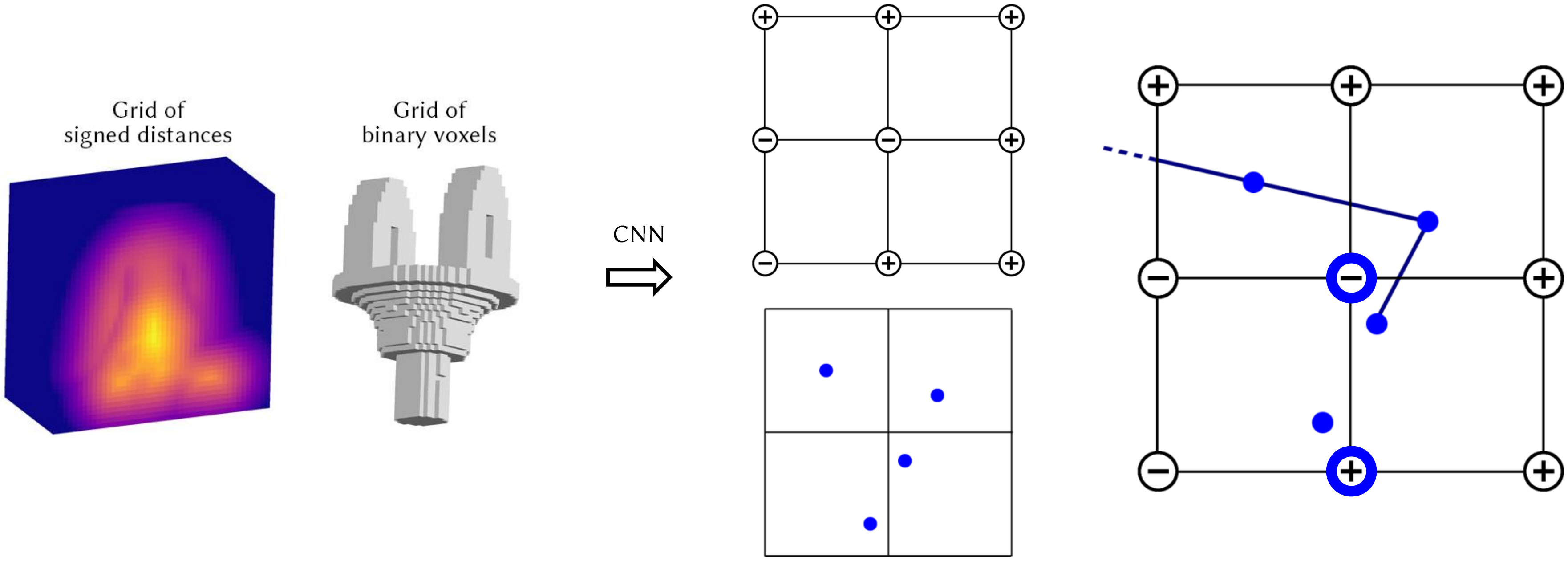
Neural Dual Contouring (NDC)



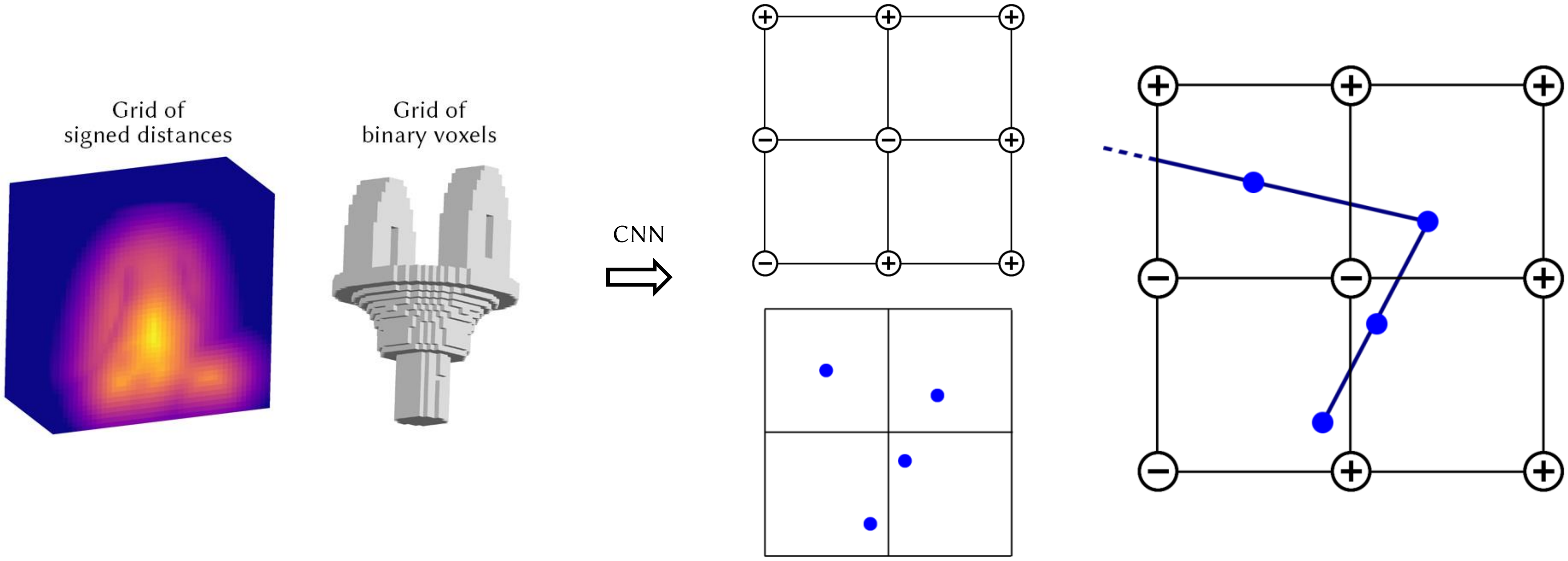
Neural Dual Contouring (NDC)



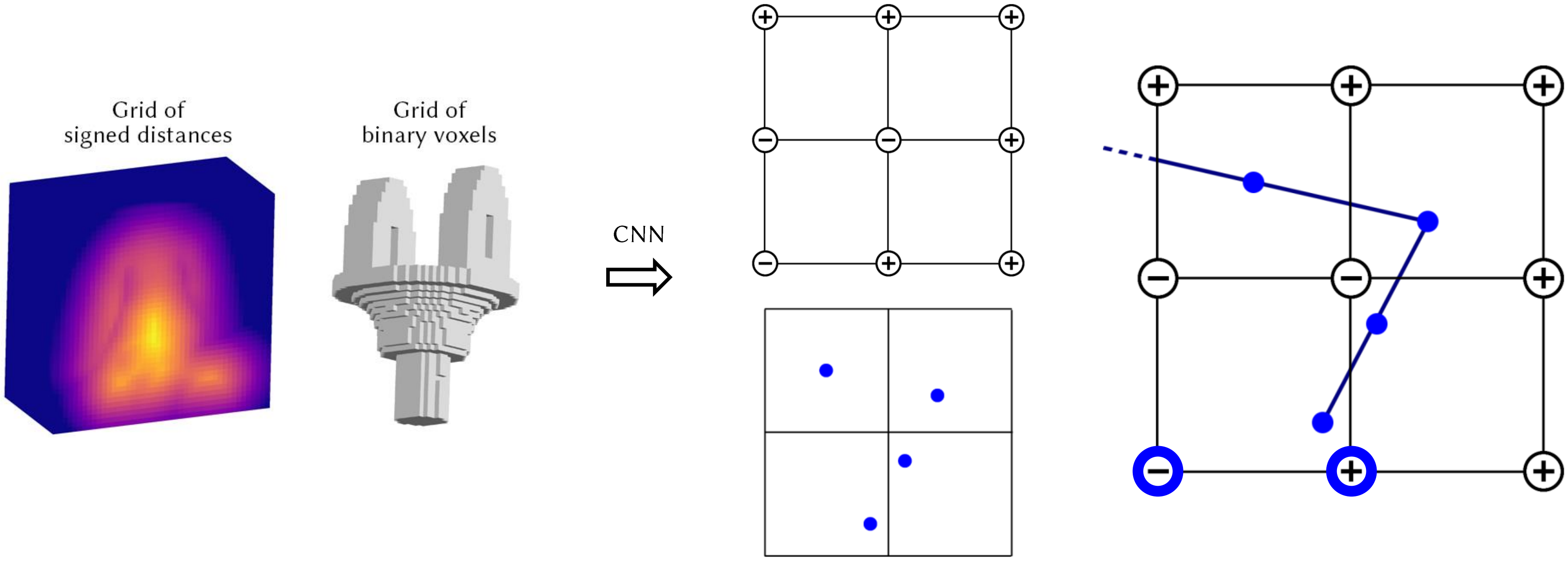
Neural Dual Contouring (NDC)



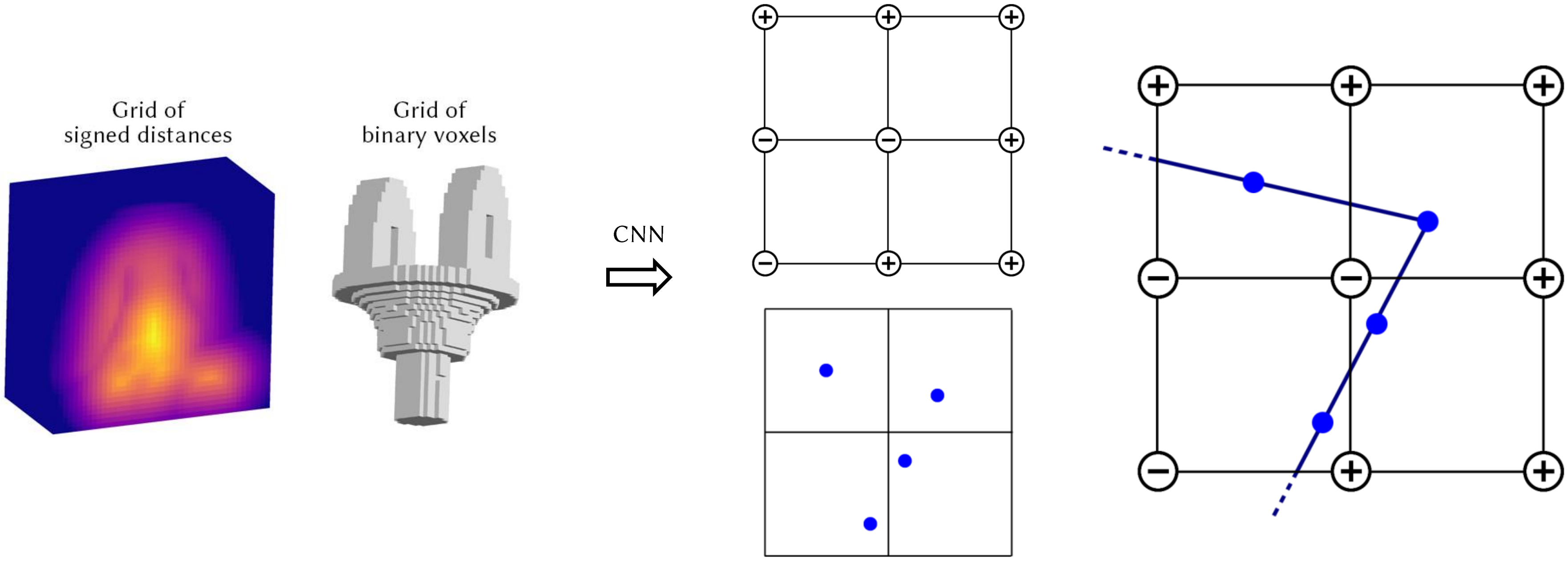
Neural Dual Contouring (NDC)



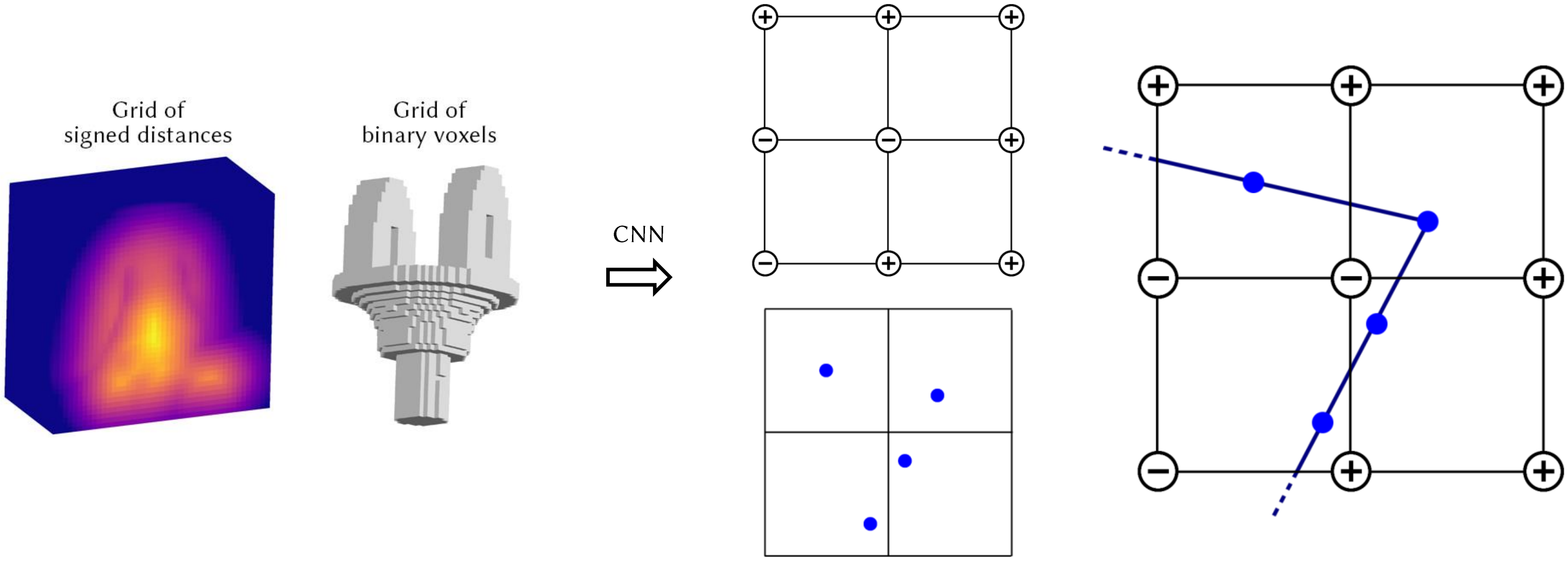
Neural Dual Contouring (NDC)



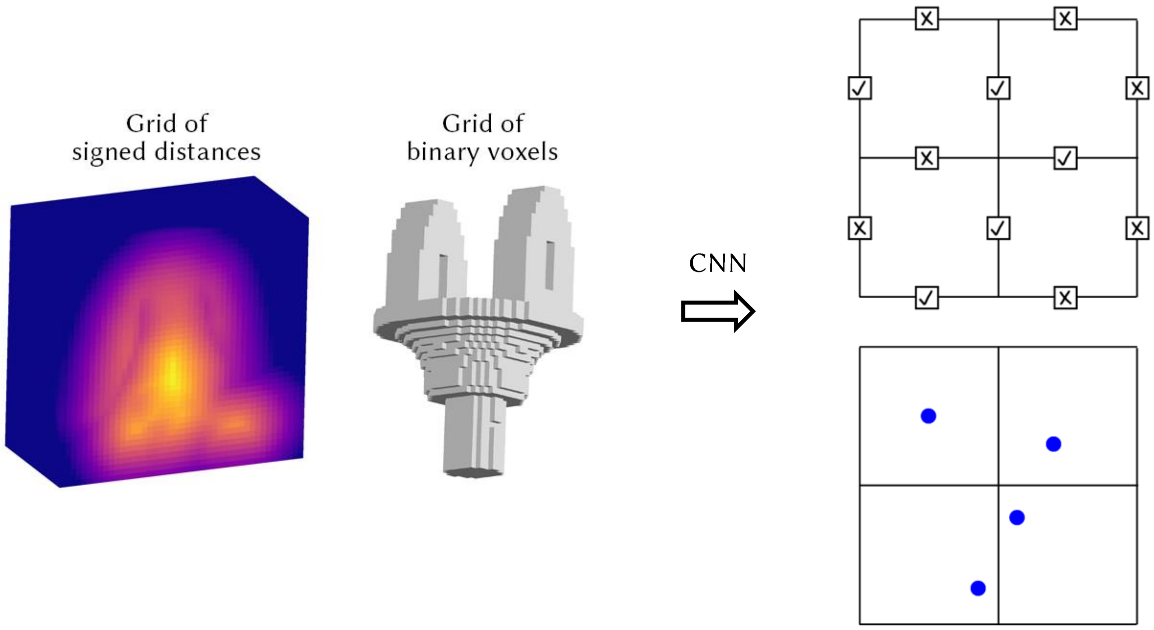
Neural Dual Contouring (NDC)



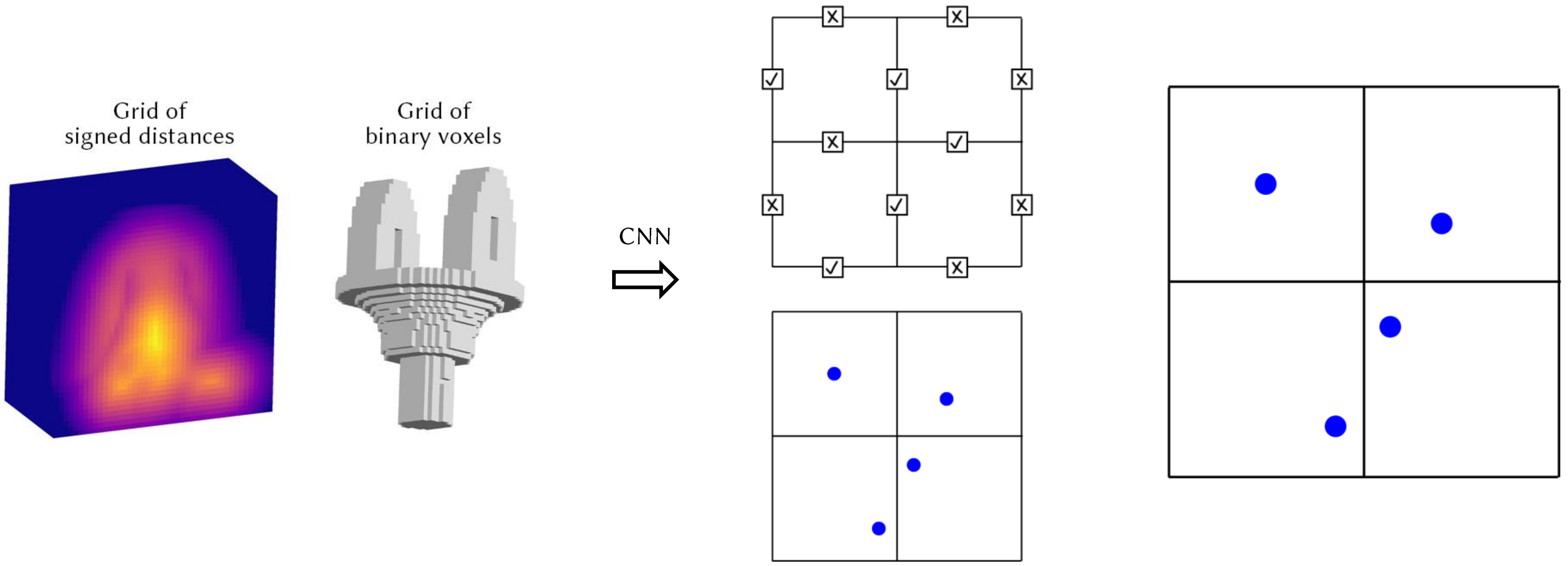
Neural Dual Contouring (NDC)



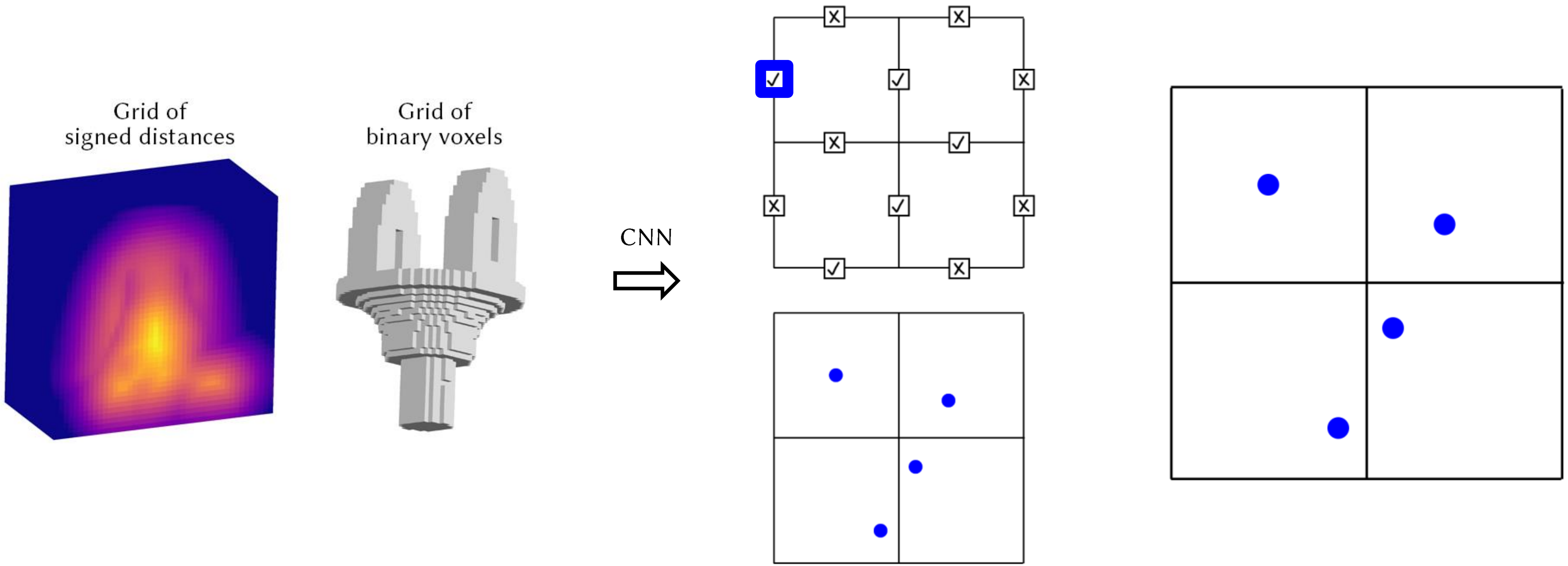
Unsigned Neural Dual Contouring (UNDC)



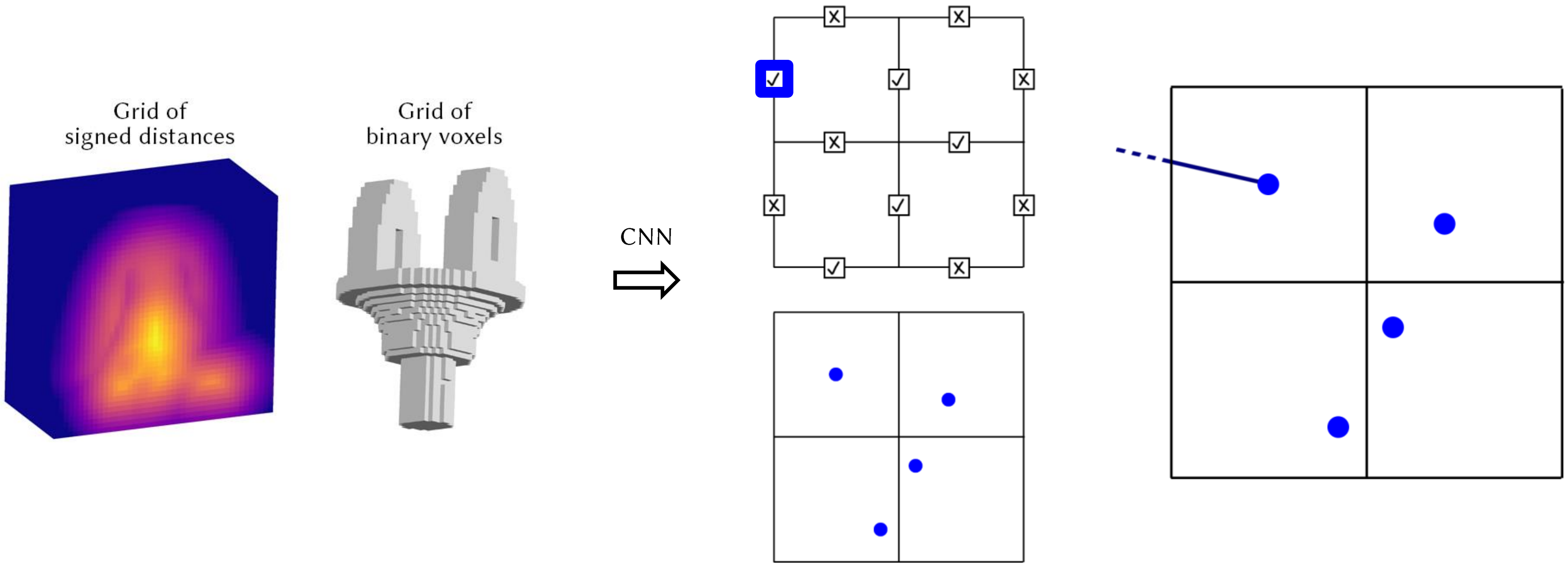
Unsigned Neural Dual Contouring (UNDC)



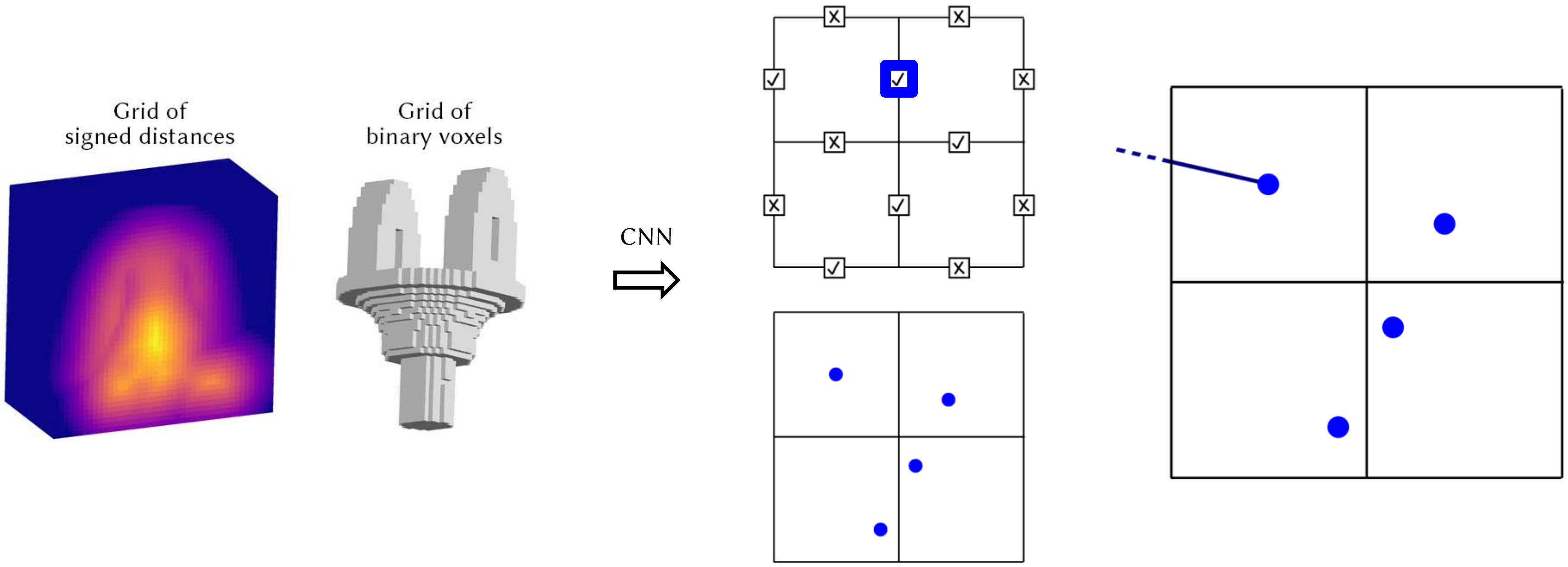
Unsigned Neural Dual Contouring (UNDC)



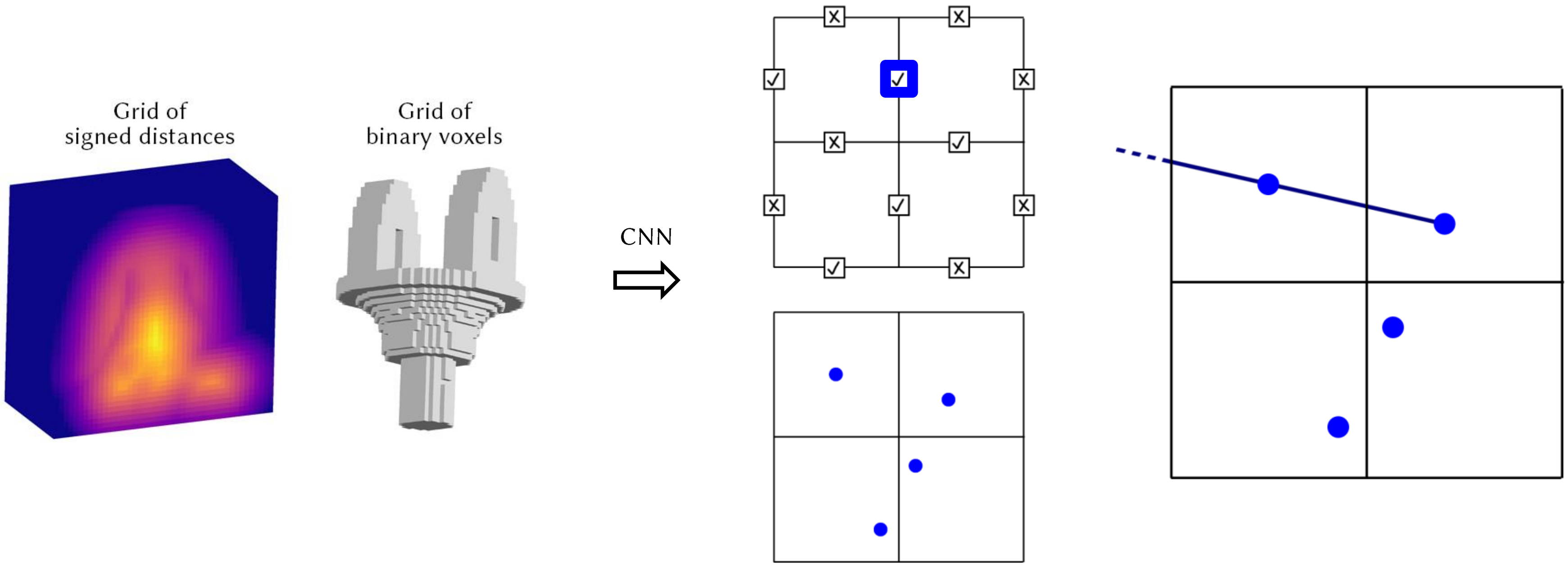
Unsigned Neural Dual Contouring (UNDC)



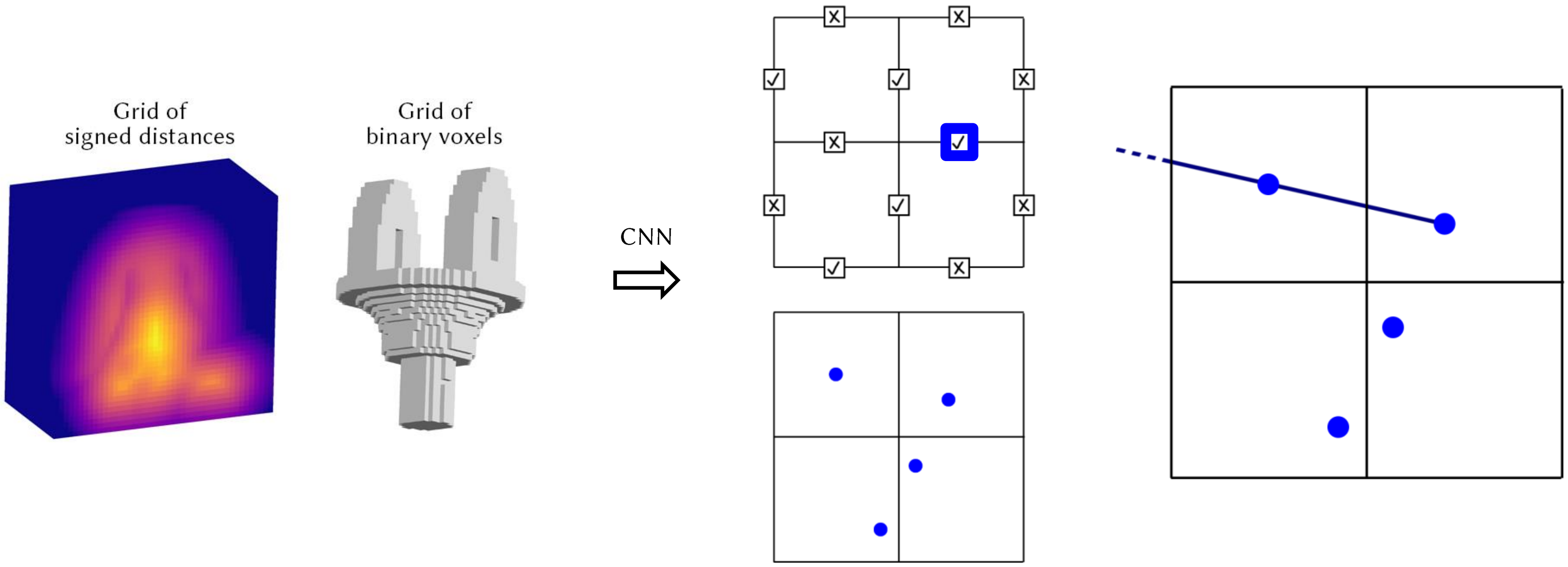
Unsigned Neural Dual Contouring (UNDC)



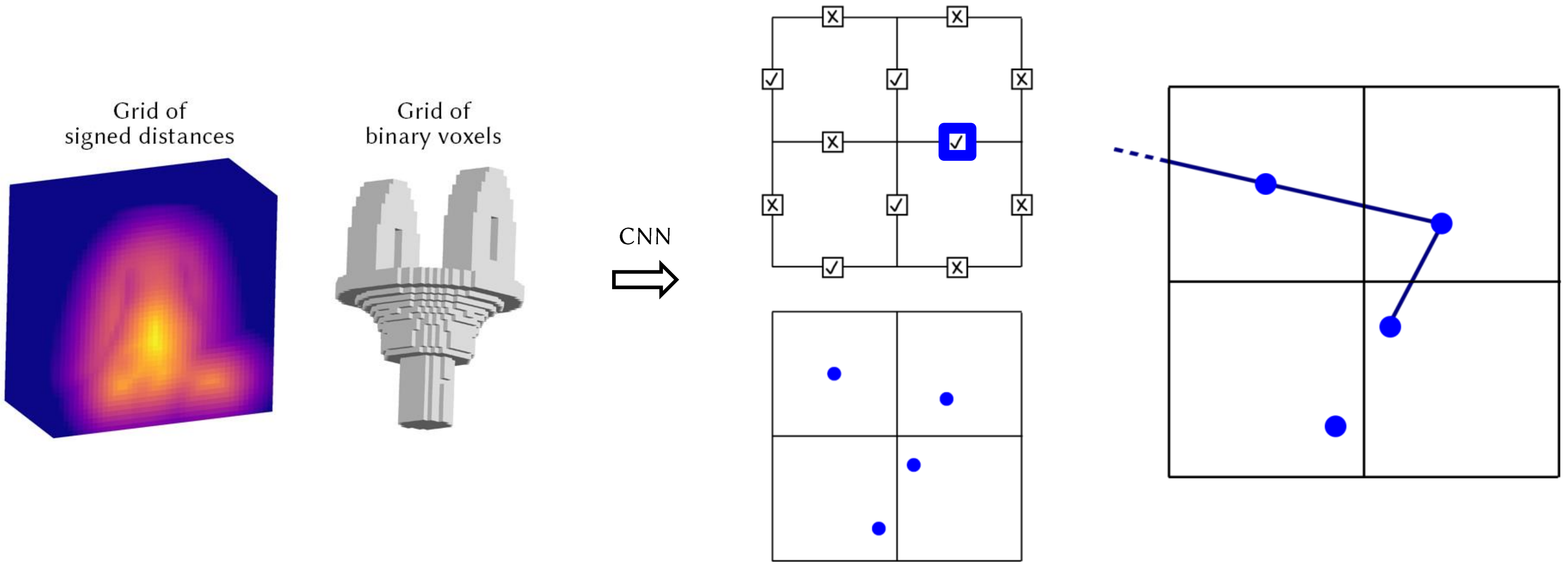
Unsigned Neural Dual Contouring (UNDC)



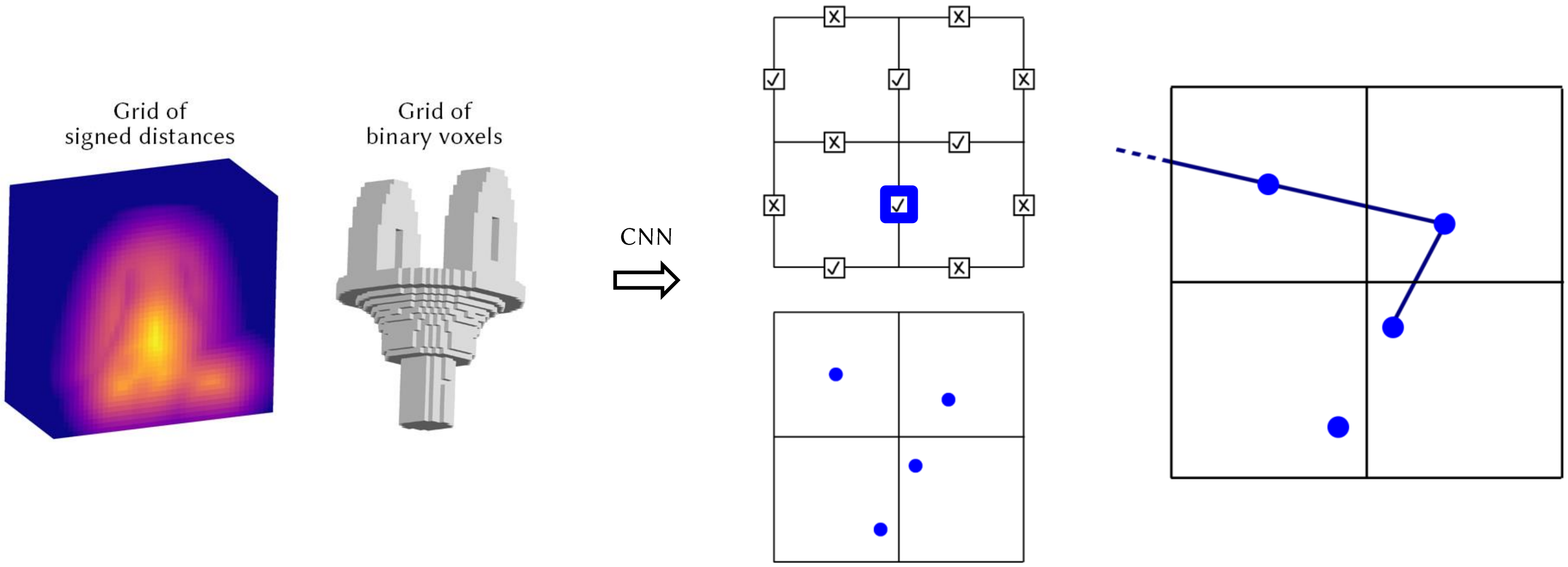
Unsigned Neural Dual Contouring (UNDC)



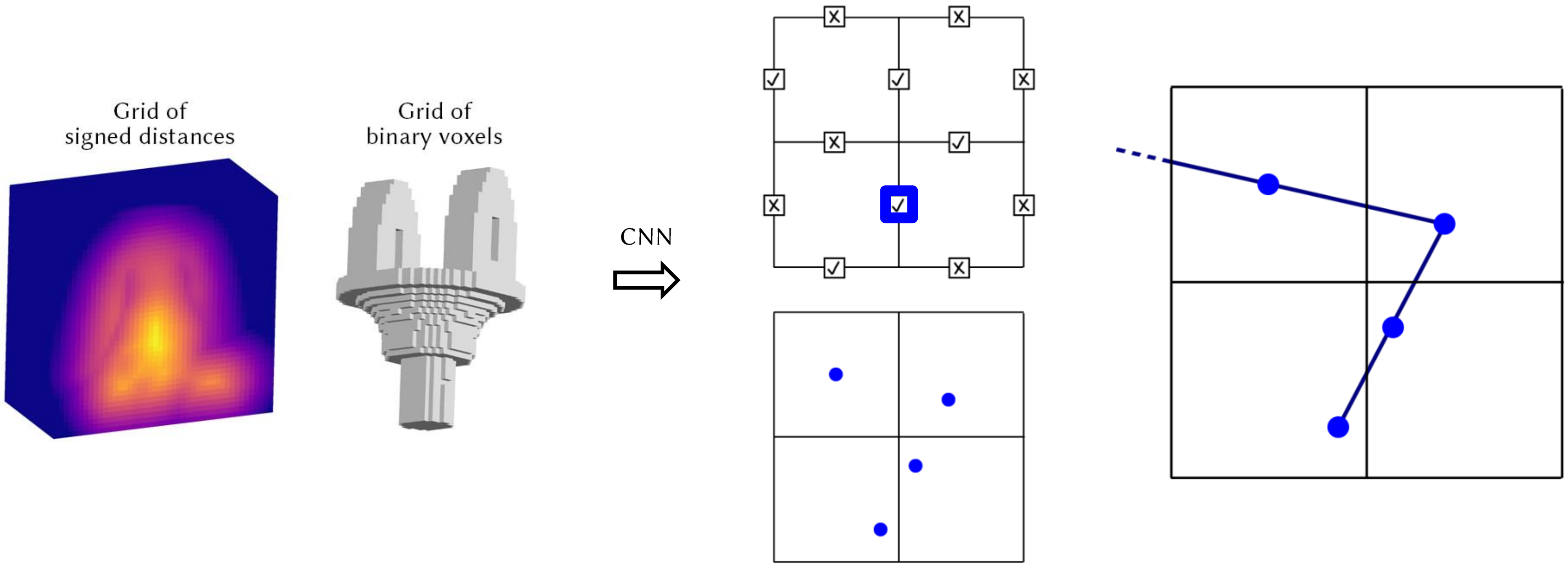
Unsigned Neural Dual Contouring (UNDC)



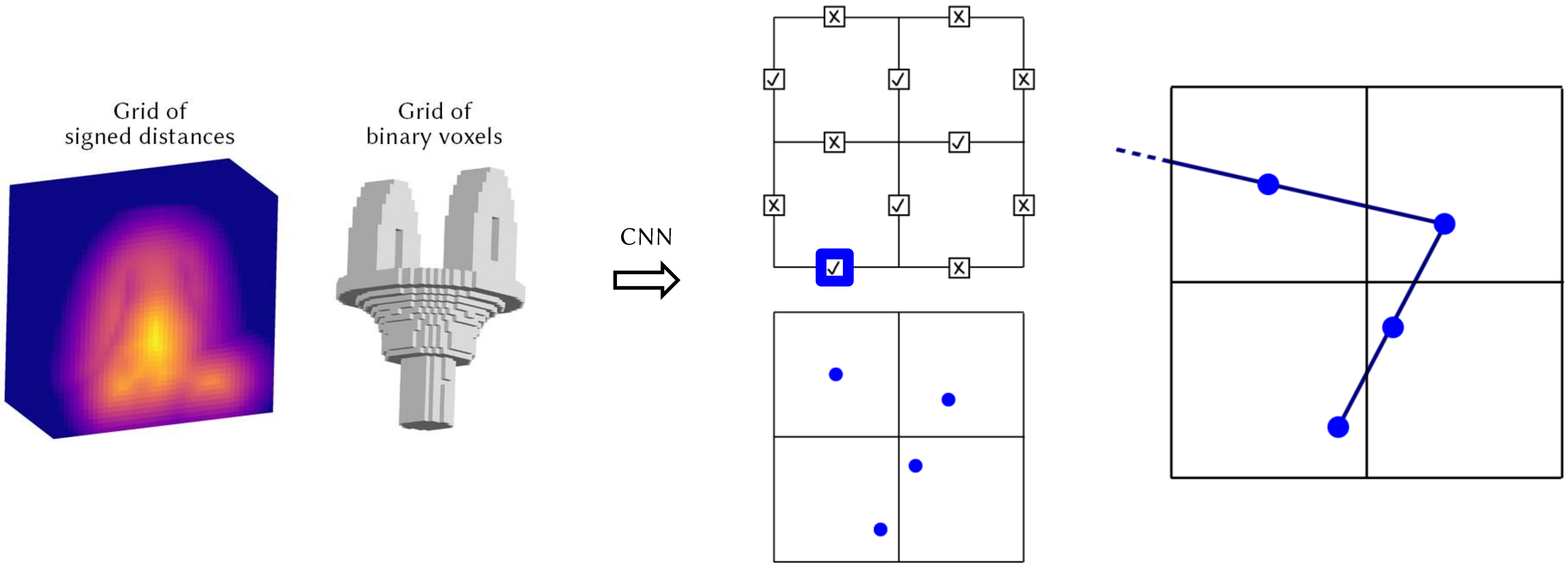
Unsigned Neural Dual Contouring (UNDC)



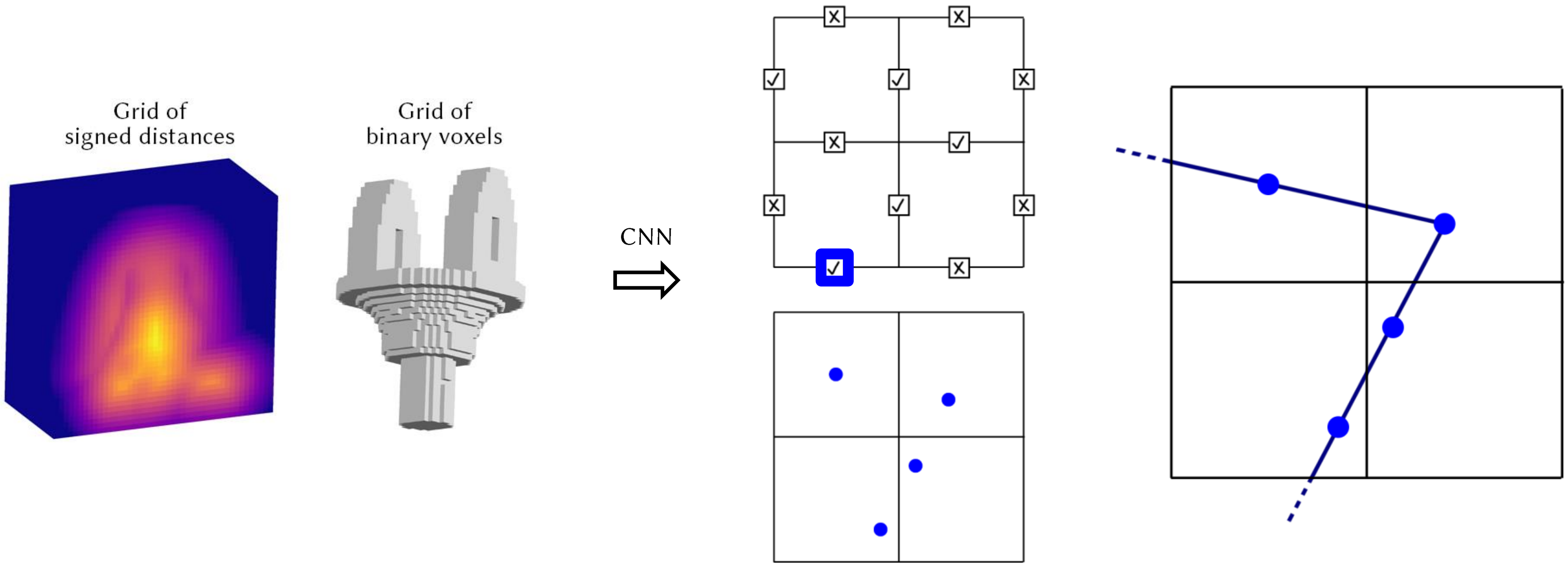
Unsigned Neural Dual Contouring (UNDC)



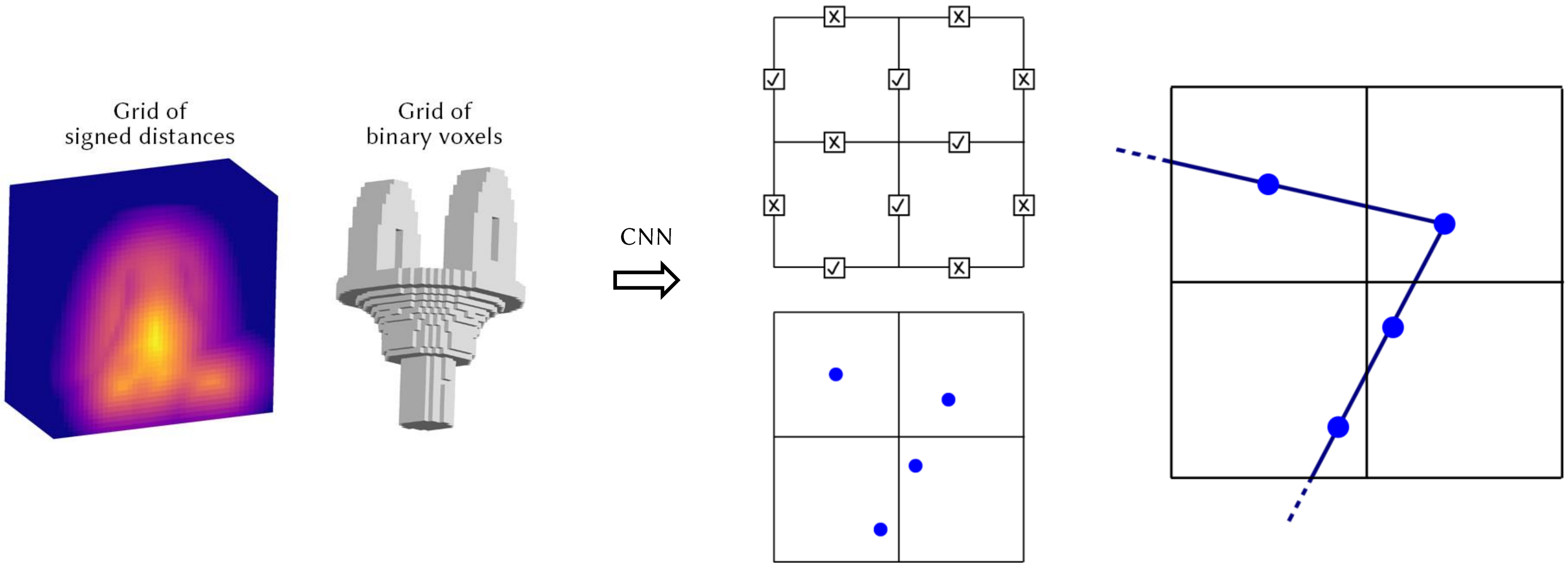
Unsigned Neural Dual Contouring (UNDC)



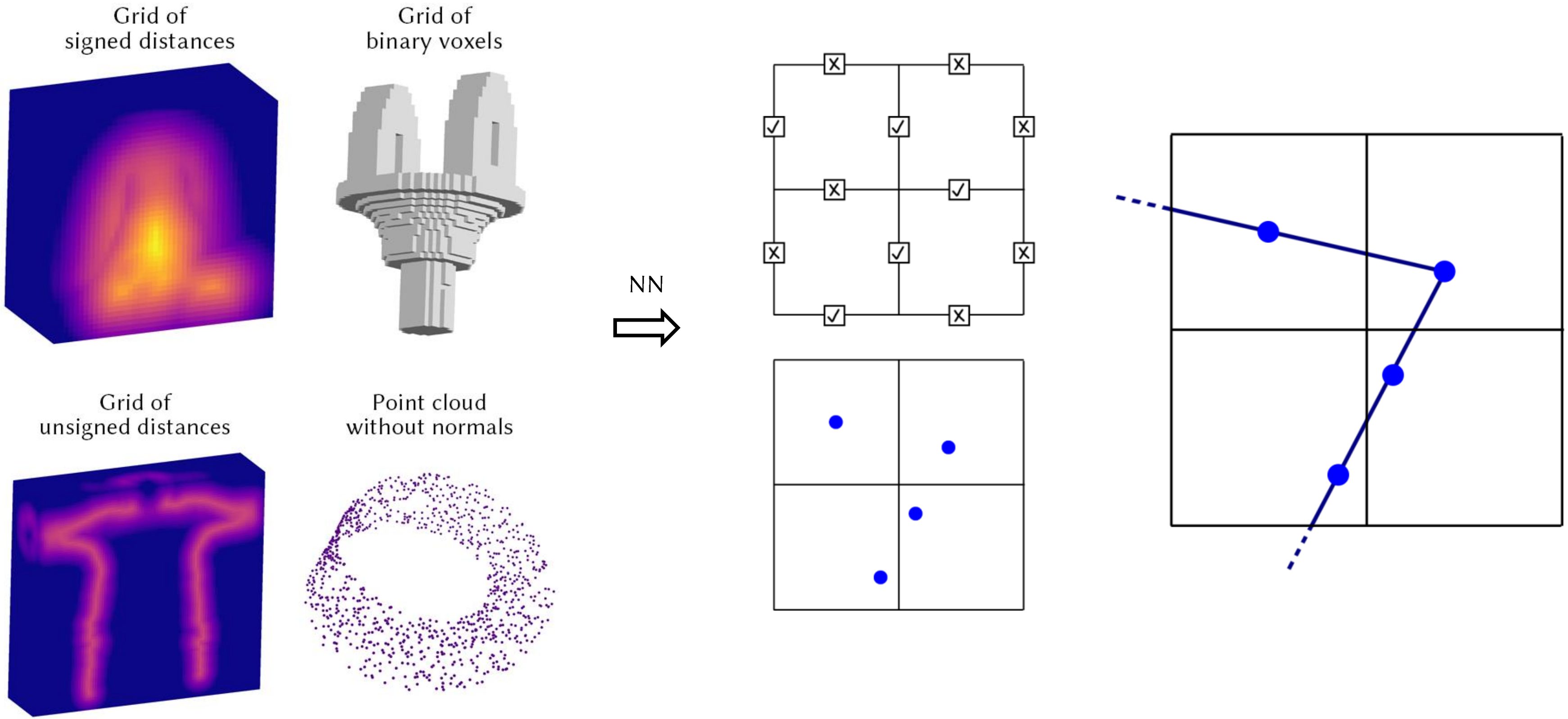
Unsigned Neural Dual Contouring (UNDC)



Unsigned Neural Dual Contouring (UNDC)

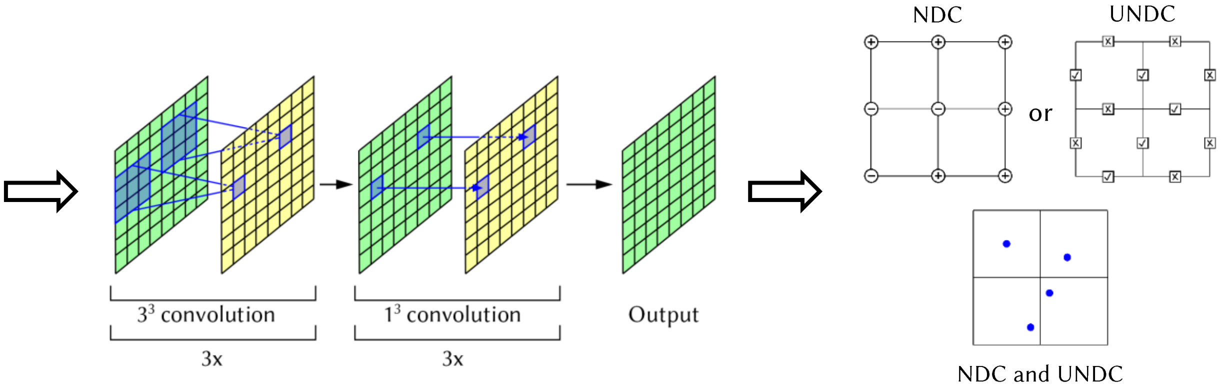


Unsigned Neural Dual Contouring (UNDC)

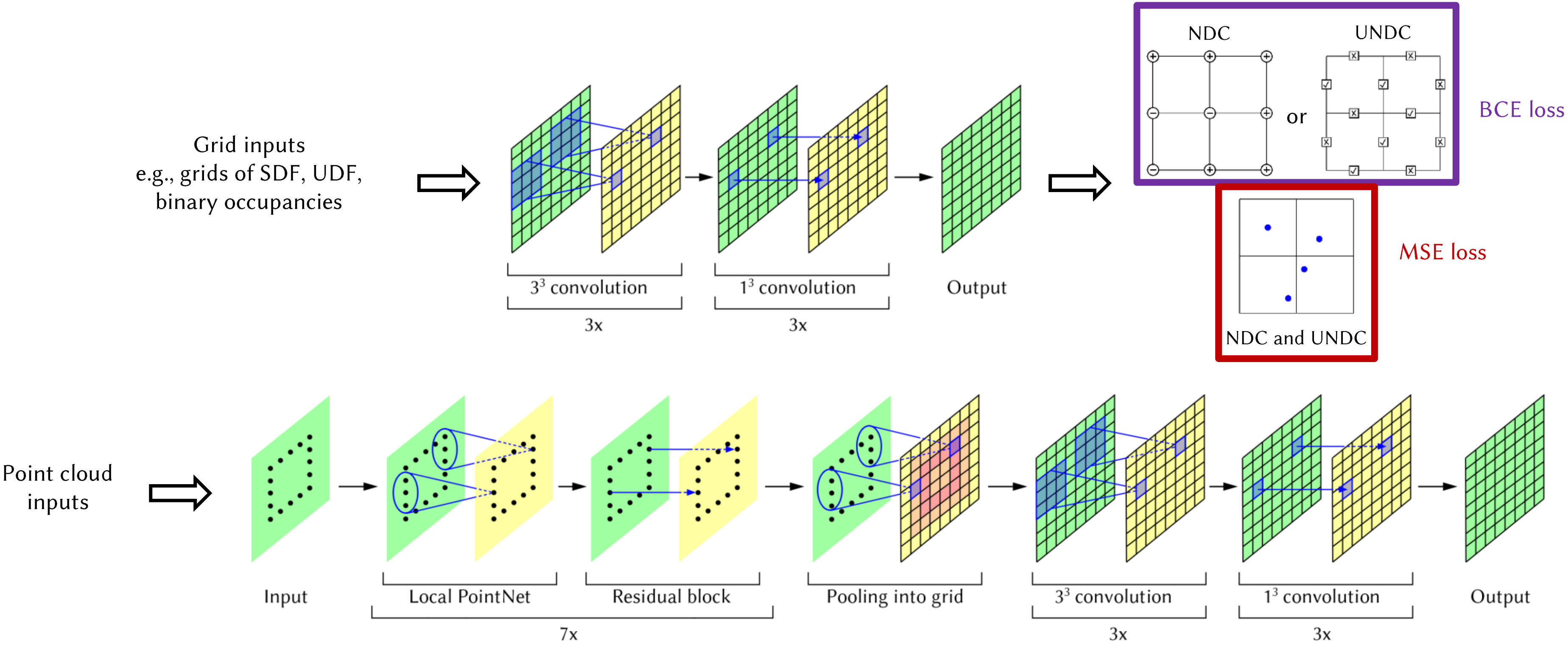


Networks

Grid inputs
e.g., grids of SDF, UDF,
binary occupancies



Networks



[3] PointNet++: Deep hierarchical feature learning on point sets in a metric space. Qi et al. NeurIPS, 2017.

Experiments

1. On grids of signed distances
2. On grids of unsigned distances
3. On grids of binary voxels
4. On point clouds without normals
5. On real scans (dense noisy point clouds without normals)

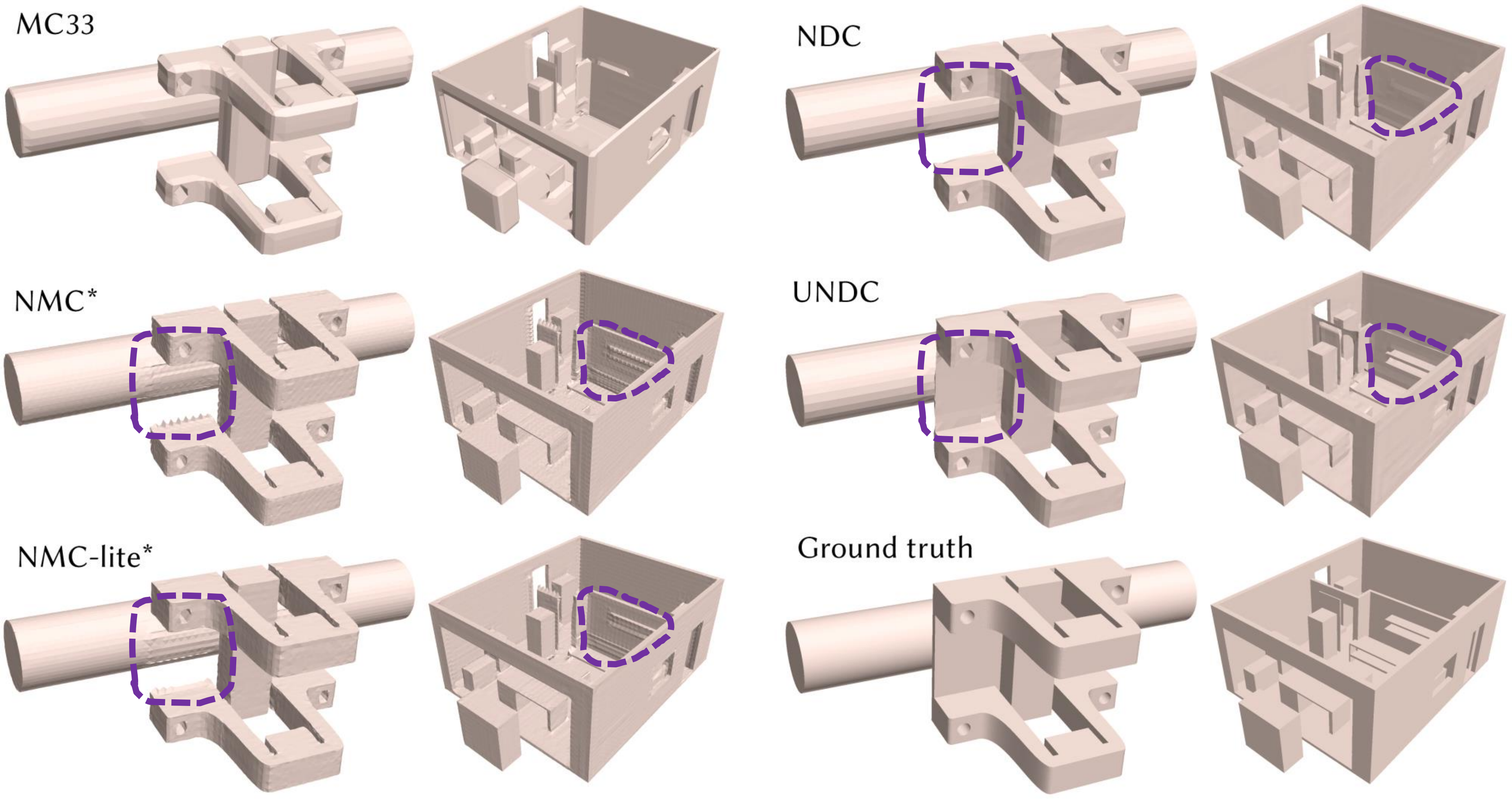
Reconstruction from grids of signed distances

Our methods:

1. NDC
2. UNDC

Compare with:

1. Marching Cubes 33 (MC33)
2. Dual Contouring with estimated normals (DC-est)
3. Neural Marching Cubes with smaller networks (NMC* and NMC-lite*)


MC33
NDC
NMC*
UNDC
NMC-lite*
Ground truth

Quantitative results

Table 2. Quantitative evaluation on **ABC** with **SDF** (signed or unsigned) inputs at two resolutions, evaluated on the test set split

64^3 SDF input	CD↓ ($\times10^5$)	F1↑	NC↑	ECD↓ ($\times10^2$)	EF1↑	#V	#T	Inference time
DC-est	4.673	0.827	0.958	3.810	0.167	**5,459**	10,969	0.421s
MC33	4.873	0.788	0.950	5.759	0.103	5,473	**10,954**	**0.005s**
NMC*	4.400	0.874	0.972	0.409	0.715	42,767	85,544	0.158s
NMC-lite*	4.386	**0.875**	0.973	0.416	0.725	21,933	43,877	0.153s
NDC	4.463	0.867	0.970	0.338	0.745	**5,459**	10,969	0.027s
UNDC	**0.930**	0.873	**0.974**	**0.328**	**0.746**	5,584	11,295	0.051s

128^3 SDF input	CD↓ ($\times10^5$)	F1↑	NC↑	ECD↓ ($\times10^2$)	EF1↑	#V	#T	Inference time
DC-est	4.132	0.879	0.977	2.215	0.266	22,088	44,213	1.765s
MC33	4.144	0.870	0.972	4.247	0.193	**22,048**	**44,107**	**0.030s**
NMC*	4.116	0.882	0.978	0.257	0.779	175,926	351,867	1.126s
NMC-lite*	4.114	0.882	0.979	0.283	0.785	88,419	176,853	1.112s
NDC	4.131	0.881	0.978	0.214	0.802	22,088	44,213	0.207s
UNDC	**0.789**	**0.890**	**0.983**	**0.149**	**0.813**	22,578	45,411	0.410s

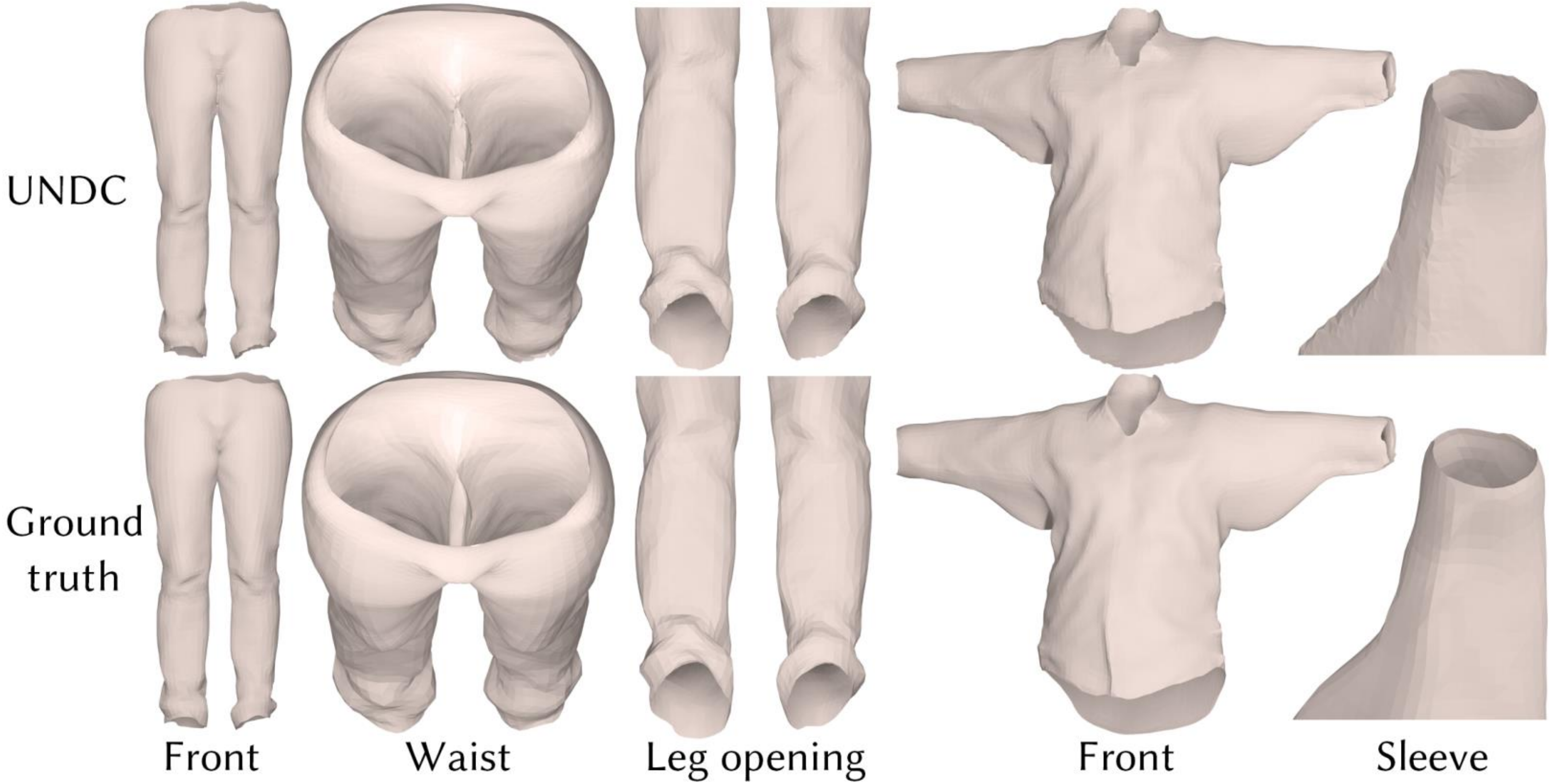
Table 3. Quantitative results on **Thingi10K** with **SDF** input.

128^3 SDF input	CD↓ ($\times10^5$)	F1↑	ECD↓ ($\times10^2$)	EF1↑	#V	#T	% IN > 5°	% SA < 10°
MC33	2.421	0.890	2.657	0.197	22,324	44,656	19.08	2.43
NMC*	2.613	0.902	0.269	0.760	169,211	338,427	20.99	0.77
NMC-lite*	2.651	0.902	0.254	0.772	89,260	178,527	17.04	1.74
NDC	2.300	0.901	0.215	0.792	**22,295**	**44,631**	**12.52**	**0.24**
UNDC	**0.757**	**0.904**	**0.189**	**0.795**	22,478	45,043	12.66	0.29

Table 4. Quantitative results on **FAUST** with **SDF** input.

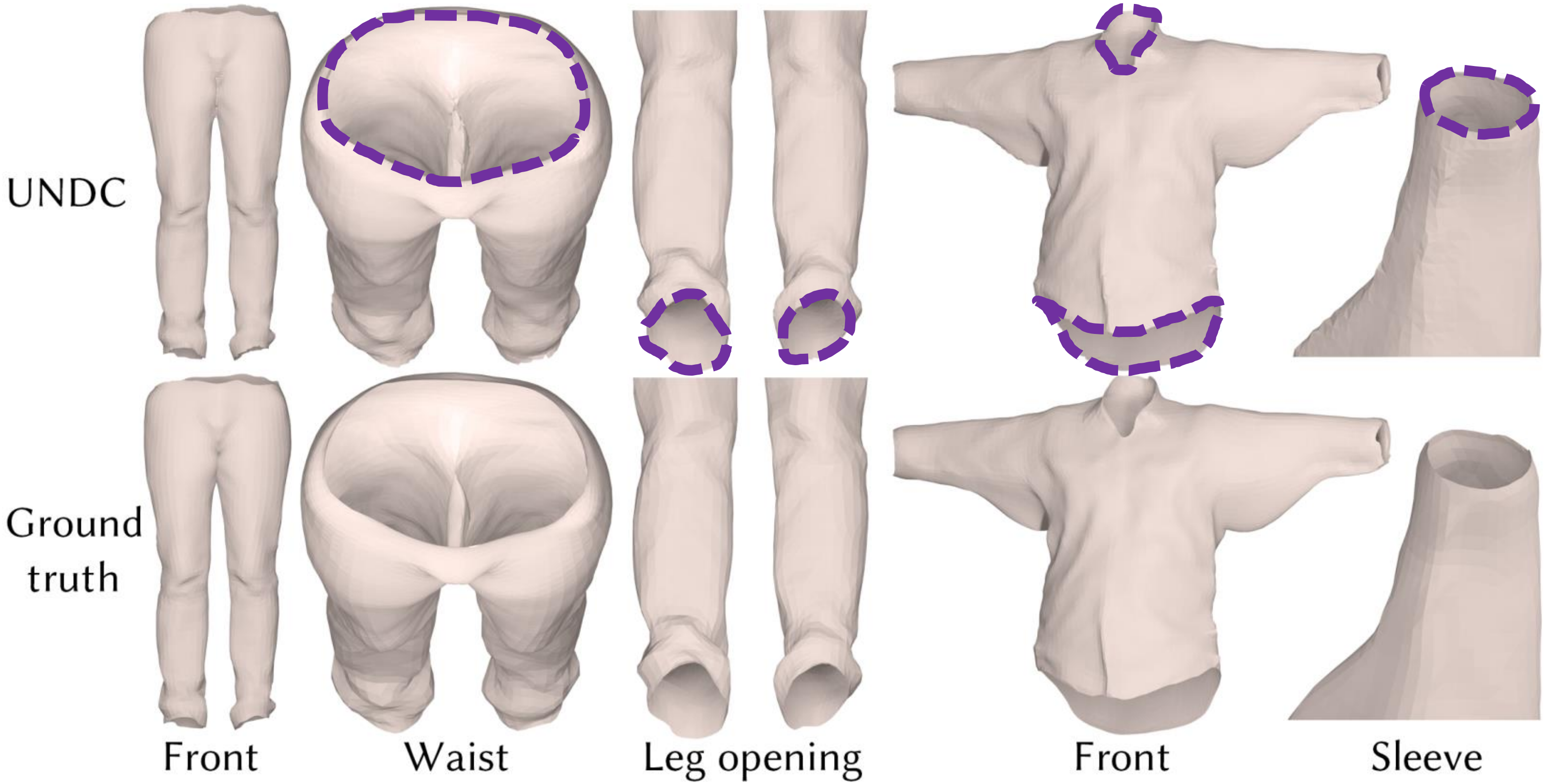
128^3 SDF input	CD↓ ($\times10^5$)	F1↑	ECD↓ ($\times10^2$)	EF1↑	#V	#T	% IN > 5°	% SA < 10°
MC33	0.453	0.985	0.086	0.387	12,551	**25,076**	**34.28**	4.23
NMC*	0.385	0.990	0.146	0.552	83,024	166,038	44.58	1.18
NMC-lite*	0.381	0.991	0.119	0.567	50,207	100,404	38.33	2.63
NDC	0.397	0.989	0.044	0.530	**12,538**	25,100	38.38	**0.11**
UNDC	**0.362**	**0.992**	**0.038**	**0.574**	12,609	25,258	37.38	0.16

Reconstruction from grids of unsigned distances



[9] Multi-Garment Net: Learning to Dress 3D People from Images. Bhatnagar et al. ICCV, 2019.

Reconstruction from grids of unsigned distances



Reconstruction from point clouds

Our method: UNDC

Compare with:

1. Ball-pivoting
2. Screened Poisson

[10] The ball-pivoting algorithm for surface reconstruction. Bernardini et al. TVCG, 1999.
[11] Screened Poisson surface reconstruction. Kazhdan et al. ACM Transactions on Graphics, 2013.

Reconstruction from point clouds

Our method: UNDC

Compare with:
1. Ball-pivoting
2. Screened Poisson

3. SIREN
4. Local Implicit Grids (LIG)

5. Convolutional Occupancy Networks (ConvONet)

[12] Implicit neural representations with periodic activation functions. Sitzmann et al. NeurIPS, 2020.
[13] Local implicit grid representations for 3d scenes. Jiang et al. CVPR, 2020.
[14] Convolutional occupancy networks. Peng et al. ECCV, 2020.

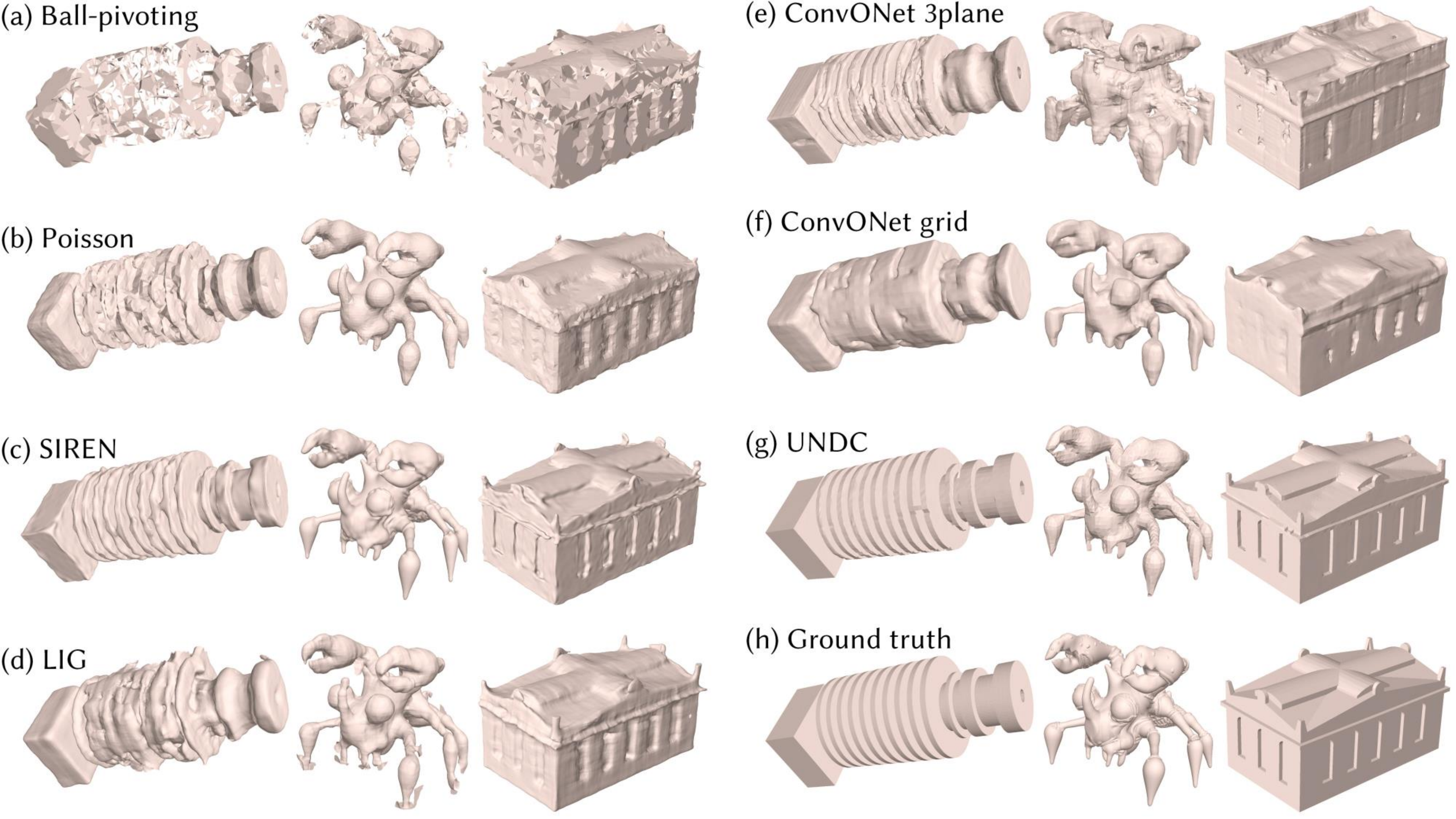
Reconstruction from point clouds

Our method: UNDC

Compare with:
1. Ball-pivoting
2. Screened Poisson
3. SIREN
4. Local Implicit Grids (LIG)
5. Convolutional Occupancy Networks (ConvONet)

Table 6. Quantitative results on **ABC** test set with **point cloud** input. (+n) indicates that the method additionally requires point normals as input.

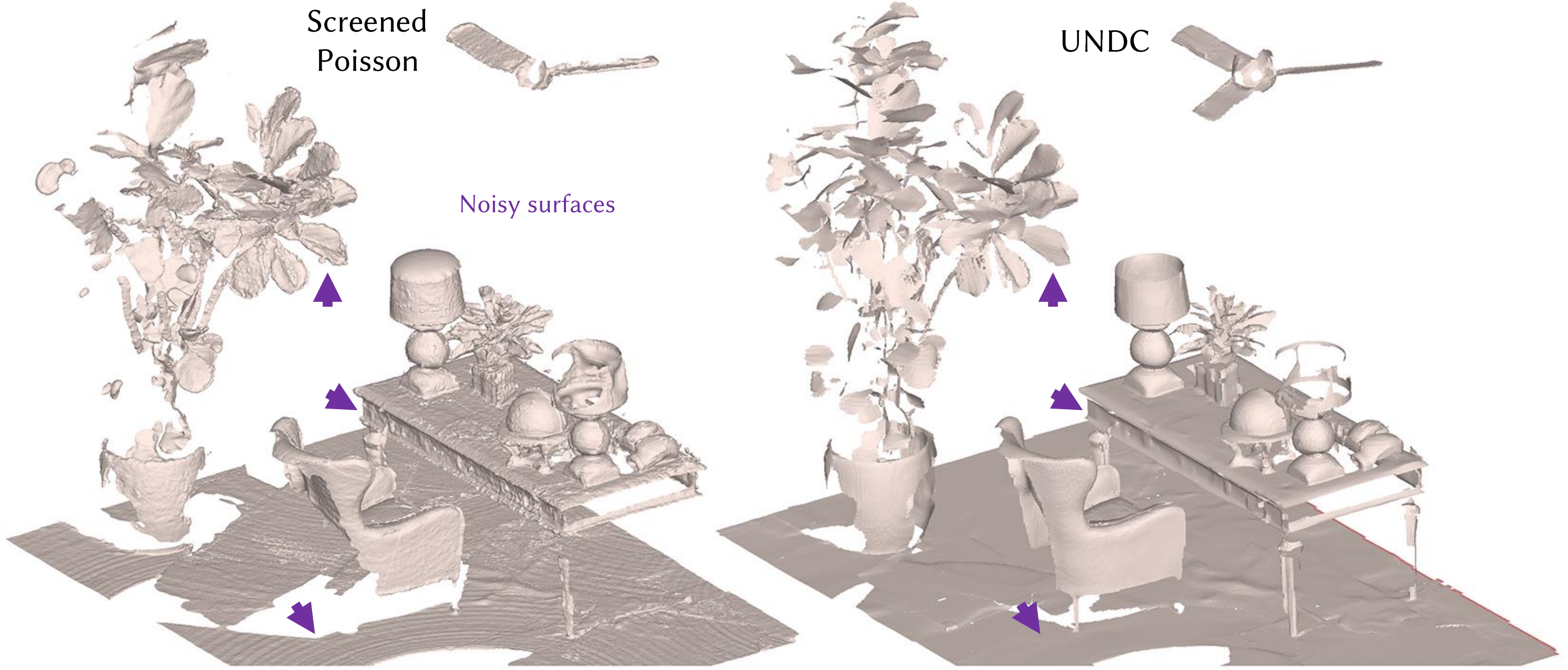
point cloud (4,096)	CD↓ ($\times 10^5$)	F1↑	NC↑	ECD↓ ($\times 10^2$)	EF1↑	#V	#T	Inference time
Ball-pivoting (+n)	3.080	0.791	0.944	0.556	0.269	**4,096**	**7,439**	1.292s
Poisson (+n)	4.705	0.727	0.939	4.138	0.067	11,241	22,496	1.476s
SIREN (+n)	1.340	0.814	0.969	2.636	0.152	97,219	194,543	168.595s
LIG (+n)	3.413	0.721	0.947	11.868	0.022	149,860	299,166	66.866s
ConvONet 3plane	18.073	0.536	0.935	4.113	0.105	75,342	150,689	2.692s
ConvONet grid	8.844	0.488	0.939	9.701	0.036	74,171	148,337	2.404s
UNDC	**0.893**	**0.873**	**0.974**	**0.289**	**0.757**	5,578	11,261	**0.194s**


(a) Ball-pivoting
(b) Poisson
(c) SIREN
(d) LIG
(e) ConvONet 3plane
(f) ConvONet grid
(g) UNDC
(h) Ground truth

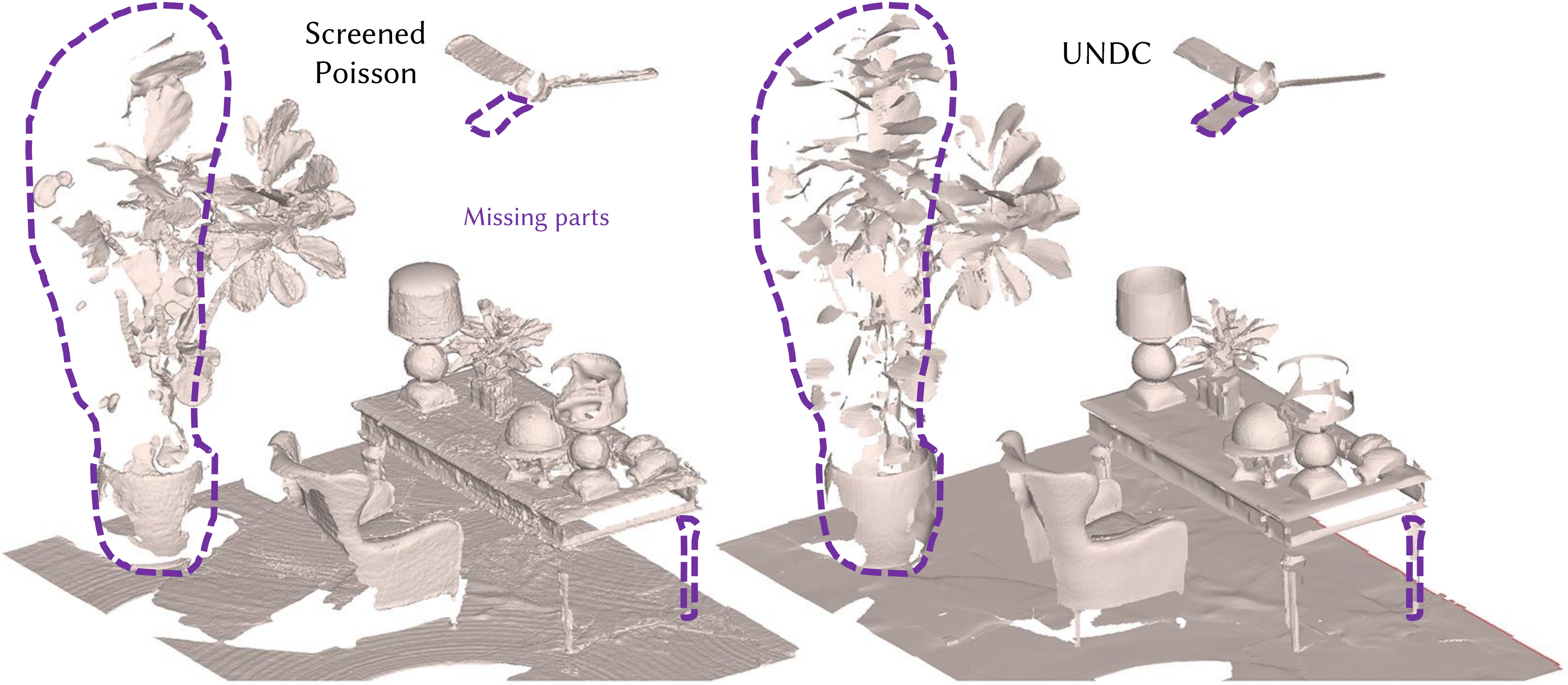
Reconstruction from noisy real scans

[15] Matterport3D: Learning from RGB-D Data in Indoor Environments. Chang et al. 3DV, 2017.

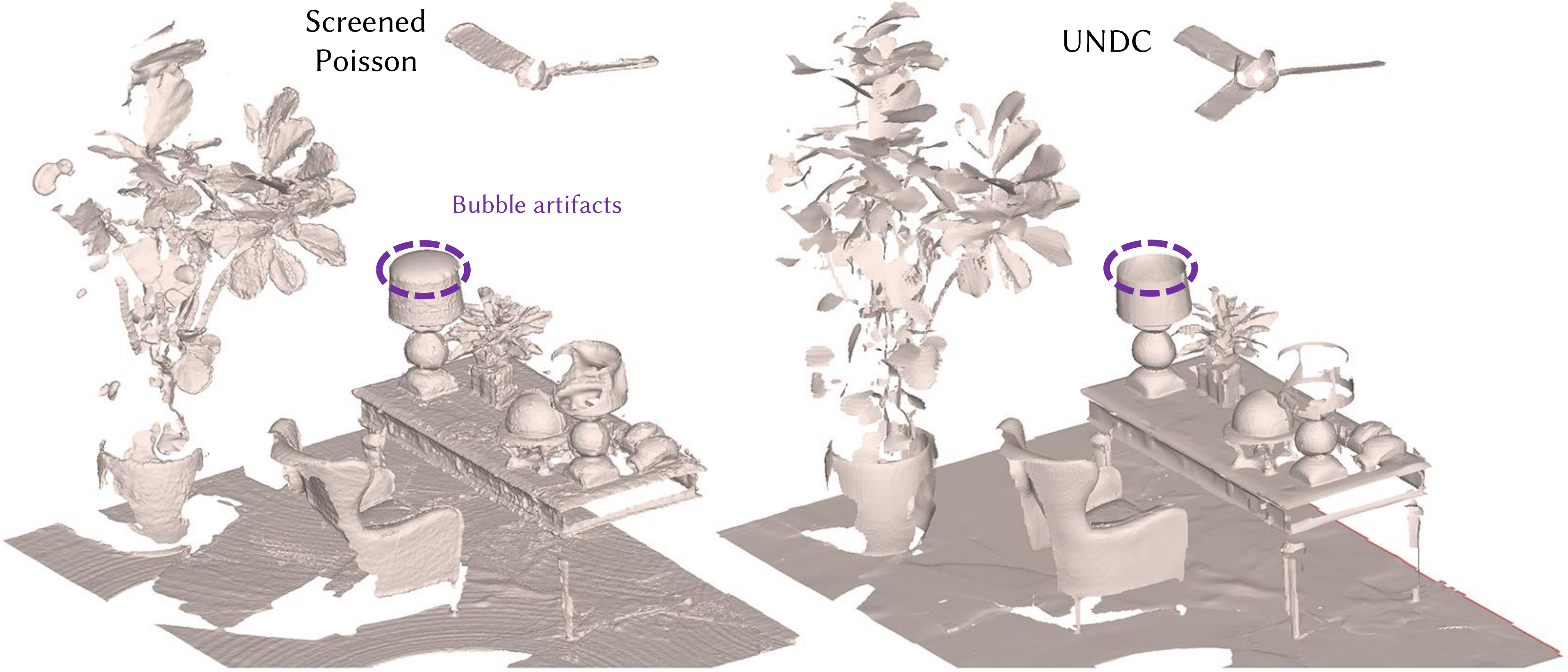
Reconstruction from noisy real scans



Reconstruction from noisy real scans

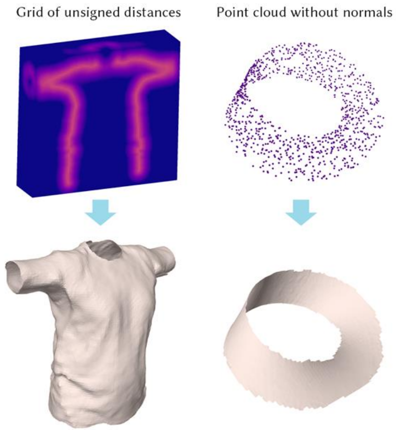


Reconstruction from noisy real scans

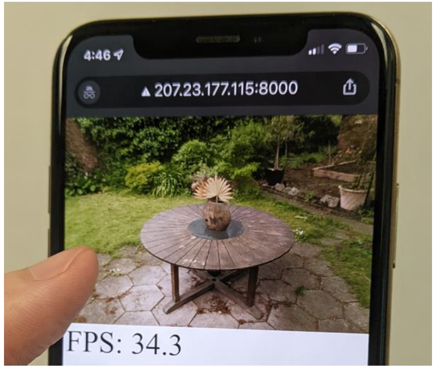


Overview

Neural Marching Cubes
(SIGGRAPH Asia 2021)



Neural Dual Contouring
(SIGGRAPH 2022)

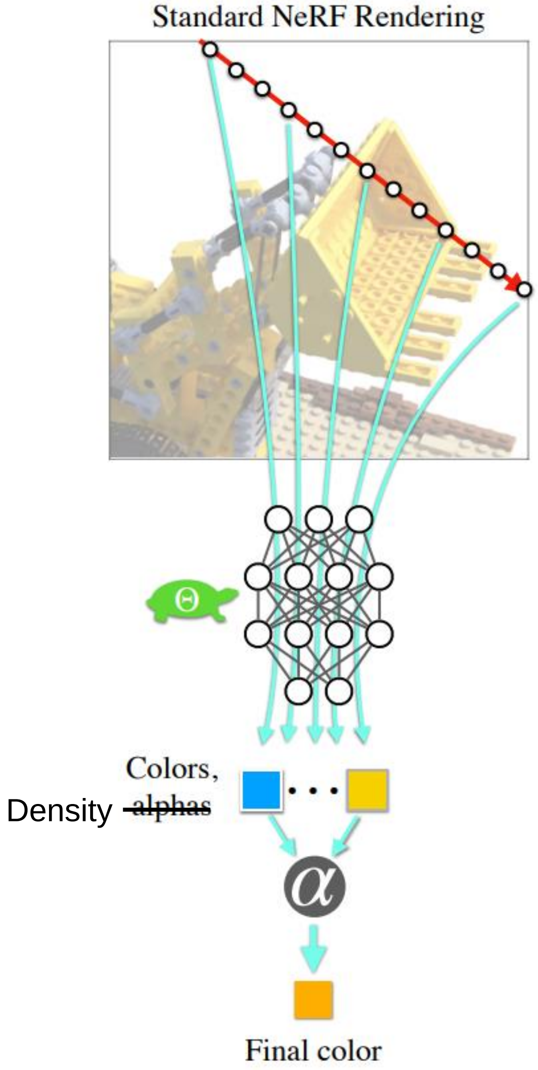


MobileNeRF
(Arxiv 2022)

Motivation

Traditional NeRF methods rely on volumetric rendering.

> Slow: because many sampled points have to be evaluated for each ray (pixel).



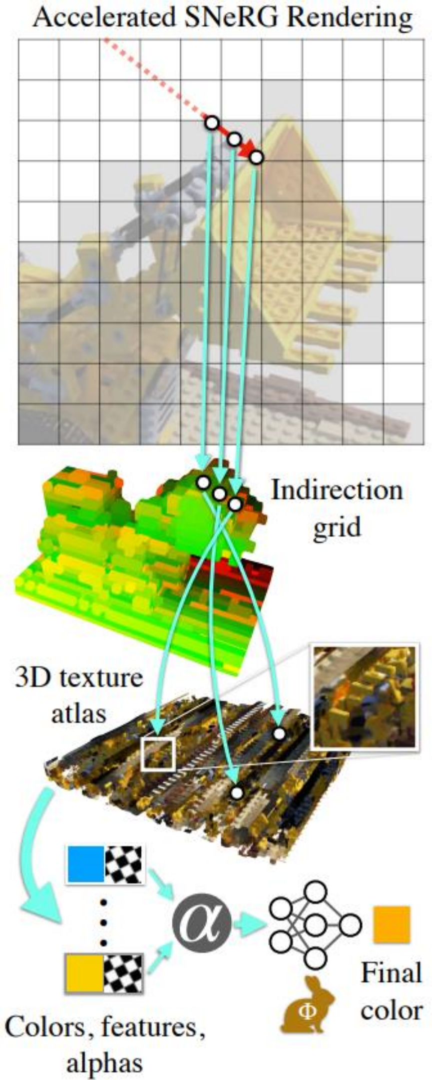
* The figure is taken from SNeRG - Baking Neural Radiance Fields for Real-Time View Synthesis, Hedman et al. ICCV 2021.

Motivation

Recent NeRF methods speed up inference by "baking" the MLP evaluation results into sparse 3D voxel grids.

E.g., SNeRG, PlenOctrees.

> Large: Because 3D texture has be to stored in GPU for fast accessing.

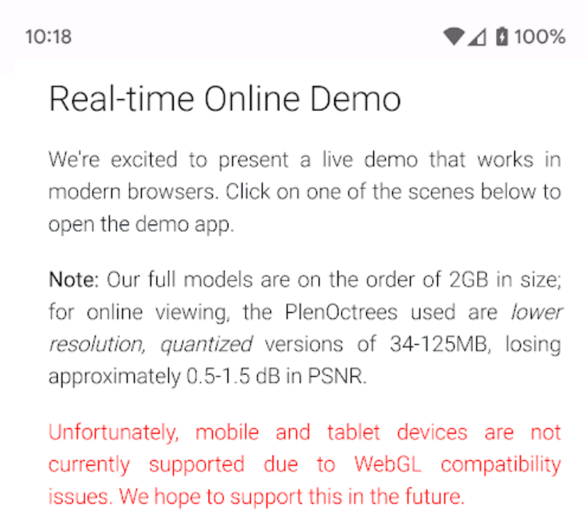


* The figure is taken from SNeRG - Baking Neural Radiance Fields for Real-Time View Synthesis, Hedman et al. ICCV 2021.

Motivation

> Compatibility: Most NeRF methods need cuda and high-end machines.

SNeRG



PlenOctrees

Our method

We want to use textured triangle mesh as the representation.

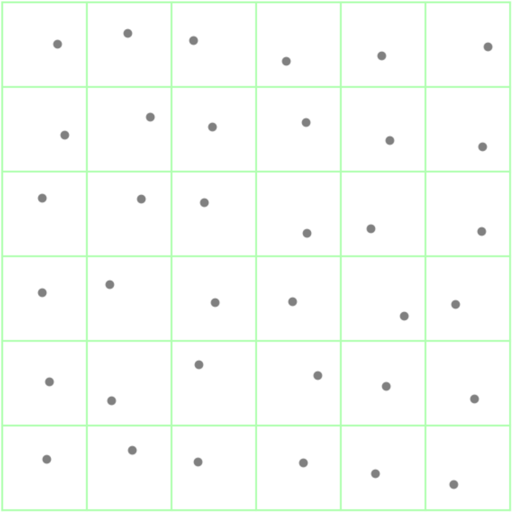
> Compatibility: All GPUs in modern devices can render triangles.

> Speed: GPUs are optimized to render triangles extremely fast.

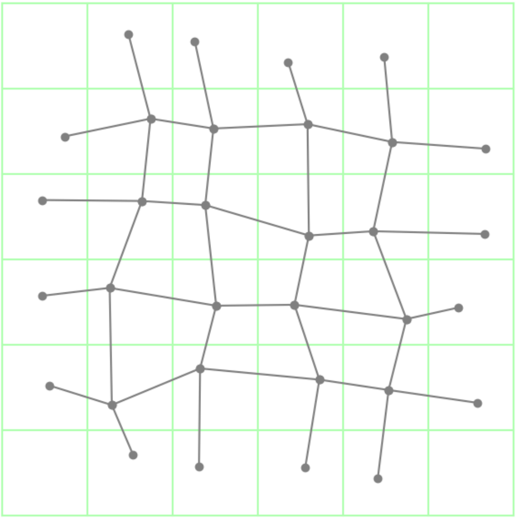
> Memory: Storing 2D textures consumes much less memory than 3D textures.

UNDC + Differentiable renderer

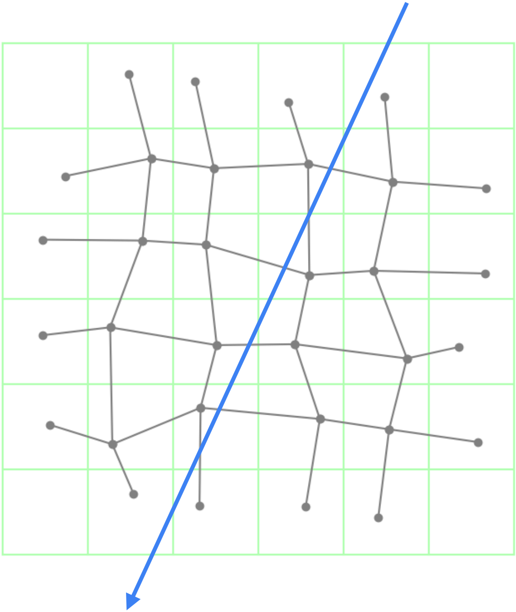
1. Store a grid of vertices



2. Connect adjacent vertices to form faces



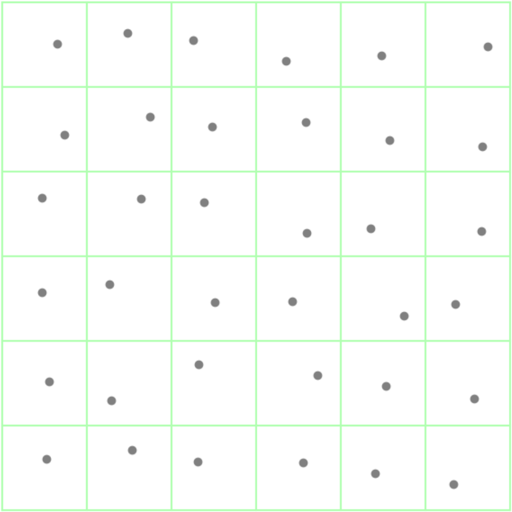
3. Compute intersections; then do NeRF



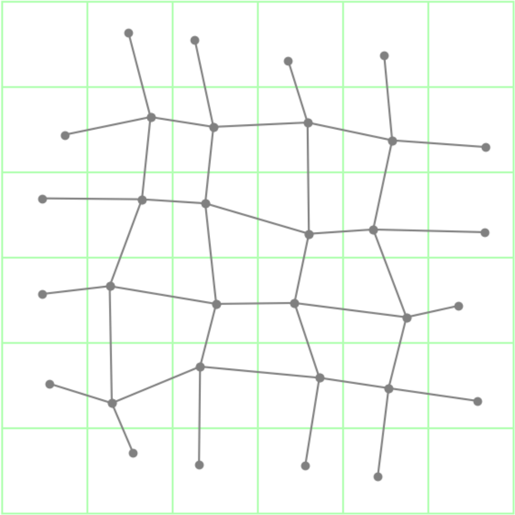
UNDC + Differentiable renderer

Intersection computation
is efficient!

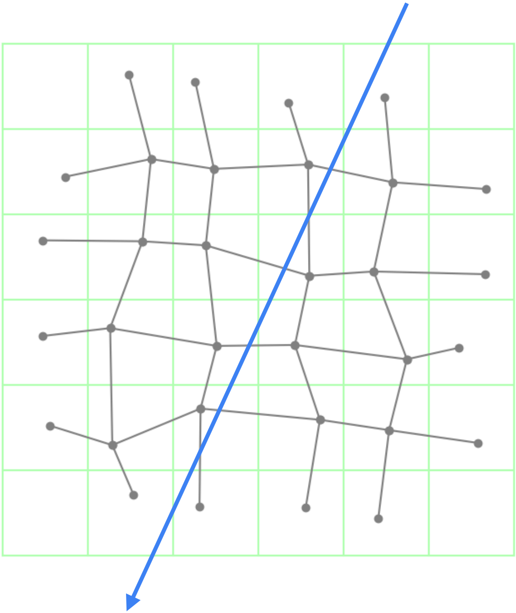
1. Store a grid of vertices



2. Connect adjacent
vertices to form faces

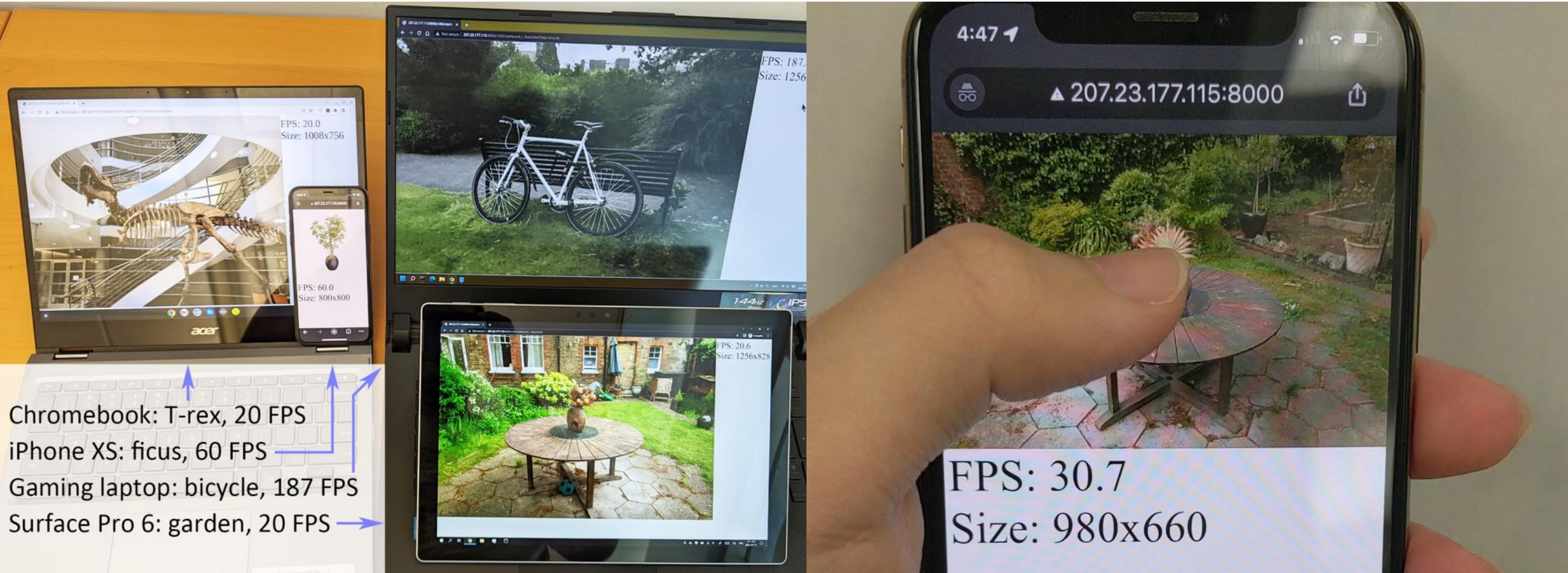


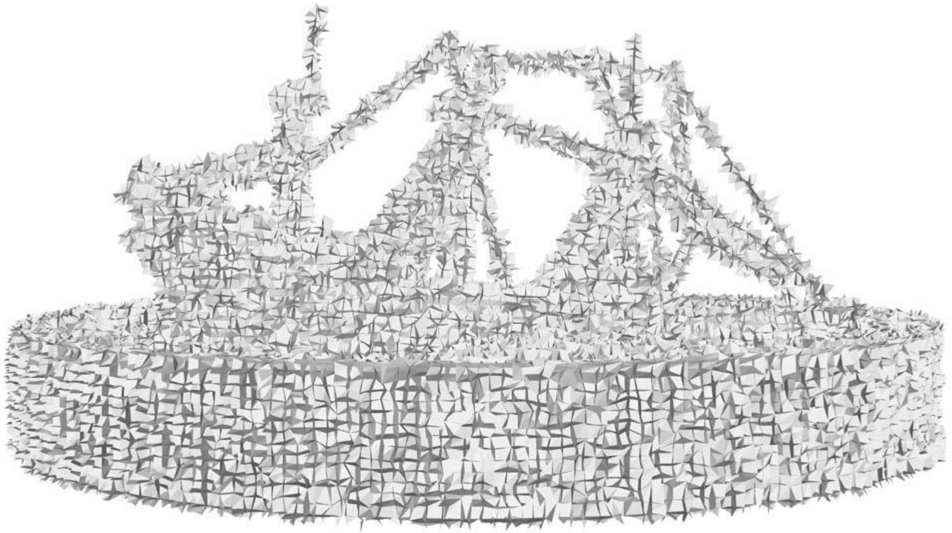
3. Compute intersections;
then do NeRF



MobileNeRF: Exploiting the Polygon Rasterization Pipeline for Efficient Neural Field Rendering on Mobile Architectures

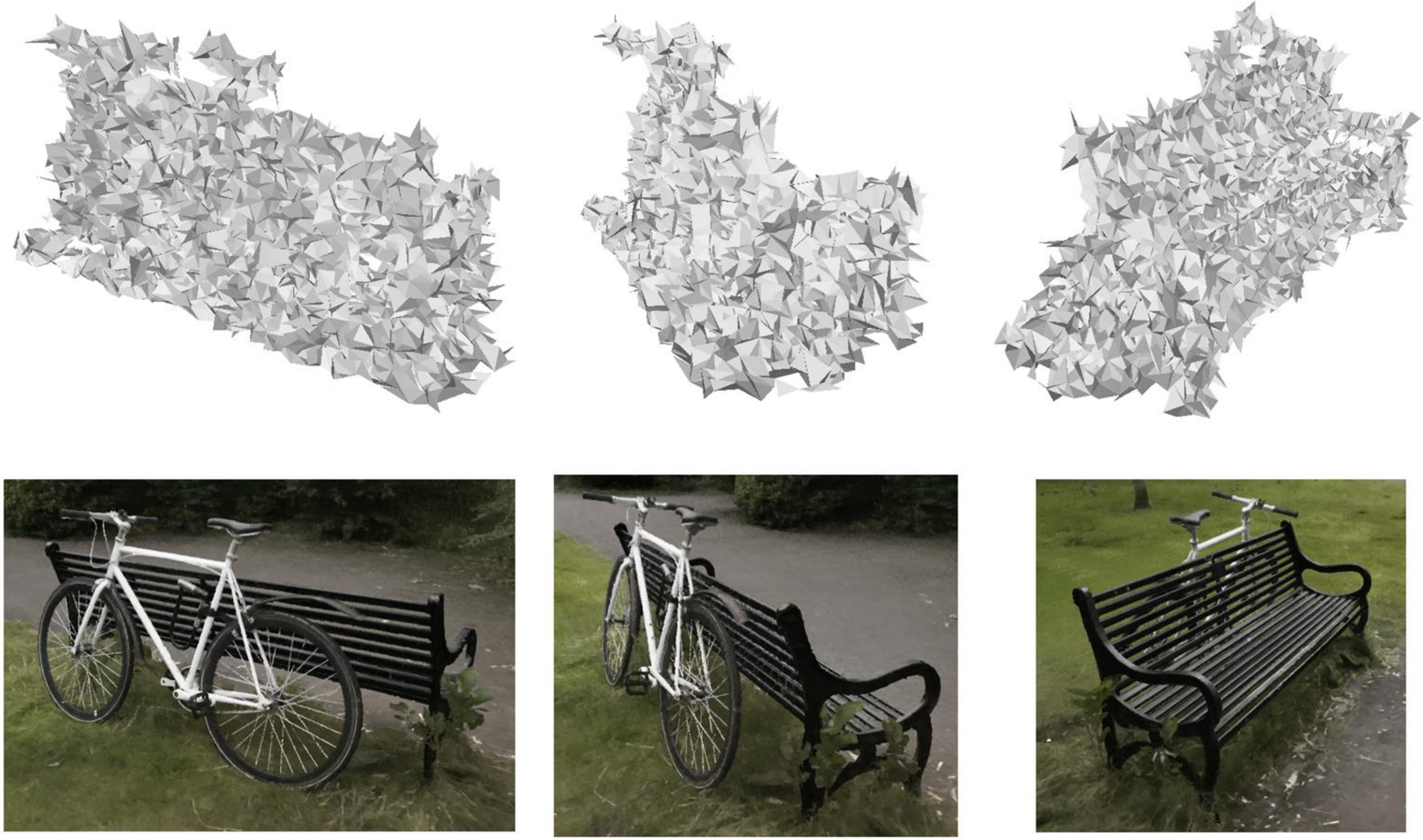
Zhiqin Chen, Peter Hedman, Thomas Funkhouser, Andrea Tagliasacchi





Our triangle mesh

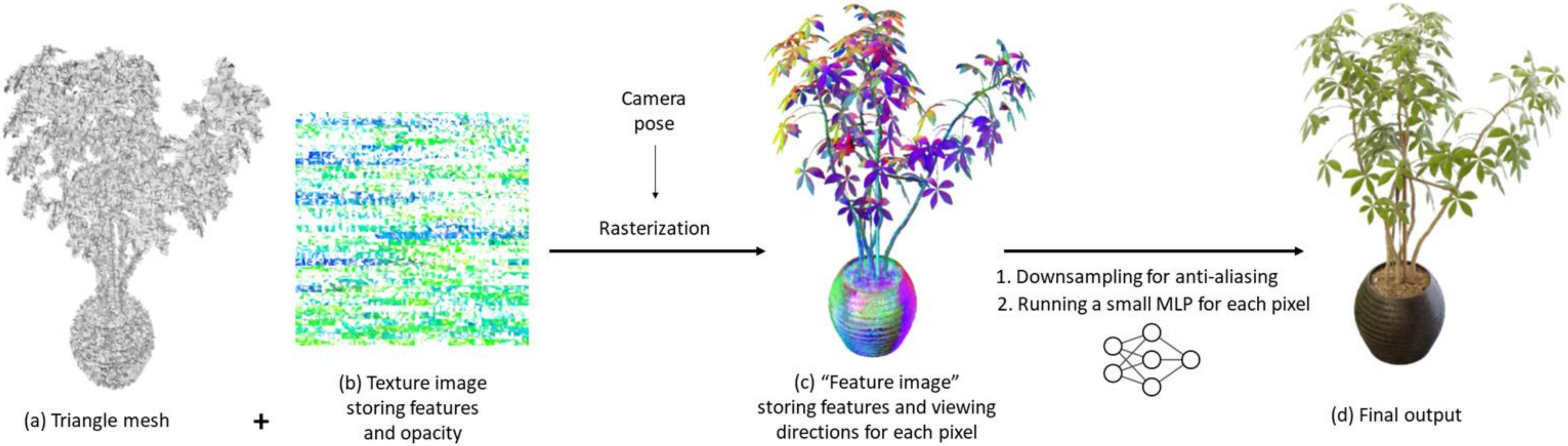
Our rendered output



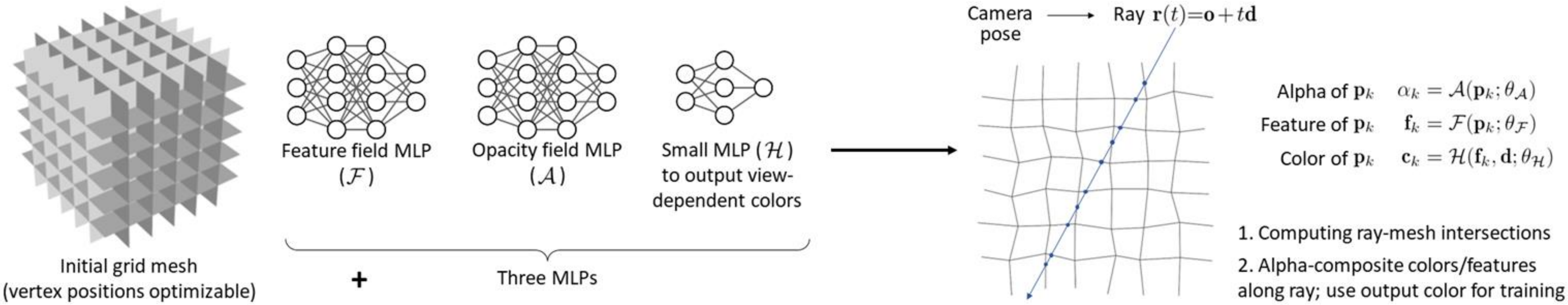
View dependent effects

> store 8-d features instead of 3-d RGB colors in the texture image.

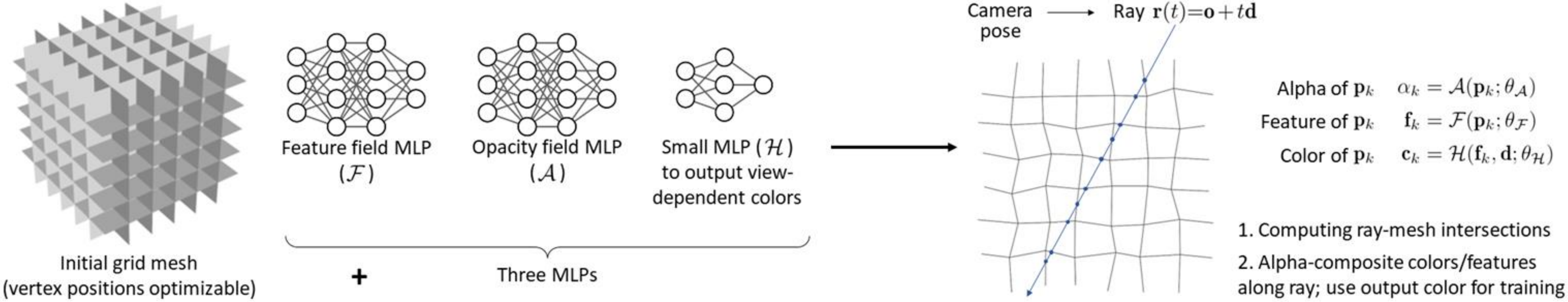
> use a tiny MLP running in a GLSL fragment shader to produce the output color.



Training - stage 1



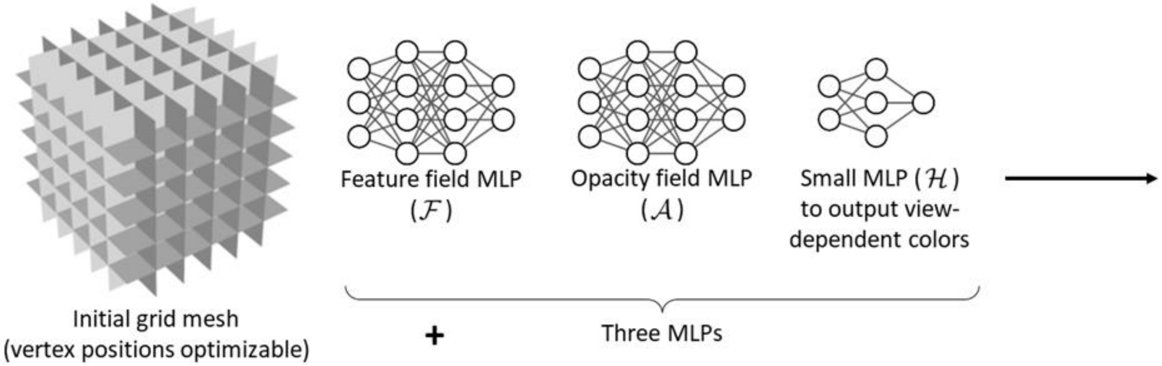
Training - stage 1

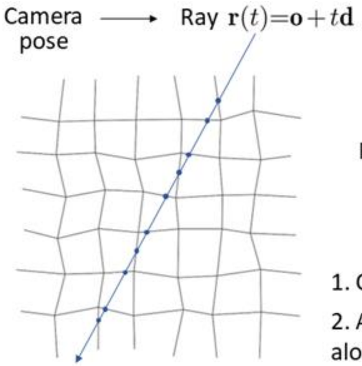


$$\mathcal{L}_\mathrm{C} = \mathbb{E}_\mathbf{r} \|\mathbf{C}(\mathbf{r}) - \mathbf{C}_\mathrm{gt}(\mathbf{r})\|_2^2.$$

$$\mathbf{C}(\mathbf{r}) = \sum_{k=1}^{K} T_k \alpha_k \mathbf{c}_k, \quad T_k = \prod_{l=1}^{k-1} (1 - \alpha_l)$$

Training - stage 1



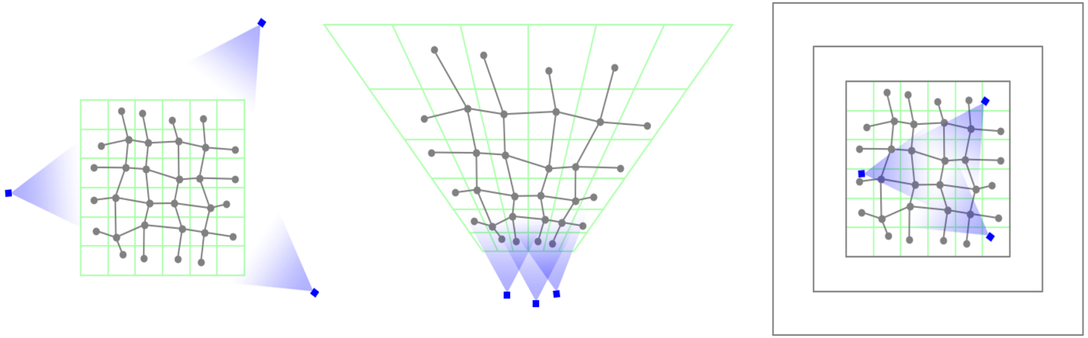


Alpha of $\mathbf{p}_k$ $\quad \alpha_k = \mathcal{A}(\mathbf{p}_k; \theta_{\mathcal{A}})$

Feature of $\mathbf{p}_k$ $\quad \mathbf{f}_k = \mathcal{F}(\mathbf{p}_k; \theta_{\mathcal{F}})$

Color of $\mathbf{p}_k$ $\quad \mathbf{c}_k = \mathcal{H}(\mathbf{f}_k, \mathbf{d}; \theta_{\mathcal{H}})$

1. Computing ray-mesh intersections
2. Alpha-composite colors/features along ray; use output color for training



(a) Synthetic 360° scene (b) Forward-Facing scene (c) Unbounded 360° scene

$$\mathcal{L}_\mathrm{C} = \mathbb{E}_\mathbf{r} \| \mathbf{C}(\mathbf{r}) - \mathbf{C}_\mathrm{gt}(\mathbf{r}) \|_2^2 .$$

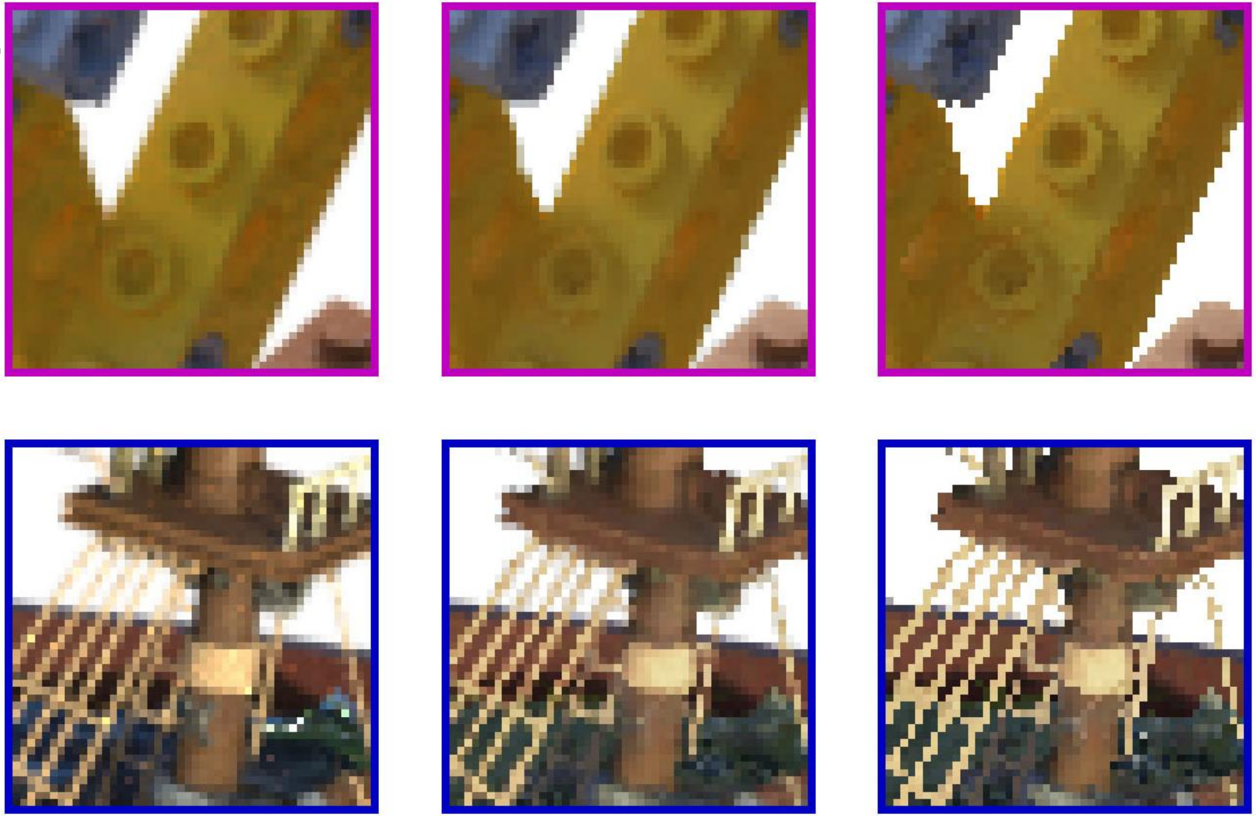
$$\mathbf{C}(\mathbf{r}) = \sum_{k=1}^{K} T_k \alpha_k \mathbf{c}_k, \quad T_k = \prod_{l=1}^{k-1} (1 - \alpha_l)$$

Training - stage 2

1. Binarization

2. Supersampling

 (for antialiasing)



Ground truth | With SS | Without SS

Training - stage 3

Extract the mesh

> store visible triangles in OBJ files.

Bake textures

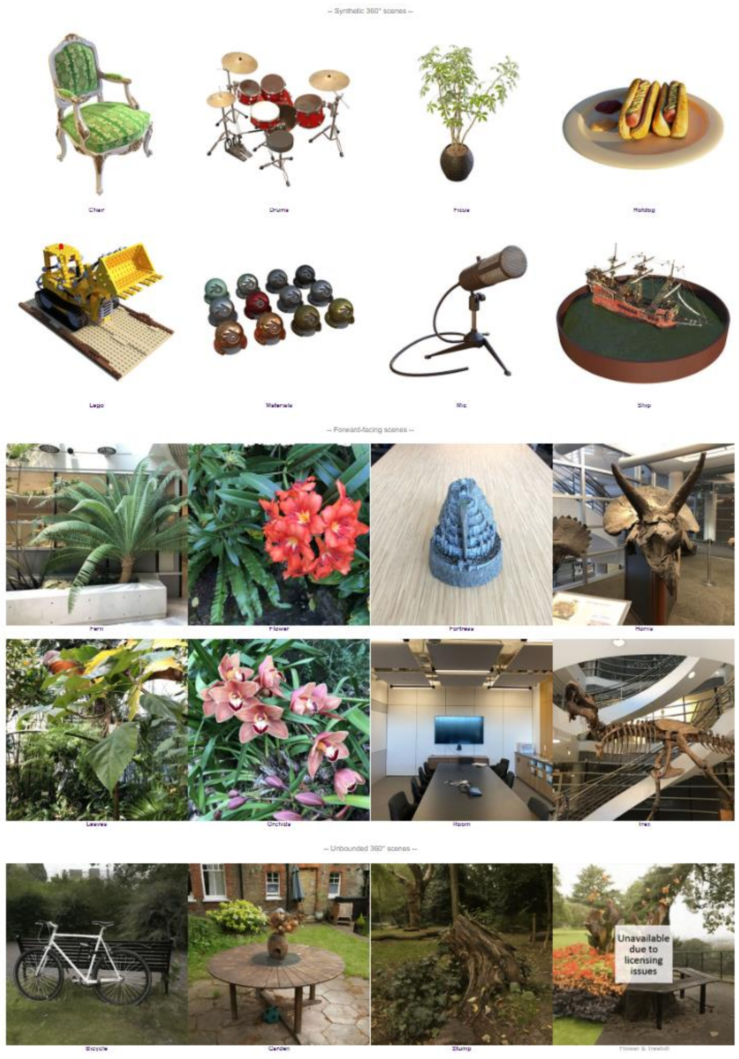
> store the features and alpha into PNG texture images.

Cache the neural renderer

> store the MLP weights into a JSON file

Online demo

https://mobile-nerf.github.io



Results

Device	Type	OS	GPU	Power
iPhone XS	Phone	iOS 15	Integrated GPU	6W
Pixel 3	Phone	Android 12	Integrated GPU	9W
Surface Pro 6	Tablet	Windows 10	Integrated GPU	15W
Chromebook	Laptop	Chrome OS	Integrated GPU	15W
Gaming laptop	Laptop	Windows 11	NVIDIA RTX 2070	115W
Desktop	PC	Ubuntu 16.04	NVIDIA RTX 2080 Ti	250W

Table 1. Devices used in our experiments. The power is the max GPU power for discrete NVIDIA cards, and the combined max CPU and GPU power for integrated GPUs.

Dataset	Synthetic 360°		Forward-facing		Unbounded 360°
Method	Ours	SNeRG	Ours	SNeRG	Ours
GPU memory	**538.38**	2707.25	**759.25**	4312.13	1162.20
Disk storage	125.75	**86.75**	**201.50**	337.25	344.60

Table 3. GPU memory and disk storage in MB.

Dataset	Synthetic 360°		Forward-facing		Unbounded 360°
Method	Ours	SNeRG	Ours	SNeRG	Ours
iPhone XS	**55.89**	$0.0\frac{8}{8}$	**27.19**$\frac{2}{8}$	$0.0\frac{8}{8}$	$22.20\frac{4}{5}$
Pixel 3	**37.14**	$0.0\frac{8}{8}$	**12.40**	$0.0\frac{8}{8}$	9.24
Surface Pro 6	**77.40**	Unsupported	**21.51**	Unsupported	19.44
Chromebook	**53.67**	$22.62\frac{2}{8}$	**19.44**	$7.85\frac{3}{8}$	15.28
Gaming laptop	**178.26**	$8.30\frac{1}{8}$	**57.72**	3.63	55.32
Gaming laptop 🔌	**606.73**	$43.87\frac{1}{8}$	**250.17**	26.01	192.59
Desktop 🔌	**744.91**	207.26	**349.34**	50.71	279.70

Table 2. The rendering speed on various devices in Frames Per Second (FPS). The devices are on battery, except for the gaming laptop and the desktop which are plugged in, indicated with a 🔌. The mobile devices (first four rows) have almost identical rendering speed when plugged in. $\frac{M}{N}$ means that M out of N testing scenes failed to run due to out-of-memory error.

Results

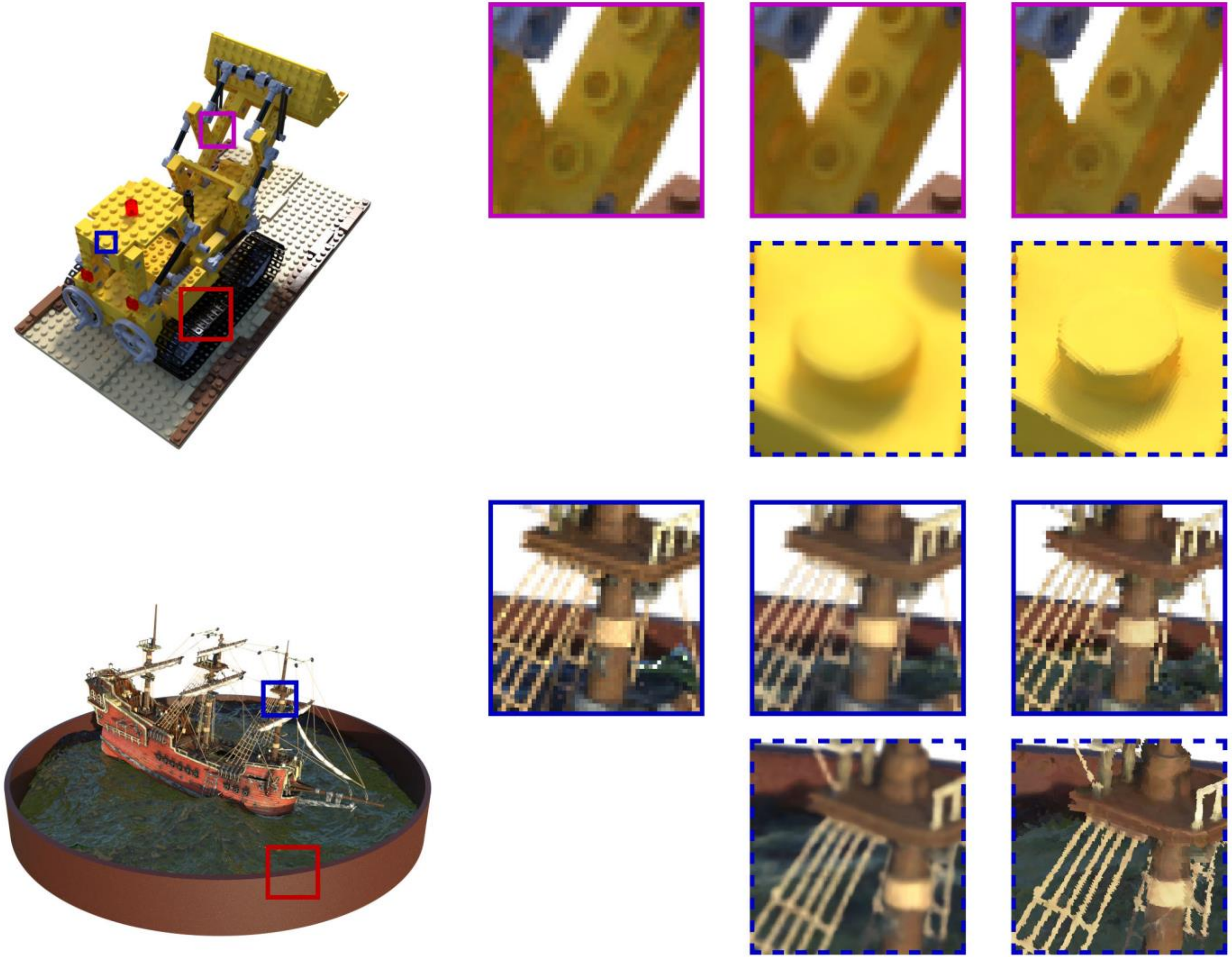
	Synthetic 360°			Forward-facing		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
NeRF [26]	31.00	0.947	0.081	26.50	0.811	0.250
JAXNeRF [12]	31.65	0.952	0.051	26.92	0.831	0.173
SNeRG [17]	30.38	**0.950**	**0.050**	25.63	0.818	**0.183**
Ours	**30.90**	0.947	0.062	**25.91**	**0.825**	**0.183**

Table 4. Quantitative results on synthetic and forward-facing scenes.

	Unbounded 360°		
	PSNR↑	SSIM↑	LPIPS↓
NeRF [26]	21.46	0.458	0.515
NeRF++ [43]	22.76	0.548	0.427
Ours	21.95	0.470	0.470

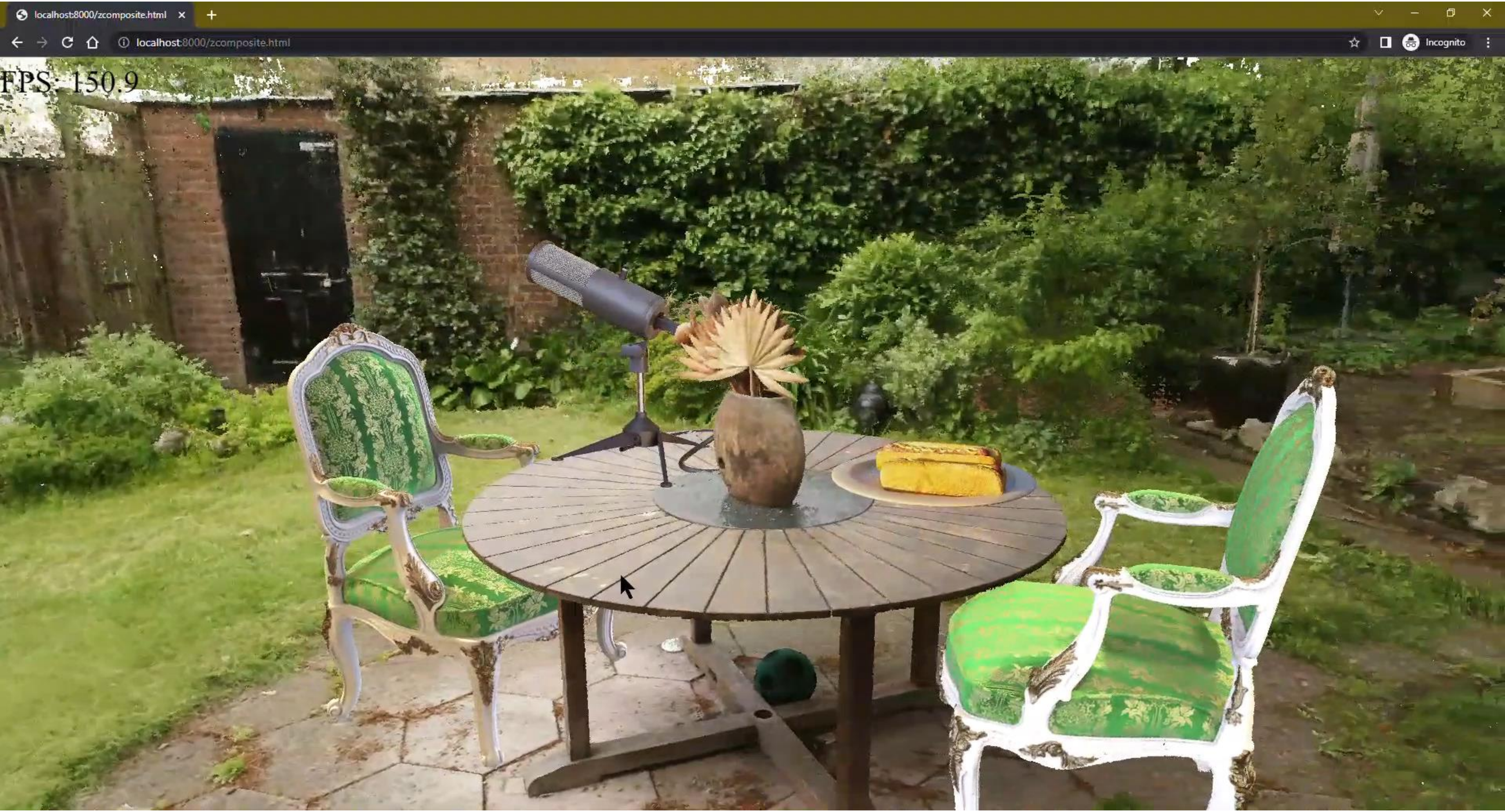
Table 5. Quantitative results on unbounded 360° scenes.

Visual results



(a) Ground truth (b) SNeRG (c) **Our method**

Scene editing


localhost:8000/zcomposite.html
localhost:8000/zcomposite.html
Incognito
FPS: 150.9

Limitations

Scene: room, forward-facing — (a) Wrong geometry

Scene: drums, synthetic 360° — (b) No semi-transparency

Scene: flower, unbounded 360° — (c) Fixed mesh resolution

Acknowledgements

Peter Hedman

Google Research

Andrea Tagliasacchi

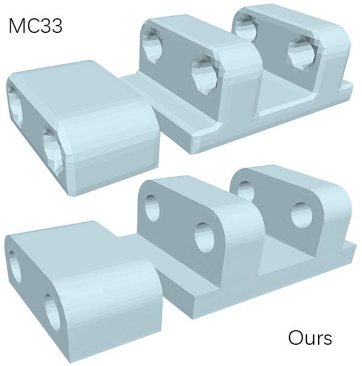
Simon Fraser University
Google Research

Thomas Funkhouser

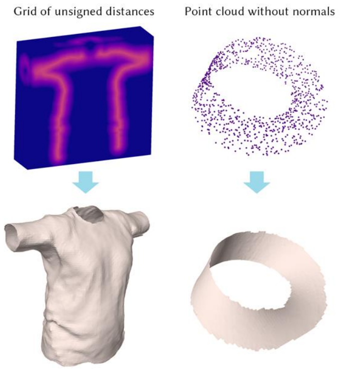
Google Research

Hao (Richard) Zhang

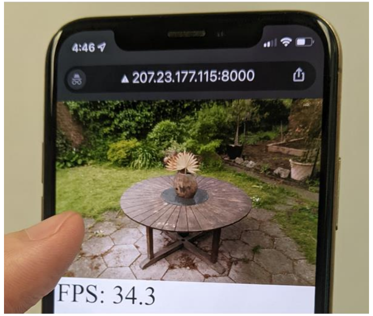
Simon Fraser University
Amazon



Neural Marching Cubes
(SIGGRAPH Asia 2021)



Neural Dual Contouring
(SIGGRAPH 2022)



MobileNeRF
(Arxiv 2022)

Code:
[NMC] https://github.com/czq142857/NMC
[NDC] https://github.com/czq142857/NDC
[MobileNeRF] https://github.com/google-research/jax3d/tree/main/jax3d/projects/mobilenerf

Thank you!