
Supplementary Material for High Resolution UDF Meshing via Iterative Networks

Federico Stella¹, Nicolas Talabot¹, Hieu Le², Pascal Fua¹

¹CVLab, EPFL ²UNC Charlotte

¹{firstname.lastname}@epfl.ch, ²hle40@charlotte.edu

A.1 Complete quantitative results

For the sake of completeness, we report in Tabs. A.3a, A.3b, A.4a, A.4b, A.5a, A.5b, A.6a and A.6b the mean results on the datasets and methods used in the main paper, alongside their respective standard deviations. We also report the results of DC-based methods on point cloud reconstruction from CAP-L [12] and DiffUDF [4] in Tab. A.2. We observe that the trends are consistent with the median results shown in the main paper, with our method outperforming all baselines at high resolutions on complex shapes, and competing closely with existing methods at lower resolutions and on simpler shapes. The standard deviations computed on our method are also consistently lower than the baselines at high resolution, while remaining competitive at lower resolutions, showing that our method is also more robust to shape variations compared to existing baselines. As in the main tables, notice that UNDC [3] failed to reconstruct meshes at high resolution due to the method’s VRAM requirements. DCUDF [6] failed to reconstruct some of the shapes in the experiments, producing unbound metrics. We discard such shapes from the reported results of DCUDF, which means that its numbers are not directly comparable to the other baselines.

A.2 Implementation details and other ablation studies

Table A.1: **Number of additional training iterations.** Median Image Consistency (IC \uparrow) of the *last* iteration at resolution 512. *Resolution is halved.

Max training iter.	MGN*	Cars	Chairs	Planes
1 iter.	94.9	88.5	86.2	87.0
3 iter.	94.9	88.4	86.0	86.8
5 iter.	94.8	88.9	87.2	87.1
7 iter.	94.9	88.5	85.9	86.5

Our network architecture consists of 2 fully connected hidden layers, with 1024 nodes each, and an output layer with 128 outputs. The input layer accepts UDF values and gradients at the 8 cell corners, making up 32 inputs. Additionally, 128 inputs per cell are needed to enable multiple iterations. We consider the current cell and the 6 cells that share a face with it, for a total of $7 * 128$ additional inputs, which brings the total number of input nodes to 928 and the total number of trainable weights to around 2.1M. Each layer, except for the final one, is followed by a leaky ReLU activation function with a negative slope of 0.01. The output layer is followed by a sigmoid activation before being used as input for the next iteration, and by a softmax function for the cross entropy loss. The network is trained using the Adam optimizer [7] with a learning rate of 5×10^{-4} for 50 epochs.

The learning rate is lower than in [10], with more epochs. We have found this helps our iterative process converge better. For training, we use the first 80 watertight shapes from ABC [8] sampled at resolution 128^3 , yielding around 5.5M training cells. As mentioned in the method section of the main

Table A.2: **Neural Unsigned Distance Fields from point clouds (DC-based methods).** L2 Mesh Chamfer Distance $\times 10^{-5}$ with 2M sample points (CD), F1 score (F1) and Image Consistency (IC) are reported at varying grid resolutions. Median scores are reported for cars; mean for scenes due to the low number of samples. The best results are in bold. UNDC failed at resolution 512 due to its large GPU memory requirements.

Res.	Method	CAP-L scenes [12]			CAP-L cars [12]			DiffUDF cars [4]		
		CD ↓	F1 ↑	IC ↑	CD ↓	F1 ↑	IC ↑	CD ↓	F1 ↑	IC ↑
128	UNDC [3]	3.34	69.7	84.4	7.85	49.4	85.5	4.46	59.1	87.6
	DualMesh-UDF [11]	53.5	63.4	69.5	9.44	50.4	85.6	6.03	62.3	84.9
	NSD-UDF + DualMesh-UDF [10]	33.6	69.8	84.5	5.49	52.4	87.2	3.07	68.2	88.2
	Ours + DualMesh-UDF	33.6	70.0	84.4	5.35	52.8	87.3	3.68	65.3	88.1
256	UNDC [3]	7.85	69.1	76.9	7.77	48.5	84.2	6.84	56.1	80.9
	DualMesh-UDF [11]	50.4	63.0	68.3	8.51	49.2	84.9	5.01	61.9	83.1
	NSD-UDF + DualMesh-UDF [10]	34.3	68.9	83.4	6.74	49.7	86.5	2.72	72.3	88.3
	Ours + DualMesh-UDF	220	49.9	79.6	6.35	50.6	86.9	2.89	70.0	88.5
512	UNDC [3]	-	-	-	-	-	-	-	-	-
	DualMesh-UDF [11]	51.7	59.8	61.8	9.26	46.3	83.5	5.07	59.4	79.4
	NSD-UDF + DualMesh-UDF [10]	35.2	66.5	77.9	7.69	47.0	85.7	3.35	69.6	85.6
	Ours + DualMesh-UDF	34.0	68.6	82.8	7.81	48.2	86.5	2.6	73.3	88.4

paper, we limit the number of additional training iterations to 5. In Tab. A.1, we provide an ablation study showing that training the network with a single additional pass already achieves good results, with 5 iterations achieving the best. Using even more iterations did not bring measurable benefits. The training takes around 2 hours on an NVIDIA A100-40G GPU.

As an ablation, we also trained the network without the noise augmentation described in the method section of the main paper. The method presented in this work achieved a median Mesh Chamfer Distance $\times 10^{-5}$ of 5.64, 5.23 and 8.84 at resolutions 128^3 , 256^3 and 512^3 respectively, using the UDF auto-decoder trained on ShapeNet [2] cars in Section 4.3 of the main paper. Without noise augmentation, the same experiment achieved 8.01, 12.5 and 47.8 respectively, showing that the noise augmentation is crucial to achieve good performance in practical scenarios.

A.3 Training convergence

In the method section of the main paper we state that our pipeline applies a sigmoid function to the network outputs before using them as input for the next iteration. We have experimentally found that, when using an identity activation (i.e. direct input) and training with only one iteration, the network goes from an IC of 87.0 on ShapeNet [2] cars at resolution 512 to 88.6, however it starts diverging after that. Using a softmax activation, which helps normalizing the otherwise unbound network outputs, the training converges, but the network does not produce significant improvements over iterations, going from an IC of 87.3 to 87.5 after two iterations. Using a sigmoid activation, instead, showed a more steady improvement over iterations, going from 87.6 to 88.5 after one iteration, and 88.9 after an additional one. However, training it with more than one iteration did not show significant improvements. Using a random number of training iterations, instead, helped the network to converge until 5 iterations, achieving similar IC scores but better CD scores (9.90×10^{-5} vs 8.84×10^{-5}), as shown in the main paper, signifying an overall similar accuracy but better surface retrieval capabilities at high resolutions.

A.4 Auto-decoder training

For our auto-decoder experiments we used the traditional auto-decoder architecture proposed in DeepSDF [9]. The input meshes are rescaled and centered within a $[-1, 1]^3$ volume, and during the data preparation phase, training points are sampled. For each mesh, 20k points are uniformly sampled within the volume, while 400k points are sampled near the surface. To obtain the surface points, 200k points are first uniformly distributed on the surface, and then small amounts of Gaussian noise are added. Gaussian noise with a mean of 0 and a standard deviation of $\sqrt{0.005}$ is applied to the first 200k surface points, and noise with a mean of 0 and a standard deviation of $\sqrt{0.0005}$ is added to the remaining 200k points. The auto-decoder network consists of 12 layers, each with 1024 hidden nodes and ReLU activations, and latent codes of size 512. It is trained using L1 loss, without regularization or Fourier encoding, for 10k epochs with a batch size of 16. To focus the network’s capacity on the

Table A.3: **Triangulating auto-decoder-based Neural Unsigned Fields using Marching Cubes-based method.** L2 Mesh Chamfer Distance $\times 10^{-5}$ with 2M sample points (CD), F1 score (F1) and Image Consistency (IC) are reported at varying grid resolutions. The best results are in bold. DCUDF failed to mesh some of the shapes, but its numbers are reported nonetheless. *Resolution is halved for experiments with MGN due to the lower complexity of the shapes.

(a) Mean results.

Res.	Method	MGN* [1]			ShapeNet cars [2]			ShapeNet chairs [2]			ShapeNet planes [2]		
		CD ↓	F1 ↑	IC ↑	CD ↓	F1 ↑	IC ↑	CD ↓	F1 ↑	IC ↑	CD ↓	F1 ↑	IC ↑
128-MC	CAP-UDF [12]	19.7	68.2	78.6	71.7	49.5	81.8	518	51.3	68.9	19.5	68.1	79.0
	MeshUDF [5]	2.79	82.0	93.3	11.9	57.9	88.7	21.8	68.0	88.7	19.4	73.5	81.0
	DCUDF [6]	14100	3.40	3.84	8000	10.6	13.7	22700	14.4	16.8	2900	28.9	24.3
	DCUDF-T [6]	116	3.04	86.6	66.5	55.4	86.8	2720	65.2	78.2	902	69.4	78.8
	DCUDF-T-nocut [6]	-	-	-	14.4	60.9	89.2	-	-	-	-	-	-
	NSD-UDF + MC [10]	1.55	82.9	94.1	9.40	59.9	88.5	17.7	68.7	88.5	4.72	78.0	84.5
	Ours + MC	2.26	80.8	93.4	6.79	59.5	88.9	7.27	67.8	89.5	3.49	77.6	84.8
256-MC	CAP-UDF [12]	3.23	85.4	91.2	48.3	58.8	86.2	223	65.5	78.5	8.95	83.2	85.0
	MeshUDF [5]	1.16	88.2	94.5	17.1	61.1	88.2	67.1	69.8	85.4	4.83	85.2	85.1
	DCUDF [6]	18200	5.17	4.12	1090	50.6	75.1	7900	50.7	55.3	278	82.4	80.2
	DCUDF-T [6]	10.3	86.1	94.8	1080	50.3	75.0	7840	50.7	55.3	797	80.1	77.9
	DCUDF-T-nocut [6]	-	-	-	68.4	56.4	80.7	-	-	-	-	-	-
	NSD-UDF + MC [10]	0.973	88.7	94.8	14.7	61.2	87.6	53.7	69.8	85.4	3.81	86.9	85.2
	Ours + MC	1.02	87.6	94.4	7.26	63.1	89.0	10.6	70.6	89.3	2.64	88.2	86.6
512-MC	CAP-UDF [12]	2.14	89.1	94.1	48.6	60.2	86.4	202	67.9	79.3	8.94	87.3	85.5
	MeshUDF [5]	1.18	89.3	94.4	136	54.4	78.8	799	57.5	64.2	21.2	87.4	83.4
	DCUDF [6]	49.4	85.8	88.8	478	52.9	80.2	7510	55.7	63.8	199	84.1	81.0
	DCUDF-T [6]	32.8	85.9	88.9	478	52.9	80.2	7380	55.9	64.2	284	83.2	79.9
	DCUDF-T-nocut [6]	-	-	-	61.4	58.5	84.7	-	-	-	-	-	-
	NSD-UDF + MC [10]	1.08	89.5	94.4	80.8	56.8	82.0	394	62.9	71.2	13.4	88.2	83.8
	Ours + MC	0.880	89.3	94.4	12.3	63.4	88.2	36.6	71.6	86.5	3.31	90.1	86.5

(b) Standard deviation for each metric, computed across the dataset.

Res.	Method	MGN* [1]			ShapeNet cars [2]			ShapeNet chairs [2]			ShapeNet planes [2]		
		CD	F1	IC	CD	F1	IC	CD	F1	IC	CD	F1	IC
128-MC	CAP-UDF [12]	12.1	7.83	3.46	51.9	12.2	5.69	649	15.4	14.7	21.3	5.63	3.45
	MeshUDF [5]	1.28	7.9	2.34	5.21	12.1	2.22	30.3	13.9	5.58	27.4	5.54	2.56
	DCUDF [6]	561	0.901	1.08	5560	4.81	5.74	18800	8.78	16.0	3010	7.84	7.45
	DCUDF-T [6]	169	1.30	3.97	57.6	11.4	4.28	5020	14.0	17.6	1400	6.72	6.79
	DCUDF-T-nocut [6]	-	-	-	9.59	11.1	2.49	-	-	-	-	-	-
	NSD-UDF + MC [10]	0.817	7.87	1.91	5.24	12.2	2.33	28.0	13.7	5.64	2.87	6.10	2.28
	Ours + MC	0.872	7.75	2.09	3.16	11.7	1.92	8.55	13.4	4.93	1.49	6.02	1.89
256-MC	CAP-UDF [12]	4.83	7.23	2.37	40.8	12.6	4.46	220	15.9	12.7	7.44	5.32	3.03
	MeshUDF [5]	0.733	6.78	1.56	11.7	11.6	2.64	74.9	14.3	8.51	4.18	5.51	2.41
	DCUDF [6]	12700	2.22	1.86	3250	14.0	15.2	9500	17.7	19.9	664	8.45	6.52
	DCUDF-T [6]	12.5	7.00	1.68	3250	14.0	15.1	9440	17.6	19.9	1440	8.70	7.70
	DCUDF-T-nocut [6]	-	-	-	51.9	12.3	7.02	-	-	-	-	-	-
	NSD-UDF + MC [10]	0.636	6.78	1.45	10.3	11.8	2.79	68.9	14.4	8.36	3.04	5.80	3.13
	Ours + MC	0.557	6.55	1.46	3.91	10.8	1.95	13.0	14.0	5.12	2.37	4.37	2.06
512-MC	CAP-UDF [12]	3.97	6.78	1.80	40.6	12.4	4.19	217	13.8	12.2	7.65	4.03	3.20
	MeshUDF [5]	1.30	6.63	1.42	115	12.9	8.34	964	16.3	17.8	20.0	5.30	4.67
	DCUDF [6]	362	10.2	7.60	989	14.7	12.8	9890	19.0	22.0	537	8.35	4.99
	DCUDF-T [6]	219	9.98	7.33	989	14.7	12.8	9950	19.0	22.5	774	8.53	5.93
	DCUDF-T-nocut [6]	-	-	-	47.4	12.4	5.51	-	-	-	-	-	-
	NSD-UDF + MC [10]	1.09	6.66	1.48	65.7	12.8	6.40	442	15.1	15.8	11.3	5.08	4.38
	Ours + MC	0.563	6.49	1.27	8.41	11.2	2.48	48.5	12.7	7.55	2.83	4.08	2.73

surface, the UDF is clamped to 0.1. The Adam optimizer [7] is used with learning rates of 0.0005 for the model and 0.001 for the latent codes, with learning rate decay applied at epochs 1600 and 3500 by a factor of 0.35.

A.5 Different auto-decoder UDF architecture

To test the robustness of our method to different UDF architectures, we trained an auto-decoder with a different architecture: using a softplus activation function and Eikonal loss with a weight of

Table A.4: **Triangulating auto-decoder-based Neural Unsigned Fields using Dual Contouring-based methods.** L2 Mesh Chamfer Distance $\times 10^{-5}$ with 2M sample points (CD), F1 score (F1) and Image Consistency (IC) are reported at varying grid resolutions. The best results are in bold. UNDC failed at resolution 512 due to high GPU memory requirements. *Resolution is halved for experiments with MGN due to the lower complexity of the shapes.

(a) Mean results.

Res.	Method	MGN* [1]			ShapeNet cars [2]			ShapeNet chairs [2]			ShapeNet planes [2]		
		CD ↓	F1 ↑	IC ↑	CD ↓	F1 ↑	IC ↑	CD ↓	F1 ↑	IC ↑	CD ↓	F1 ↑	IC ↑
128-DC	UNDC [3]	1.26	86.1	93.8	17.1	60.8	86.0	91.3	66.8	78.8	3.82	81.8	85.7
	DualMesh-UDF [11]	1120	61.1	60.9	1600	33.0	42.8	12400	15.0	13.0	341	72.4	72.3
	DualMesh-UDF-T [11]	0.939	88.7	94.9	7.45	62.6	89.4	19.1	71.4	88.7	2.91	83.3	87.3
	NSD-UDF + DualMesh-UDF [10]	0.901	89.2	94.5	8.60	63.9	89.1	17.3	70.2	88.3	3.12	85.9	86.5
	Ours + DualMesh-UDF	0.904	89.2	94.5	5.89	64.2	89.7	7.28	70.1	89.8	2.21	86.7	88.0
256-DC	UNDC [3]	1.32	87.9	91.2	122	49.2	69.0	598	50.9	55.9	21.4	81.9	78.7
	DualMesh-UDF [11]	993	60.0	59.0	1440	33.3	42.6	11900	15.2	13.0	158	76.1	72.5
	DualMesh-UDF-T [11]	0.899	89.7	94.7	14.3	61.8	86.6	62.2	69.0	82.4	3.26	87.8	86.8
	NSD-UDF + DualMesh-UDF [10]	0.838	89.5	94.3	15.6	61.8	87.0	56.0	68.3	82.4	3.72	88.3	85.4
	Ours + DualMesh-UDF	0.806	89.5	94.4	7.57	63.5	88.5	12.5	69.2	86.7	2.66	89.3	87.3
512-DC	UNDC [3]	6.86	83.0	81.2	-	-	-	-	-	-	-	-	-
	DualMesh-UDF [11]	948	58.4	57.1	1560	32.2	41.1	12400	14.7	12.2	172	75.6	70.6
	DualMesh-UDF-T [11]	1.25	88.8	92.7	51.7	55.4	77.9	211	60.6	67.2	7.68	87.7	83.7
	NSD-UDF + DualMesh-UDF [10]	1.13	88.2	91.9	86.3	54.3	77.8	412	58.4	63.9	13.8	86.7	82.1
	Ours + DualMesh-UDF	0.899	88.2	92.3	13.5	61.0	85.1	38.9	67.1	80.1	3.47	89.0	85.2

(b) Standard deviation for each metric, computed across the dataset.

Res.	Method	MGN* [1]			ShapeNet cars [2]			ShapeNet chairs [2]			ShapeNet planes [2]		
		CD	F1	IC	CD	F1	IC	CD	F1	IC	CD	F1	IC
128-DC	UNDC [3]	0.651	6.80	1.43	11.6	12.0	3.26	94.4	15.5	11.2	3.33	5.77	2.69
	DualMesh-UDF [11]	2470	25.6	25.4	2090	12.0	13.2	12800	12.0	11.3	995	10.8	12.2
	DualMesh-UDF-T [11]	0.561	6.51	1.36	4.17	10.5	2.10	29.3	13.5	5.94	2.58	5.34	1.94
	NSD-UDF + DualMesh-UDF [10]	0.546	6.48	1.50	5.14	11.4	2.21	27.2	13.3	5.55	2.76	4.96	2.33
	Ours + DualMesh-UDF	0.517	6.47	1.43	2.92	11.0	1.87	9.04	13.2	4.48	1.40	4.88	1.95
256-DC	UNDC [3]	1.25	6.80	2.41	97.7	11.9	9.59	716	17.6	16.1	21.1	6.89	5.63
	DualMesh-UDF [11]	2280	25.9	25.5	1390	11.6	12.8	12900	11.9	10.9	241	9.85	11.8
	DualMesh-UDF-T [11]	0.673	6.49	1.31	9.94	10.7	3.12	72.3	13.6	8.97	2.96	3.96	2.39
	NSD-UDF + DualMesh-UDF [10]	0.565	6.80	1.37	11.1	11.8	3.10	71.2	13.1	8.33	3.07	4.44	3.16
	Ours + DualMesh-UDF	0.516	6.81	1.26	4.21	11.1	2.22	15.9	13.0	4.98	2.57	4.08	2.30
512-DC	UNDC [3]	11.3	9.36	6.95	-	-	-	-	-	-	-	-	-
	DualMesh-UDF [11]	2080	26.0	25.6	1960	11.4	12.5	14500	11.6	10.3	267	9.88	11.7
	DualMesh-UDF-T [11]	1.36	7.14	2.51	38.5	11.5	5.96	221	13.6	14.3	6.87	4.94	3.93
	NSD-UDF + DualMesh-UDF [10]	1.18	7.48	2.92	69.7	12.6	7.41	467	14.9	15.5	11.4	5.2	4.83
	Ours + DualMesh-UDF	0.616	7.48	2.54	9.32	11.4	3.03	49.6	12.3	7.63	2.83	4.36	3.10

0.1, with the rest as in the section above. We show the results on ShapeNet [2] cars in Tab. A.7, along with the highest-scoring baselines from the main experiment. While all methods achieved slightly better performance compared to the ReLU-based architecture, the same conclusions apply. Our method outperforms the tested baselines, particularly so at high resolutions, while also achieving lower standard deviations across the dataset.

A.6 Additional figures

We show here additional qualitative results of our method compared to the baselines. In Fig. A.1 we show an additional example at different resolutions compared to the NSD-UDF [10] baseline: the shapes look similar at low resolutions, but at 256 and 512 the baseline cannot retrieve large portions of the surface.

In Fig. A.2 & A.3 we show the same shapes as in Fig. 3 of the main paper, but with different resolutions. In Fig. A.4, A.5 & A.6 we show additional shapes at all tested resolutions. As observed in the main paper, existing methods retrieve most of the surface at lower resolutions, leaving less room for improvement, whereas at higher resolutions our method shows a significant advantage.

In Fig. A.7, we show the meshing on the CAP-L 3D scene [12] used in Fig. 4 of the main text in greater size to better appreciate the details, e.g., the base of the statues.



Figure A.1: **Meshing at different resolutions, additional examples.** While NSD-UDF [10] retrieves most of the surface well at a low resolution, it struggles at higher ones. In contrast, our method, recovers the surface well at all resolutions. We use Marching Cubes with both methods.

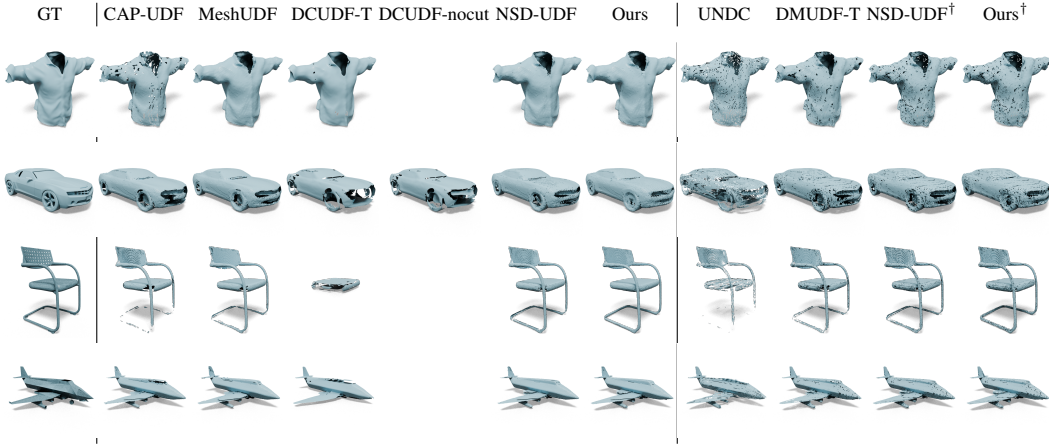


Figure A.2: **Qualitative comparison at resolution 256.** Surface meshing results of auto-decoder-based neural UDFs with all methods at resolution of 256 (and 128 for MGN). [†] indicates that the method is combined with DMUFD. The shapes are the same as in the main text.

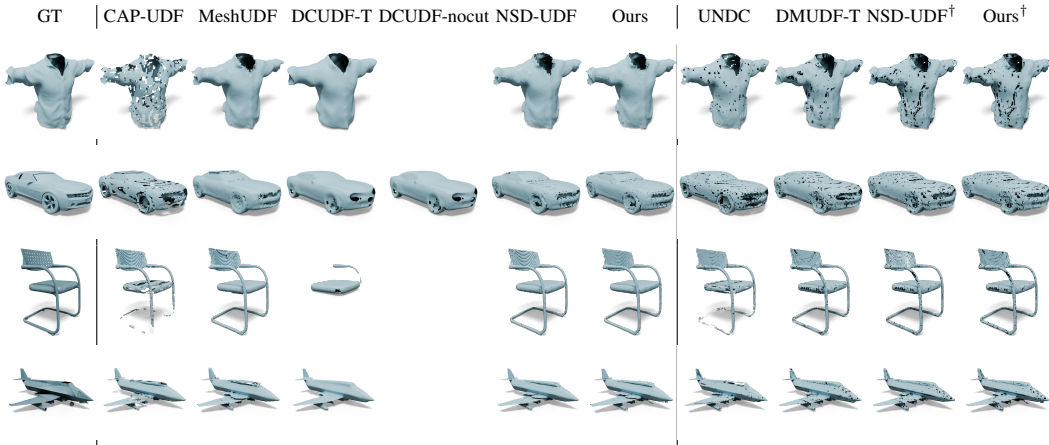


Figure A.3: **Qualitative comparison at resolution 128.** Surface meshing results of auto-decoder-based neural UDFs with all methods at resolution of 128 (and 64 for MGN). [†] indicates that the method is combined with DMUFD. The shapes are the same as in the main text.

Table A.5: **Neural Unsigned Distance Fields from point clouds (MC-based methods).** L2 Mesh Chamfer Distance $\times 10^{-5}$ with 2M sample points (CD), F1 score (F1) and Image Consistency (IC) are reported at varying grid resolutions. The best results are in bold.

(a) Mean results.

Res.	Method	CAP-L scenes [12]			CAP-L cars [12]			DiffUDF cars [4]		
		CD ↓	F1 ↑	IC ↑	CD ↓	F1 ↑	IC ↑	CD ↓	F1 ↑	IC ↑
128-MC	CAP-UDF [12]	4.27	68.0	83.8	11.4	48.5	85.3	7.96	60.1	86.2
	MeshUDF [5]	4.40	68.2	84.5	11.9	49.7	86.1	9.52	61.4	87.0
	DCUDF-T [6]	279	59.7	84.0	131	45.0	83.7	99.4	53.2	87.0
	DCUDF-T-nocut [6]	-	-	-	10.7	51.3	87.6	15.3	58.6	90.2
	NSD-UDF + MC [10]	3.34	69.8	86.7	11.1	51.7	86.3	4.86	66.0	87.8
	Ours + MC	3.44	69.3	85.3	9.76	51.8	86.8	6.65	63.6	87.5
256-MC	CAP-UDF [12]	3.65	70.1	86.1	10.1	53.4	86.4	4.04	69.0	86.6
	MeshUDF [5]	3.42	69.6	86.3	10.2	52.3	86.1	5.61	66.8	87.5
	DCUDF-T [6]	4.75	69.9	84.6	42.1	50.7	84.7	106	63.8	84.6
	DCUDF-T-nocut [6]	-	-	-	12.0	53.1	85.7	6.37	68.8	87.1
	NSD-UDF + MC [10]	3.30	70.3	86.3	11.5	52.9	86.1	3.85	70.6	86.7
	Ours + MC	3.07	71.4	86.6	9.93	54.0	86.8	4.23	69.2	88.0
512-MC	CAP-UDF [12]	3.57	70.5	86.1	10.8	53.8	86.2	4.87	69.3	84.0
	MeshUDF [5]	4.49	69.9	84.5	10.9	52.7	85.7	4.25	68.8	86.2
	DCUDF-T [6]	140	68.9	83.3	37.0	50.9	85.0	727	61.8	78.3
	DCUDF-T-nocut [6]	-	-	-	13.5	51.5	84.4	6.64	68.2	86.4
	NSD-UDF + MC [10]	3.72	70.0	84.1	12.3	52.5	85.6	5.92	67.1	81.1
	Ours + MC	3.08	71.8	86.8	10.5	54.1	86.4	3.48	71.9	87.8

(b) Standard deviation for each metric, computed across the dataset.

Res.	Method	CAP-L scenes [12]			CAP-L cars [12]			DiffUDF cars [4]		
		CD	F1	IC	CD	F1	IC	CD	F1	IC
128-MC	CAP-UDF [12]	1.98	19.9	2.43	4.71	12.2	3.07	3.88	11.2	2.53
	MeshUDF [5]	1.38	20.1	1.42	5.77	13.9	3.13	5.23	13.4	2.97
	DCUDF-T [6]	385	20.1	6.53	318	12.9	5.91	109	12.0	5.37
	DCUDF-T-nocut [6]	-	-	-	5.37	12.6	3.06	12.5	12.5	1.65
	NSD-UDF + MC [10]	1.67	20.8	2.20	6.91	14.2	3.31	3.54	12.6	2.69
	Ours + MC	1.67	20.0	1.8	6.26	13.3	2.75	5.29	12.1	2.30
256-MC	CAP-UDF [12]	1.91	21.1	1.94	6.65	14.4	3.46	2.35	12.6	3.41
	MeshUDF [5]	1.79	21.3	2.03	5.65	14.9	3.48	3.06	13.3	2.78
	DCUDF-T [6]	3.71	21.8	2.99	46.7	15.5	4.38	202	14.3	7.80
	DCUDF-T-nocut [6]	-	-	-	6.85	15.2	4.13	3.66	13.4	3.40
	NSD-UDF + MC [10]	1.76	21.5	2.63	7.27	15.1	3.69	2.47	13.0	3.56
	Ours + MC	1.60	20.3	2.04	6.63	14.4	3.11	3.31	11.9	2.24
512-MC	CAP-UDF [12]	1.98	21.3	2.44	7.26	15	3.64	3.14	14.2	5.29
	MeshUDF [5]	3.38	21.5	3.45	6.35	15.3	3.94	2.46	13.3	3.30
	DCUDF-T [6]	191	21.3	4.07	28.2	15.3	4.21	1310	15.8	12.2
	DCUDF-T-nocut [6]	-	-	-	7.45	15.3	5.10	3.78	13.5	3.57
	NSD-UDF + MC [10]	2.07	21.1	3.21	7.66	15.4	4.07	3.95	15.6	7.05
	Ours + MC	1.63	20.4	2.54	6.99	15.0	3.48	2.15	12.4	2.73

Table A.6: **Neural Unsigned Distance Fields from point clouds (DC-based methods).** L2 Mesh Chamfer Distance $\times 10^{-5}$ with 2M sample points (CD), F1 score (F1) and Image Consistency (IC) are reported at varying grid resolutions. The best results are in bold. UNDC failed at resolution 512 due to its large GPU memory requirements.

(a) Mean results.

Res.	Method	CAP-L scenes [12]			CAP-L cars [12]			DiffUDF cars [4]		
		CD ↓	F1 ↑	IC ↑	CD ↓	F1 ↑	IC ↑	CD ↓	F1 ↑	IC ↑
128-DC	UNDC [3]	3.34	69.7	84.4	12.7	52.3	84.8	4.76	60.9	87.6
	DualMesh-UDF [11]	53.5	63.4	69.5	11.3	54.0	85.0	5.77	62.5	84.3
	NSD-UDF + DualMesh-UDF [10]	33.6	69.8	84.5	7.01	56.0	87.0	3.51	68.3	88.3
	Ours + DualMesh-UDF	33.6	70.0	84.4	6.73	56.2	87.2	4.37	66.7	88.2
256-DC	UNDC [3]	7.85	69.1	76.9	9.89	52.9	83.9	17.5	58.3	81.2
	DualMesh-UDF [11]	50.4	63.0	68.3	10.8	53.2	84.3	5.43	61.1	82.5
	NSD-UDF + DualMesh-UDF [10]	34.3	68.9	83.4	8.42	54.0	86.2	3.08	71.5	87.7
	Ours + DualMesh-UDF	220	49.9	79.6	7.95	54.6	86.7	3.36	70.4	88.3
512-DC	UNDC [3]	-	-	-	-	-	-	-	-	-
	DualMesh-UDF [11]	51.7	59.8	61.8	11.4	50.6	83.1	5.69	58.2	79.0
	NSD-UDF + DualMesh-UDF [10]	35.2	66.5	77.9	10.1	51.7	85.4	3.99	68.8	85.1
	Ours + DualMesh-UDF	34.0	68.6	82.8	9.15	52.5	86.2	3.05	72.3	87.9

(b) Standard deviation for each metric, computed across the dataset.

Res.	Method	CAP-L scenes [12]			CAP-L cars [12]			DiffUDF cars [4]		
		CD	F1	IC	CD	F1	IC	CD	F1	IC
128-DC	UNDC [3]	2.55	20.2	3.97	13.4	13.0	3.27	2.22	13.1	2.14
	DualMesh-UDF [11]	35.7	18.9	7.07	6.58	13.8	4.27	2.77	13.7	3.42
	NSD-UDF + DualMesh-UDF [10]	42.8	17.6	0.739	3.98	13.1	2.88	2.04	11.7	2.14
	Ours + DualMesh-UDF	43.0	17.3	0.803	3.70	12.8	2.60	3.02	11.8	2.00
256-DC	UNDC [3]	8.26	21.8	9.61	5.38	14.4	3.88	30.5	10.3	1.65
	DualMesh-UDF [11]	35.7	19.5	7.01	6.34	14.7	4.60	2.66	13.9	3.16
	NSD-UDF + DualMesh-UDF [10]	43.3	18.6	0.403	4.82	14.3	3.51	1.96	12.5	2.82
	Ours + DualMesh-UDF	243	28.1	6.58	4.43	14.0	3.11	2.34	12.0	2.17
512-DC	UNDC [3]	-	-	-	-	-	-	-	-	-
	DualMesh-UDF [11]	35.0	19.9	8.32	6.45	14.6	4.86	2.68	13.5	3.03
	NSD-UDF + DualMesh-UDF [10]	42.3	18.6	2.04	5.74	14.7	4.04	2.50	14.0	3.80
	Ours + DualMesh-UDF	43.2	18.2	0.0709	5.05	14.6	3.53	1.89	12.4	2.86

Table A.7: **Triangulating a Softplus-based auto-decoder.** L2 Chamfer Distance $\times 10^{-5}$ with 2M sample points (CD), F1 score (F1) and Image Consistency (IC) are reported at varying grid resolutions on the ShapeNet [2] cars dataset. The best results are in bold.

Res.	Method	Median			Mean			Std		
		CD ↓	F1 ↑	IC ↑	CD ↓	F1 ↑	IC ↑	CD	F1	IC
128-MC	CAP-UDF [12]	60.9	52.5	80.7	69.2	49.9	80.1	56.6	10.5	4.51
	MeshUDF [5]	8.06	61.4	88.2	10.0	61.3	88.6	6.40	11.6	2.17
	NSD-UDF + MC [10]	5.95	64.9	88.6	7.69	63.5	88.7	5.13	11.9	2.26
	Ours + MC	4.80	64.0	89.0	5.85	63.1	89.1	3.53	11.3	1.94
256-MC	CAP-UDF [12]	23.7	65.2	86.8	30.0	63.9	86.0	23.9	12.3	3.96
	MeshUDF [5]	13.4	67.9	88.4	15.1	66.3	88.3	10.4	11.7	2.56
	NSD-UDF + MC [10]	8.63	69.3	88.3	10.4	66.7	88.0	8.94	11.9	2.57
	Ours + MC	4.95	70.4	89.6	6.28	68.2	89.3	4.69	11.3	1.89
512-MC	CAP-UDF [12]	23.8	67.3	87.2	30.3	65.8	86.4	24.6	12.6	3.97
	MeshUDF [5]	58.3	62.3	77.9	69.5	59.6	77.4	45.2	13.2	7.15
	NSD-UDF + MC [10]	21.2	68.0	85.8	24.6	65.0	84.8	19.9	12.8	4.44
	Ours + MC	8.83	71.4	88.9	10.7	68.7	88.7	9.25	11.6	2.37

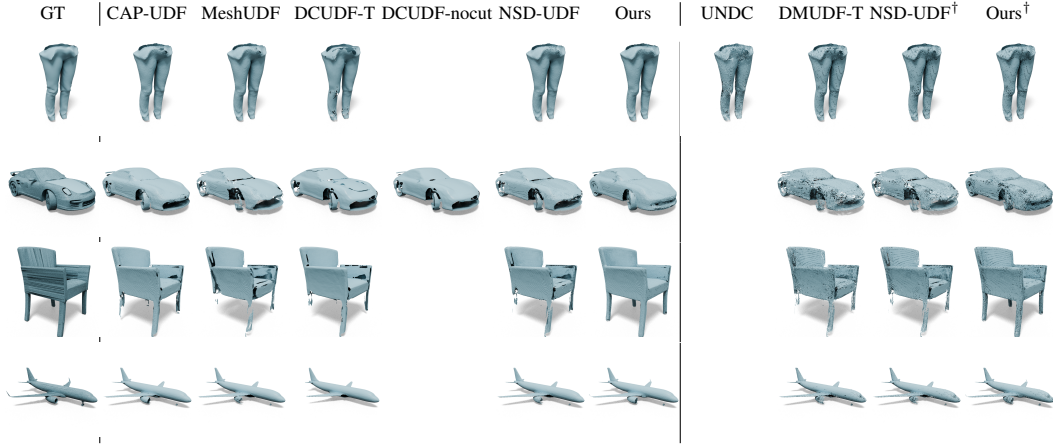


Figure A.4: **Additional qualitative comparison at resolution 512.** Surface meshing results of auto-decoder-based neural UDFs with all methods at resolution of 512 (and 256 for MGN). UNDC failed at resolution 512 due to high GPU memory requirements. [†] indicates that the method is combined with DMUFD.

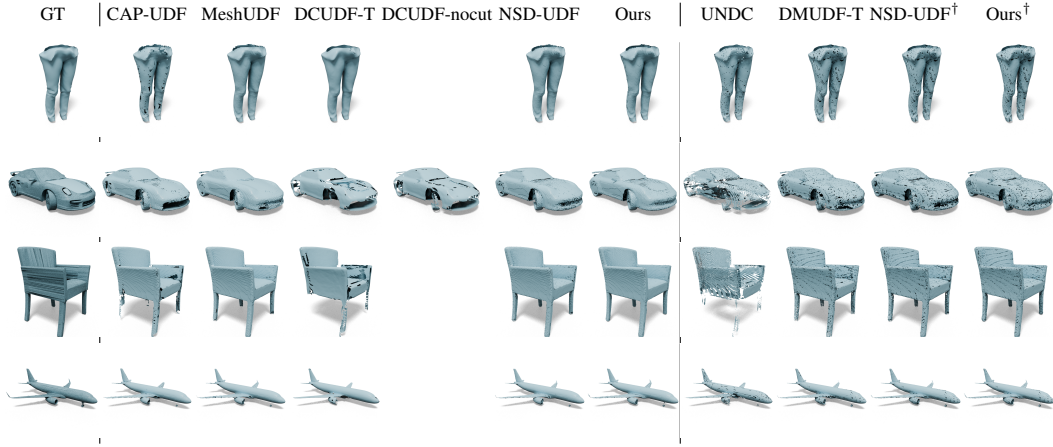


Figure A.5: **Additional qualitative comparison at resolution 256.** Surface meshing results of auto-decoder-based neural UDFs with all methods at resolution of 256 (and 128 for MGN). [†] indicates that the method is combined with DMUFD.

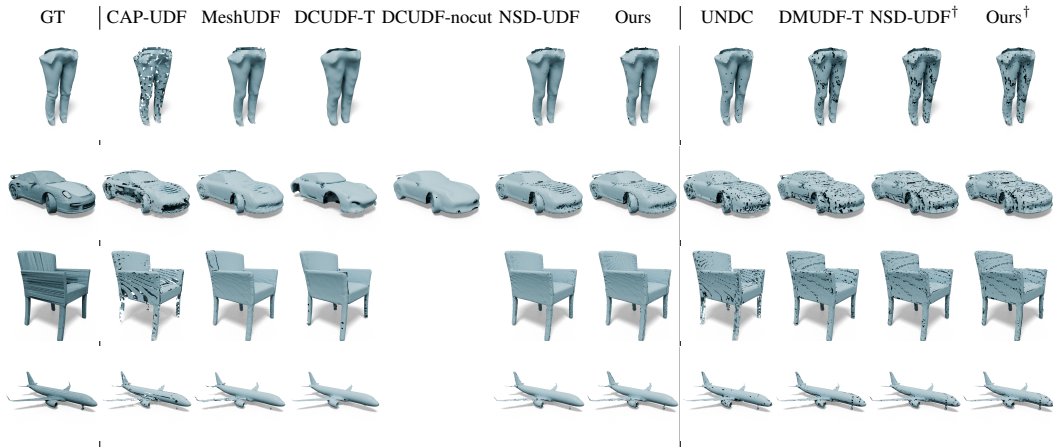


Figure A.6: **Additional qualitative comparison at resolution 128.** Surface meshing results of auto-decoder-based neural UDFs with all methods at resolution of 128 (and 64 for MGN). [†] indicates that the method is combined with DMUFD.

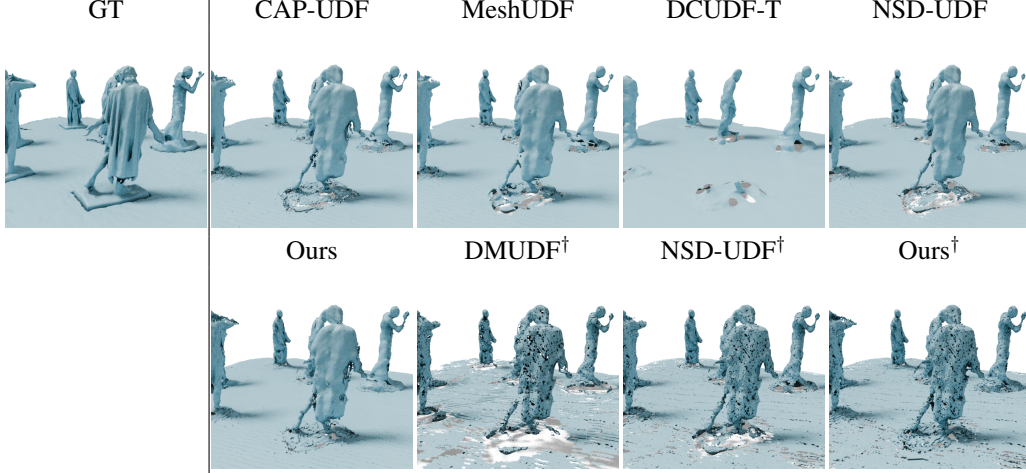


Figure A.7: **Meshing of a 3D scene from CAP-L [12]**. Surface meshing results of the "Burghers" scene with all methods at resolution of 512. UNDC failed at resolution 512 due to high GPU memory requirements. [†] indicates that the method is Dual Contouring-based, otherwise it is Marching Cubes-based.

A.7 DoubleCoverUDF [6] tuning and min-cut

To achieve the better results with DCUDF [6] compared to its default parameters, we have run the algorithm 3 times per experiment and per resolution, with different parameters, and we have selected the best run in each scenario. Additionally, in Fig. A.8, we present the meshing results of DCUDF [6] on UDFs from different models on cars, both with and without the final min-cut step, at resolutions of 128 and 512. Without the min-cut, more surfaces are retained, though they are double-layered. Moreover, the overall meshing quality deteriorates at higher resolutions, following the trend observed for the other baselines.

A.8 Meshing ground-truth UDF

Although our method is designed to handle imperfect UDFs, we also evaluate it on true UDFs computed directly from ground-truth meshes to verify that it does not introduce unwanted artifacts. We report results on the ShapeNet [2] cars dataset in Tab. A.8 (top), and visualize several triangulations at a resolution of 512 in Fig. A.9. As observed, our method does not introduce significant artifacts; however, as expected, multiple iterations provide no benefit in this setting.

In the same table, we also compute the number of holes as surface boundaries, similarly to the description provided by DCUDF [6], at resolution 512. We report the average. We notice that the original meshes have a large number of boundaries because they contain multiple detailed components and inner structures. None of the methods faithfully recovers the correct mesh topology. Methods that rely directly on MC triangulation, such as CAP-UDF [12], NSD-UDF+MC [10] and Ours+MC, tend to suffer from micro-holes and gaps between some of the faces. MeshUDF uses a heuristic specifically designed to reduce this behavior and connect as many portions of the surface as possible, explaining the lower number of holes. DCUDF-T [6] starts from an inflated mesh, so it generally tends to have fewer holes. Simple postprocessing steps can be applied to all methods to improve the final mesh quality. We take as an example the first of the ShapeNet Cars (object 100715345ee54d7ae38b52b4ee9d36a3), and we apply Trimesh-based postprocessing (fill small holes, merge close vertices, remove spurious faces), showing that all methods improve.

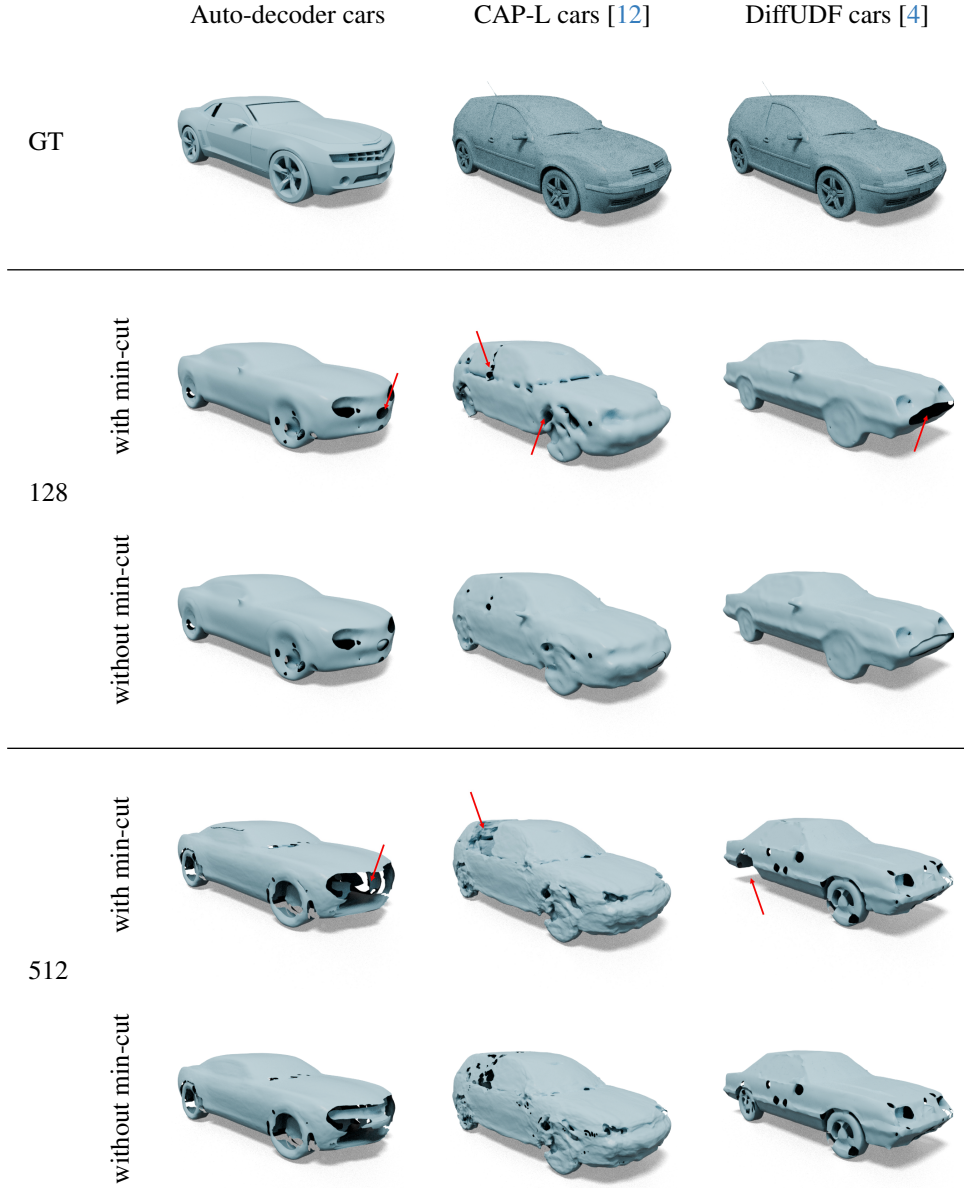


Figure A.8: **DoubleCoverUDF min-cut.** Comparing reconstructions of DCUDF [6] with and without the min-cut step at resolution 128 (middle) and 512 (bottom).

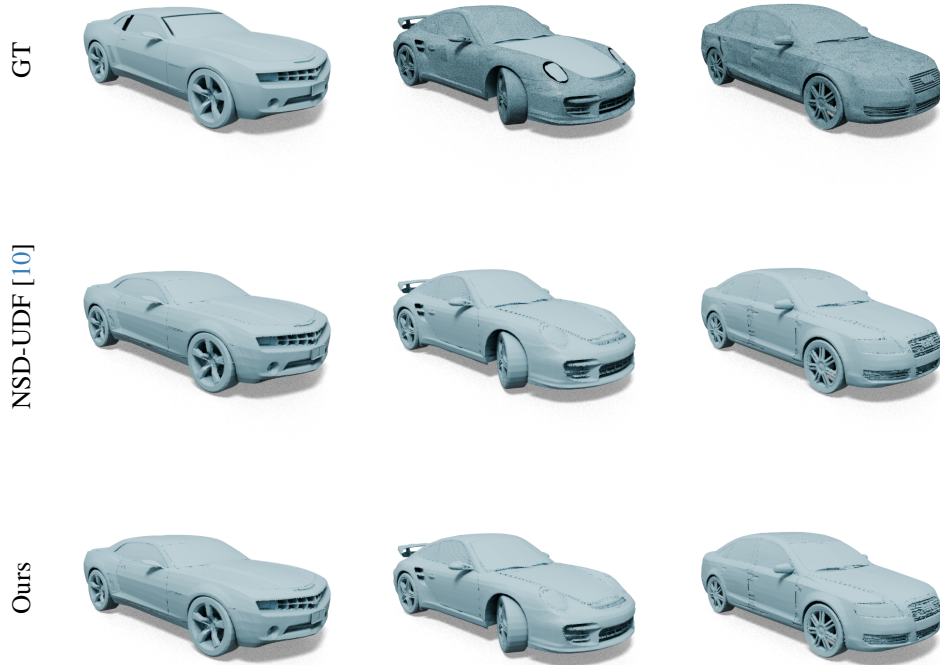


Figure A.9: **Triangulating ground-truth UDF.** Surface meshing results on the ground-truth UDF in comparison to NSD-UDF [10]. Our method does not introduce significant unwanted artifacts.

Table A.8: **Triangulating ground-truth UDFs.** Results are reported on the ShapeNet [2] cars dataset, using the true UDF computed from the mesh. We compare against NSD-UDF + MC [10] at multiple resolution with the Median L2 Chamfer Distance $\times 10^{-5}$ with 2M sample points (Chamfer-Distance), and against all MC-based methods with the average number of holes, as surface boundaries at resolution 512 (Number of holes). For the latter, we also report results on the first car of the dataset, before and after post-processing the mesh.

(a) Chamfer-Distance

Method	Res.		
	128	256	512
NSD-UDF + MC [10]	1.34	0.230	0.0300
Ours + MC at iter. 1	1.36	0.244	0.0294
Ours + MC at iter. 6	1.40	0.231	0.0300

(b) Number of holes at resolution 512

	ShapeNet cars [2]	Car 1 (before post-process.)	Car 1 (post-processed)
GT	783	1552	-
CAP-UDF [12]	10024	13265	2125
MeshUDF [5]	428	997	240
DCUFD-T [6]	7	6	6
DCUFD-T-nocut [6]	2.65	2	2
NSD-UDF+MC [10]	8280	8885	1387
Ours+MC	9284	9808	1604

References

- [1] B. L. Bhatnagar, G. Tiwari, C. Theobalt, and G. Pons-Moll. Multi-Garment Net: Learning to Dress 3D People from Images. In *International Conference on Computer Vision*, pages 5420–5430, 2019. [3](#), [4](#)
- [2] A. Chang, T. Funkhouser, L. G., P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An Information-Rich 3D Model Repository. In *arXiv Preprint*, 2015. [2](#), [3](#), [4](#), [7](#), [9](#), [11](#)
- [3] Z. Chen, A. Tagliasacchi, T. Funkhouser, and H. Zhang. Neural Dual Contouring. In *arXiv Preprint*, 2022. [1](#), [2](#), [4](#), [7](#)
- [4] M. Fainstein, V. Siless, and E. Iarussi. DUDF: Differentiable Unsigned Distance Fields with Hyperbolic Scaling. In *Conference on Computer Vision and Pattern Recognition*, 2024. [1](#), [2](#), [6](#), [7](#), [10](#)
- [5] B. Guillard, F. Stella, and P. Fua. Meshudf: Fast and Differentiable Meshing of Unsigned Distance Field Networks. In *European Conference on Computer Vision*, pages 576–592, 2022. [3](#), [6](#), [7](#), [11](#)
- [6] Fei Hou, Xuhui Chen, Wencheng Wang, Hong Qin, and Ying He. Robust Zero Level-Set Extraction from Unsigned Distance Fields Based on Double Covering. *ACM Transactions on Graphics*, 2023. [1](#), [3](#), [6](#), [9](#), [10](#), [11](#)
- [7] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimisation. In *International Conference on Learning Representations*, 2015. [1](#), [3](#)
- [8] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, and D. Panozzo. ABC: A Big CAD Model Dataset for Geometric Deep Learning. In *Conference on Computer Vision and Pattern Recognition*, pages 9601–9611, 2019. [1](#)
- [9] J. J. Park, P. Florence, J. Straub, R. A. Newcombe, and S. Lovegrove. DeepSdf: Learning Continuous Signed Distance Functions for Shape Representation. In *Conference on Computer Vision and Pattern Recognition*, 2019. [2](#)
- [10] F. Stella, N. Talabot, H. Le, and P. Fua. Neural Surface Localization for Unsigned Distance Fields. In *European Conference on Computer Vision*, 2024. [1](#), [2](#), [3](#), [4](#), [5](#), [6](#), [7](#), [9](#), [11](#)
- [11] C. Zhang, G. Lin, L. Yang, X. Li, T. Komura, S. Schaefer, J. Keyser, and W. Wang. Surface Extraction from Neural Unsigned Distance Fields. In *International Conference on Computer Vision*, pages 0000–0000, 2023. [2](#), [4](#), [7](#)
- [12] J. Zhou, B. Ma, S. Li, Y.-S. Liu, Y. Fang, and Z. Han. CAP-UDF: Learning Unsigned Distance Functions Progressively From Raw Point Clouds With Consistency-Aware Field Optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. [1](#), [2](#), [3](#), [4](#), [6](#), [7](#), [9](#), [10](#), [11](#)