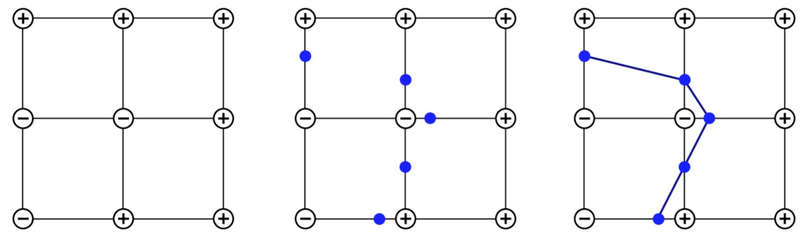
Neural Surface Detection for Unsigned Distance Fields

Federico Stella, Nicolas Talabot, Hieu Le, Pascal Fua

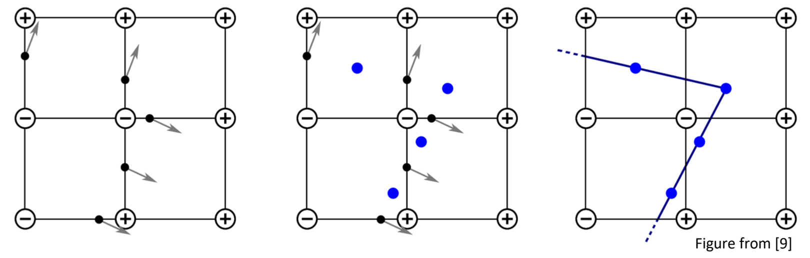
Computer Vision Laboratory, EPFL, Lausanne (Switzerland)

1. Motivation

Signed Distance [1] and Occupancy Field [2,3] Networks can represent watertight surfaces with any topology. They can be meshed with algorithms such as Marching Cubes (MC) [4] and Dual Contouring (DC) [5], which require sign flips to detect a surface.

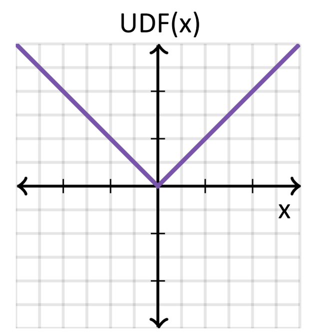


Marching Cubes [4]



Dual Contouring [5]

Unsigned Distance Fields (UDFs) have been proposed to deal with open surfaces, without preprocessing. Since they lack sign flips, they cannot be meshed using traditional methods.

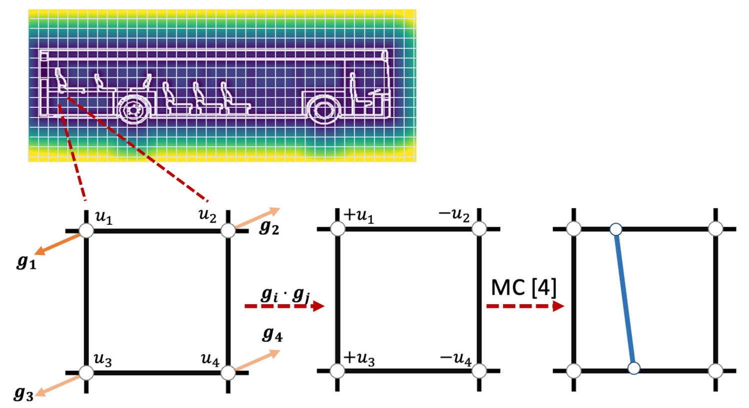


2. Existing methods

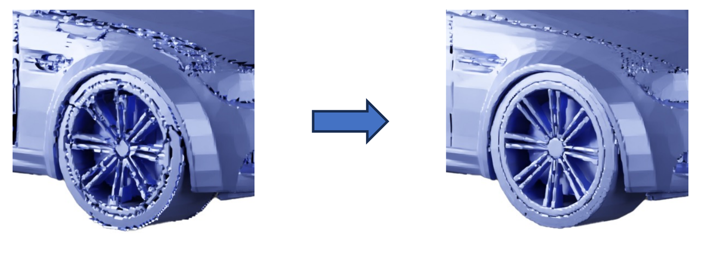
MC-based

- MeshUDF [6]
- CAP-UDF [7]
- DoubleCoveringUDF [8]

MeshUDF and CAP-UDF rely on **handcrafted** rules to find surface crossings inside cells.



MeshUDF is particularly suitable for garments but it can generate spurious triangles in more complex objects

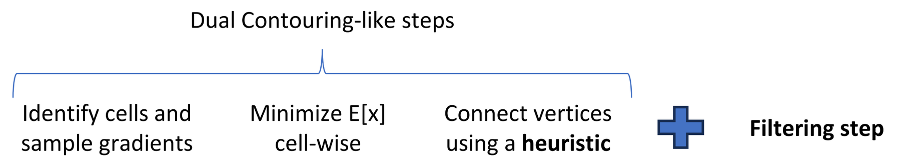


MeshUDF [6] Ours+MC [4]

DC-based

- Neural Dual Contouring [9]
- DualMesh-UDF [10]

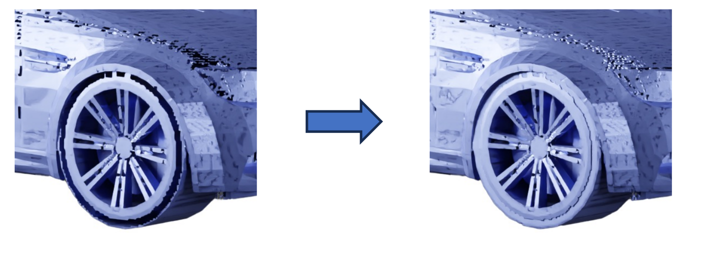
DualMesh-UDF proposes a Dual Contouring-like optimization algorithm with UDF-specific **heuristics**.



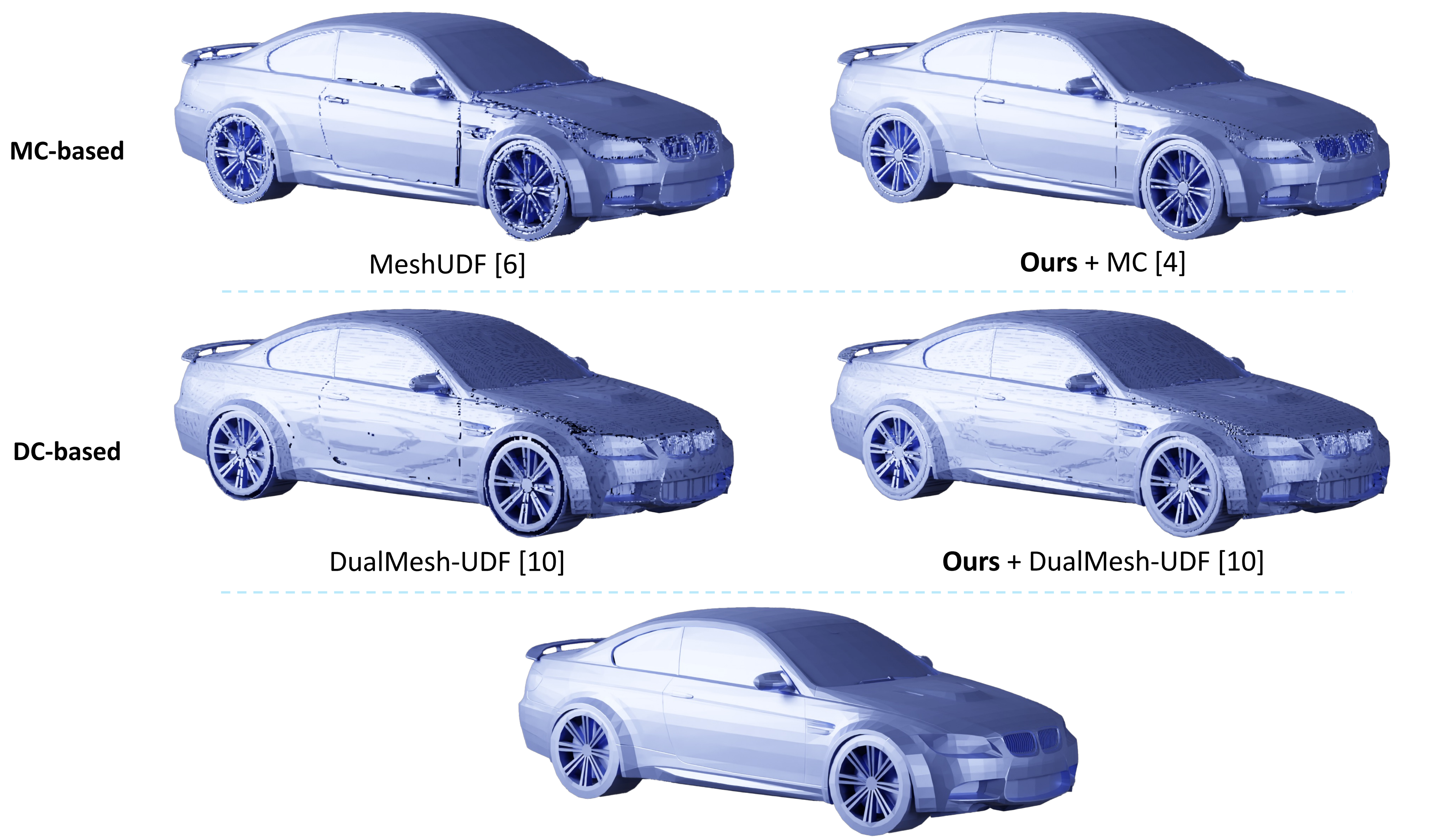
Minimize:

$$E[x] = \sum_{i=1}^{m} (\mathbf{n}_i \cdot (\mathbf{p}_i - \mathbf{x}) - d_i)^2$$

DualMesh-UDF achieves impressive accuracy, but it can miss parts of the surface if the minimization fails to converge

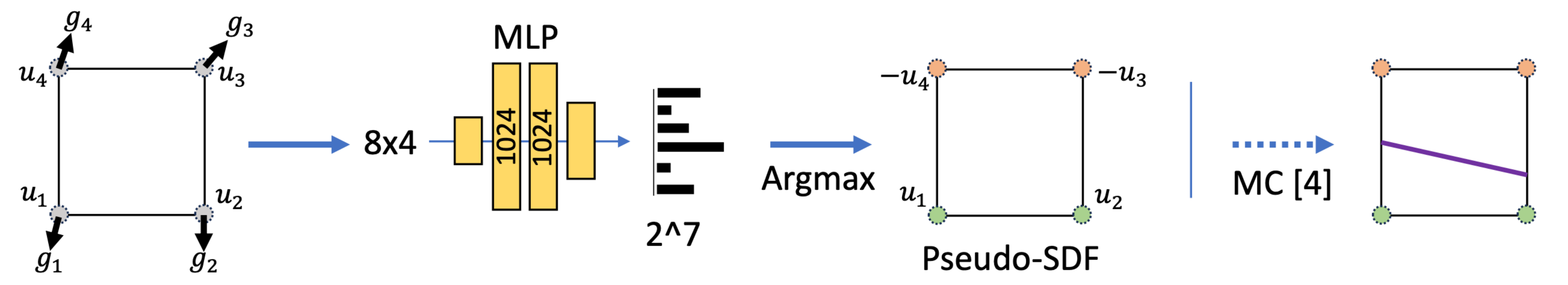


DMUDF [10] Ours+DMUDF [10]



3. Our approach

We aim to directly apply SDF meshing algorithms on a UDF by locally turning it into a Pseudo-SDF with a **data-driven** pipeline.



We use watertight shapes from ABC [11] for training, in order to have both a UDF input and an SDF ground-truth.
We normalize the field by the meshing resolution and inject noise to increase the network's robustness.

$$\mathcal{L}_{\Theta} = \sum_{\mathcal{S} \in \mathcal{D}} \sum_{c \in \mathcal{S}} CE(MLP_{\mathcal{S}}(c), GT_{\mathcal{S}}(c))$$

UDF normalization:

$$UDF_{\mathcal{S}}(c) = UDF_{\mathcal{S}}(c)/voxel\_size$$

Noise injection:

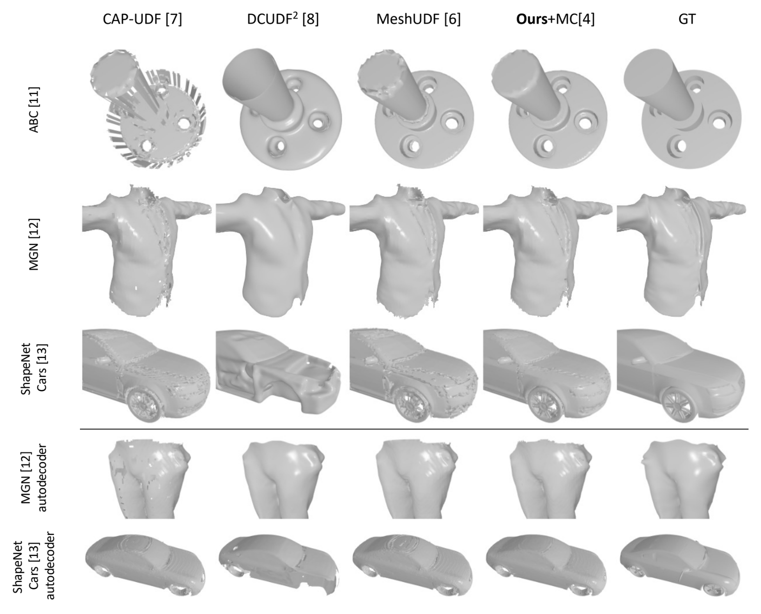
$$GT_{\mathcal{S}}(c) = GT_{\mathcal{S}}(c) * (1 + N(0,1))$$

We train the network with 80 shapes for **1 minute** on a single GPU. It can also be trained on **1 shape** with slitghly worse, but still usable, results.

Traditional meshing algorithms can be applied to the output with few modifications.

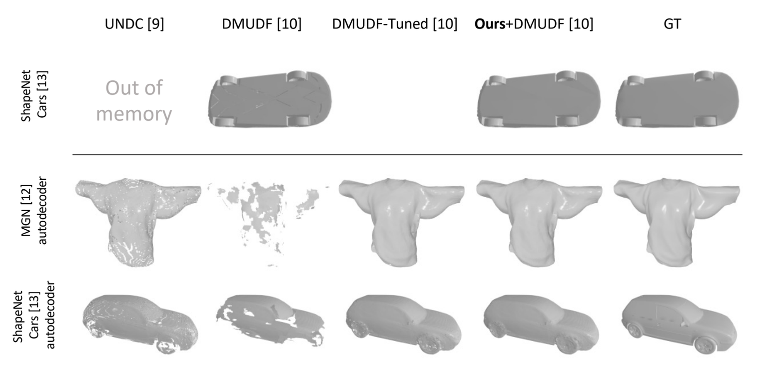
4. Results

1. **Results with MC-based methods**
We apply Marching Cubes [4] to the output of our network, recovering more accurate shapes compared to existing methods. We compute median Chamfer Distance and Image Consistency.



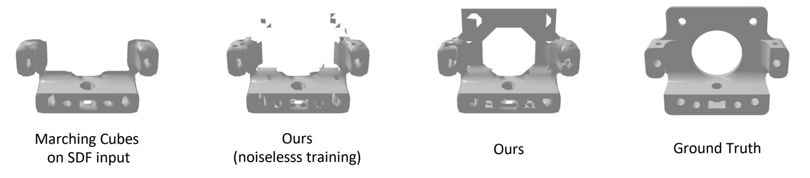
Res.	Method	ABC [11] CD ↓	ABC [11] IC ↑	MGN [12] CD ↓	MGN [12] IC ↑	Cars [13] CD ↓	Cars [13] IC ↑	MGN autodec. CD ↓	MGN autodec. IC ↑	Cars autodec. CD ↓	Cars autodec. IC ↑
32¹	CAP-UDF [7]	1070	51.0	138	64.9	54.1	79.3	219	57.2	191	71.9
	MeshUDF [6]	*48.8*	*89.4*	*18.1*	*91.0*	*20.9*	*84.4*	*18.4*	*90.9*	*24.1*	*86.4*
	DCUDF² [8]	1200	78.0	603	76.0	559	69.7	503	77.0	318	83.7
	Ours + MC [4]	**19.0**	**91.8**	**8.09**	**92.0**	**9.89**	**87.4**	**8.37**	**91.9**	**13.2**	**87.3**
	GT + MC	17.6	91.8	-	-	-	-	-	-	-	-
64¹	CAP-UDF	239	73.4	12.0	85.4	14.3	87.9	18.8	79.4	58.1	83.2
	MeshUDF	*6.76*	*94.7*	*3.55*	*94.7*	*7.51*	*89.8*	*3.89*	*94.1*	*14.9*	**88.6**
	DCUDF²	291	86.5	155	85.6	169	81.3	92.4	87.3	55.0	87.3
	Ours + MC	**4.46**	**95.4**	**2.10**	**95.5**	**4.47**	**91.2**	**2.64**	**94.7**	**10.0**	*88.5*
	GT + MC	4.52	95.7	-	-	-	-	-	-	-	-
128¹	CAP-UDF	21.3	94.1	1.99	95.7	4.19	*93.0*	3.13	91.8	39.9	87.6
	MeshUDF	*2.97*	*97.1*	*1.59*	*96.7*	*3.61*	92.8	*2.42*	95.0	*16.7*	**88.6**
	DCUDF²	44.6	92.5	26.1	92.3	113	87.7	6.34	**95.4**	355	78.2
	Ours + MC	**2.52**	**97.4**	**1.40**	**97.1**	**3.13**	**94.2**	**2.06**	*95.2*	**14.5**	*87.9*
	GT + MC	2.74	97.4	-	-	-	-	-	-	-	-
256¹	CAP-UDF	2.53	97.8	1.40	*97.7*	3.41	*95.0*	2.15	94.6	**37.8**	**87.5**
	MeshUDF	*2.42*	*98.0*	*1.36*	*97.7*	*3.10*	*95.0*	*2.05*	**94.8**	92.2	81.7
	DCUDF²	11.3	95.0	8.09	94.8	104	89.5	3.40	91.1	1930	36.0
	Ours + MC	**2.33**	**98.2**	**1.33**	**97.9**	**3.03**	**95.8**	**2.02**	**94.8**	*62.2*	*83.8*
	GT + MC	2.55	98.2	-	-	-	-	-	-	-	-

2. **Results with DC-based methods**
We relax the filtering step of DualMesh-UDF [5] and we use our pipeline as a simple filtering strategy, removing the need for hyperparameter tuning when dealing with neural UDFs.



Res.	Method	ABC [11] CD ↓	ABC [11] IC ↑	MGN [12] CD ↓	MGN [12] IC ↑	Cars [13] CD ↓	Cars [13] IC ↑	MGN autodec. CD ↓	MGN autodec. IC ↑	Cars autodec. CD ↓	Cars autodec. IC ↑
32¹	UNDC [9]	11.8	92.4	6.81	89.7	7.60	88.2	10.8	86.2	14.9	86.4
	DMUDF [10]	*9.83*	**97.0**	**2.55**	**94.3**	*4.15*	*92.2*	339	69.3	805	45.3
	DMUDF-T [10]	-	-	-	-	-	-	**2.84**	**94.0**	*6.96*	**89.8**
	Ours+DMUDF [10]	**9.66**	**97.0**	*2.74*	*94.2*	**3.92**	**92.3**	*3.00*	**94.0**	**6.73**	**89.8**
64¹	UNDC	0.783	95.8	0.926	92.9	1.35	91.0	1.83	90.7	16.8	85.9
	DMUDF	*0.579*	**98.6**	*0.195*	**96.9**	*0.813*	*94.1*	216	68.4	954	45.5
	DMUDF-T	-	-	-	-	-	-	*0.805*	**95.4**	*5.52*	*89.5*
	Ours+DMUDF	**0.574**	**98.6**	**0.194**	**96.9**	**0.787**	**94.2**	**0.803**	**95.4**	**5.43**	**89.8**
128¹	UNDC	0.0877	97.2	0.140	94.7	0.195	94.0	1.06	88.7	57.7	74.8
	DMUDF	*0.00450*	**99.0**	*0.0194*	**98.0**	*0.173*	**95.7**	176	66.4	846	45.1
	DMUDF-T	-	-	-	-	-	-	*0.722*	**95.1**	*10.6*	*87.0*
	Ours+DMUDF	**0.00492**	**99.0**	**0.0185**	**98.0**	**0.169**	**95.7**	**0.713**	**95.1**	**10.0**	**88.2**
256¹	UNDC	0.0146	98.0	0.0251	96.5	-	-	1.68	82.3	-	-
	DMUDF	**0.000143**	**99.1**	*0.00200*	**98.4**	*0.0731*	*96.3*	167	63.9	871	43.0
	DMUDF-T	-	-	-	-	-	-	*0.834*	*93.0*	*37.9*	*79.2*
	Ours+DMUDF	*0.000166*	**99.1**	**0.00199**	**98.4**	**0.0362**	**96.6**	**0.804**	**93.6**	**37.7**	**82.8**

3. **Importance of noise during training**
It improves robustness to neural UDFs and it helps detect thin surfaces at low resolutions.



4. **Meshing time**
For all methods except DCUDF, the meshing time is orders of magnitude lower than the UDF query time. Even so, when paired with MC, our meshing time is around 2x faster than the fastest MC-based method MeshUDF.

¹Resolution is doubled for Cars
²DCUDF can achieve better accuracy by removing the cutting step. However this would make the surface double layered.

References

[1] **DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation**, Park et al., CVPR 2019
[2] **Occupancy Networks: Learning 3D Reconstruction in Function Space**, Mescheder et al., CVPR 2019
[3] **Learning Implicit Fields for Generative Shape Modeling**, Chen et al., CVPR 2019
[4] **Marching Cubes: a High Resolution 3D Surface Construction Algorithm**, Lorensen and Cline, SIGGRAPH 1987
[5] **Dual Contouring of Hermite Data**, Ju et al., SIGGRAPH 2002
[6] **MeshUDF: Fast and Differentiable Meshing of Unsigned Distance Field Networks**, Guillard et al., ECCV 2022
[7] **Learning Consistency-Aware Unsigned Distance Functions Progressively from Raw Point Clouds**, Zhou et al., NeurIPS 2022
[8] **Robust Zero Level-Set Extraction from Unsigned Distance Fields Based on Double Covering**, Hou et al., SIGGRAPH Asia 2023
[9] **Neural Dual Contouring**, Chen Z., SIGGRAPH 2022
[10] **Surface Extraction from Neural Unsigned Distance Fields**, Zhang et al., CVPR 2023
[11] **ABC: A Big CAD Model Dataset for Geometric Deep Learning**, Koch et al., CVPR 2019
[12] **Multi-Garment Net: Learning to Dress 3D People from Images**, Bhatnagar et al., ICCV 2019
[13] **ShapeNet: An Information-Rich 3D Model Repository**, Chang et al., 2015

GitHub