# Neural Dual Contouring Reproduction Report

苏富宝

**Abstract**

We reproduce the algorithm of an excellent article published in Siggraph 2022, *neural dual contouring* (NDC), a new data-driven approach to mesh reconstruction based on dual contouring (DC). Like traditional DC, it produces exactly one vertex per grid cell and one quad for each grid edge intersection, a natural and efficient structure for reproducing sharp features. However, rather than computing vertex locations and edge crossings with hand-crafted functions that depend directly on difficult-to-obtain surface gradients, NDC uses a neural network to predict them.We use our own dataset, which was filtered from SPAIEN, to train the NDC network.During experiments with the pointcloud as the input data, we find that NDC generalizes well to the other methods.Furthermore, NDC provides better surface reconstruction accuracy, feature preservation, output complexity, triangle quality compared with other methods.

**Keywords:** Mesh Reconstruction, Dual Contouring, Neural Network, Reproduction.

## 1 Introduction

The reconstruction of polygonal meshes from discrete inputs such as point clouds and voxel meshes is one of the most classical and well-studied problems in computer graphics [3, 10, 5]. Current solutions to this problem are primarily model-driven and typically rely on assumptions about shape features, surface interpolation, sampling conditions, surface normals, and other reconstruction priors. Only recently have a number of data-driven meshing methods been developed[5].One exception is Neural Marching Cubes (NMC)[6], which is a deep learning-based approach to reconstructing 3D shapes from their point clouds or voxel grids. It combines the principles of the Marching Cubes algorithm with the power of neural networks to produce high-quality, smooth and detailed surface meshes. NMC uses deep learning models to predict surface vertices and edge connectivity, allowing it to handle complex and noisy shapes more effectively than traditional marching cube methods. Neural dual contouring (NDC) is a new approach proposed by Chen and Zhang with lighter and more efficient features. NDC uses deep learning models to predict surface vertices and edge connectivity, allowing it to handle complex and noisy shapes more effectively compared to traditional dual contouring methods. Like Neural Marching Cubes (NMC), NDC is a promising technique for 3D shape reconstruction and has the potential to be used in a wide range of applications, including computer graphics, robotics, and virtual/augmented reality.

In order to thoroughly test and validate the Neural Dual Contouring algorithm, I conducted numerous experiments to assess its performance and accuracy. These experiments covered a wide range of scenarios and conditions and provided valuable insight into the strengths and limitations of the ap-

proach, testing not only on the ABC dataset but also on our own dataset. The experiments show that NDC performs better than other reconstruction methods.

# 2 Related works

To introduce the work related to NDC, in this section, we focus on techniques for equivalence surfaces (i.e., mesh extraction from discrete body data) and surface reconstruction from point cloud data, highlighting data-driven methods that have recently become closely related to our work. The literature on mesh reconstruction is extensive, so we refer to a number of reviews that provide comprehensive coverage which was mentioned in the NDC paper.

## 2.1 Isosurfacing and differentiable reconstruction

The marching cube (MC) approach to isosurfacing from discrete signed distances was first proposed simultaneously by Lorensen[2] and Cline and Wyvill[11]. Since then, many variants have followed, including the best-known MC33[7], which correctly enumerated all possible topological cases for mesh tessellations based on the trilinear interpolation assumption[5].The motivation for NDC's work comes from the improvement of NMC[6]. Although NMC is a very innovative work, as it uses a neural network to learn a mapping from a voxel grid to a set of triangular mesh vertices, effectively performing the Marching Cubes algorithm in a differentiable and learned way. NDC combines deep learning with dual contouring (DC)[12] to bring key advantages of classical DC over MC into a learned mesh reconstruction model without requiring additional inputs, which improves efficiency and reconstruction quality[5].

## 2.2 Mesh reconstruction from point clouds

Mesh reconstruction is a technique used in computer graphics and visualization to represent 3D objects and scenes as a collection of interconnected polyggonal faces. The goal of mesh reconstruction is to generate a simplified, yet accurate, representation of the underlying surface geometry.For surface mesh estimation from unorganised points, many methods have been proposed. Following the taxonomy in Berger[4], previous works can be characterised on the basis of the underlying priors, e.g. smoothness[13], visibility[8], dense sampling[1], primitives[9] and learning from data[15]. Both NMC[6] and NDC[5] fall into the latter category: They learn implicit priors for local regions.

## 2.3 Dual Contouring (DC)

Marching Cubes is a 3D surface reconstruction algorithm used in computer graphics and visualization, see Figure 1. It is used to extract a triangular mesh representation of an implicit surface defined by a scalar field (e.g., a 3D volumetric data set). The algorithm works by iteratively dissecting a 3D grid into cubes, determining the surface configuration within each cube, and connecting the vertices of the triangles that define the surface. The resulting mesh approximates the surface of the implicit function, and can be used in various applications, such as visualizing medical imaging data or terrain in computer games.

Dual contouring[12] is a 3D mesh generation technique used in computer graphics to create a 3D surface representation from volumetric data, see Figure 2. It works by iteratively finding the intersections between a grid of voxels and the implicit surface, and then fitting a 3D surface to these intersections. The resulting mesh is optimised for both smoothness and accuracy, making it well suited for use in interactive 3D applications such as video games and virtual reality simulations.

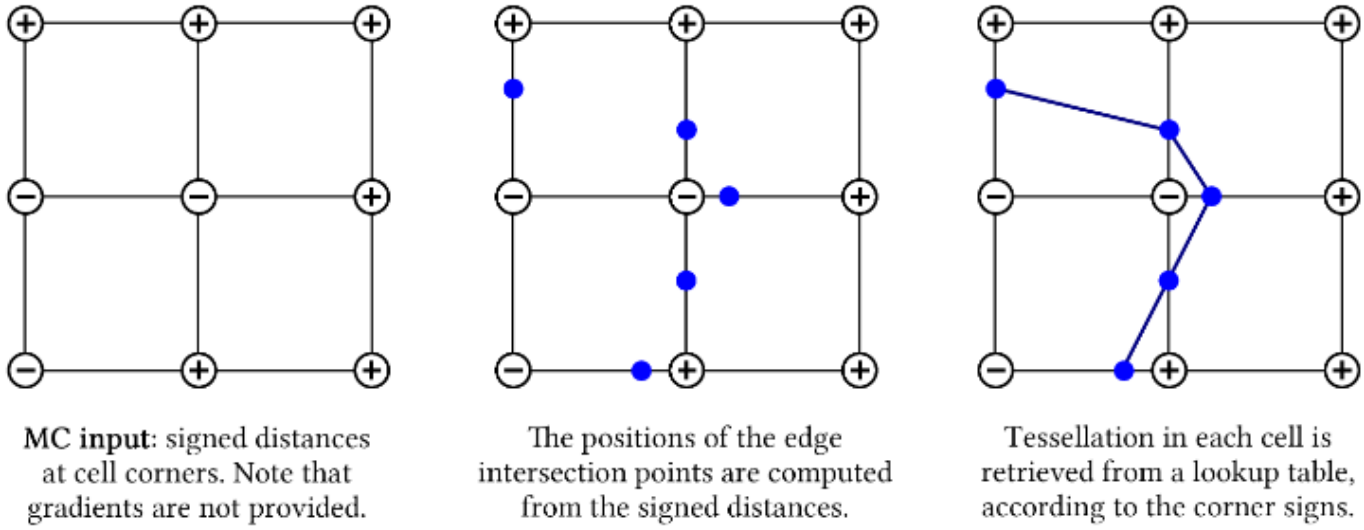


Figure 1: **Marching Cubes (MC)** visualized in 2D on example inputs.

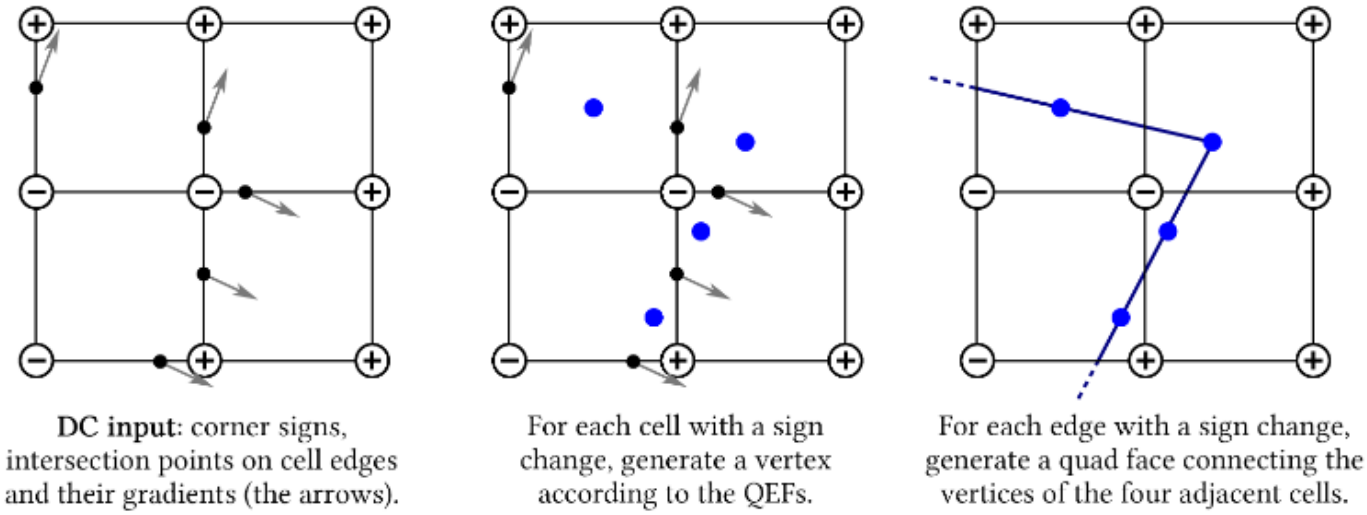


Figure 2: **Dual Contouring(DC)** visualized in 2D on example inputs.

# 3 Method

Neural Dual Contouring is a novel approach to 3D mesh reconstruction that combines deep learning and geometry-based methods, that achieves the simplicity and sharp features of DC without requiring function gradients in the input. In this section, We will describe the details of the algorithm of NDC and UNDC. We will also introduce the architecture of neural network when the point cloud is the input. Finally, we will introduce the loss function for training neural networks.

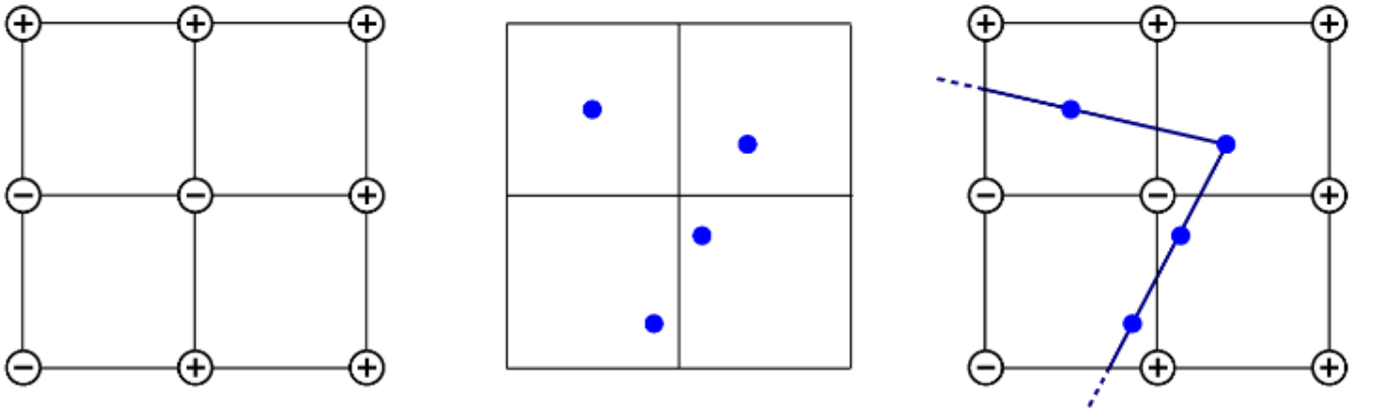


Figure 3: **Neural Dual Contouring(NDC)**

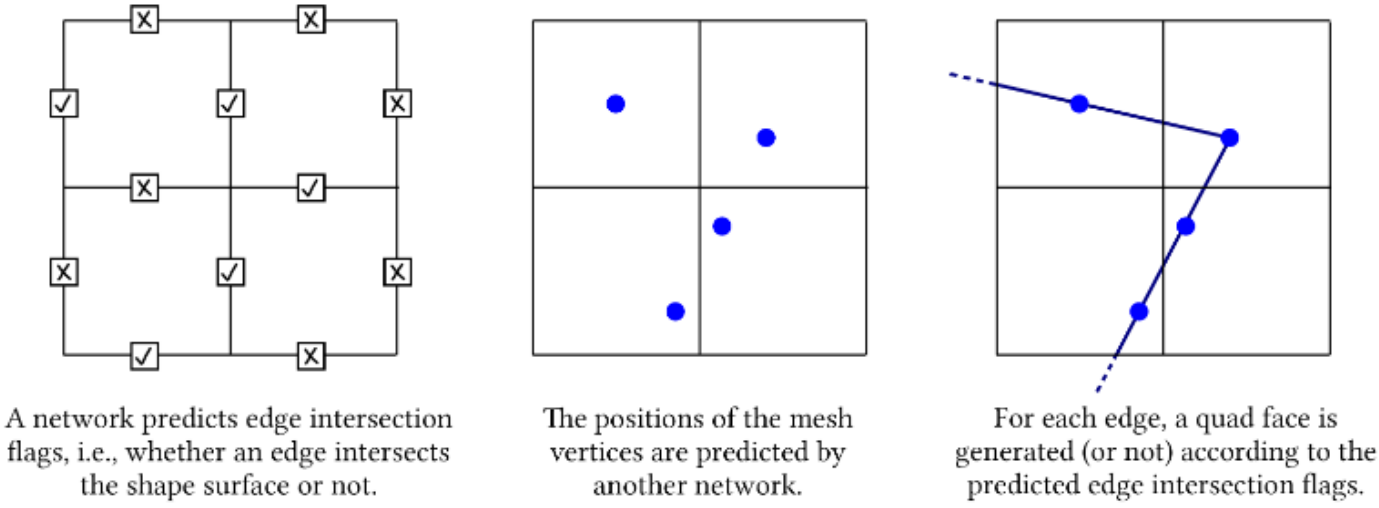


Figure 4: **Unsigned Neural Dual Contouring(UNDC)**

## 3.1 The algorithm of NDC and UNDC

In Neural Dual Contouring, the neural network is trained to predict a set of signed distances, which define the implicit surface. The output of the network is then used as input to Dual Contouring, which generates a 3D mesh that approximates the implicit surface.The key advantage of Neural Dual Contouring is its ability to incorporate the strengths of both deep learning and traditional geometry-based methods. The deep learning component provides fast, accurate surface representation, while the Dual Contouring component provides control over mesh quality and structure. The first and standard variant of NDC method, reconstructing networks on the basis of predicted signs, is simply called NDC. As Dual contouring assumes as input:

$$\mathcal{S} \in \mathbb{B}^{|X|}, \quad \mathcal{S} = f_{\mathcal{S}}(\Phi, \mathcal{G}), \quad \text{(grid signs)} \tag{1}$$

$$\mathcal{V}^{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}|\times 3}, \quad \mathcal{V}^{\mathcal{E}} = f_{\mathcal{V}^{\mathcal{E}}}(\Phi, \mathcal{G}), \quad \text{(edge vertices)} \tag{2}$$

$$\mathcal{N}^{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}|\times 3}, \quad \mathcal{N}^{\mathcal{E}} = f_{\mathcal{N}^{\mathcal{E}}}(\Phi, \mathcal{G}), \quad \text{(edge normals)} \tag{3}$$

Neural Dual Contouring(NDC) employs a neural network trained on example 3D surface data to predict the vertex locations. In particular, a 2D example was illustrated in Figure 3 .More specifically, NDC

can be algebraically formalized as:

$$\mathrm{NDC}(\mathcal{I}) = \begin{cases} \mathcal{S} = f_{\mathcal{S}}(\mathcal{I}, \mathcal{G}; \theta) \\ \mathcal{V} = f_{\mathcal{V}}(\mathcal{I}, \mathcal{G}; \theta) \\ \mathcal{F} = \mathrm{xor}\,(\mathcal{S}_i, \mathcal{S}_j) \end{cases} \tag{4}$$

Obviously, NDC obtains the sign of the edge by predicting the sign of the point from the network, thus the network can predict the sign of the edge directly, thus reducing the difficulty of the network prediction, that is Unsigned Neural Dual Contouring(UNDC), see Figure 4. UNDC can be algebraically formalized as:

$$\mathrm{UNDC}(\mathcal{I}) = \begin{cases} \mathcal{V} = f_{\mathcal{V}}(\mathcal{I}, \mathcal{G}; \theta), \\ \mathcal{F} = f_{\mathcal{F}}(\mathcal{I}, \mathcal{G}; \theta). \end{cases} \tag{5}$$

The main advantage of this variant is that it can generate surface intersections without relying on inside/outside sign differences at cell vertices. This feature allows UNDC to operate on unsigned distance fields or unoriented point clouds (we use the prefix U to indicate the ability of this variant to operate on unsigned inputs). It also allows UNDC, in regions where the underlying object parts are thinner than one voxel, to generate mesh faces, presumably in the form of thin sheets[5].

## 3.2 Neural network architecture

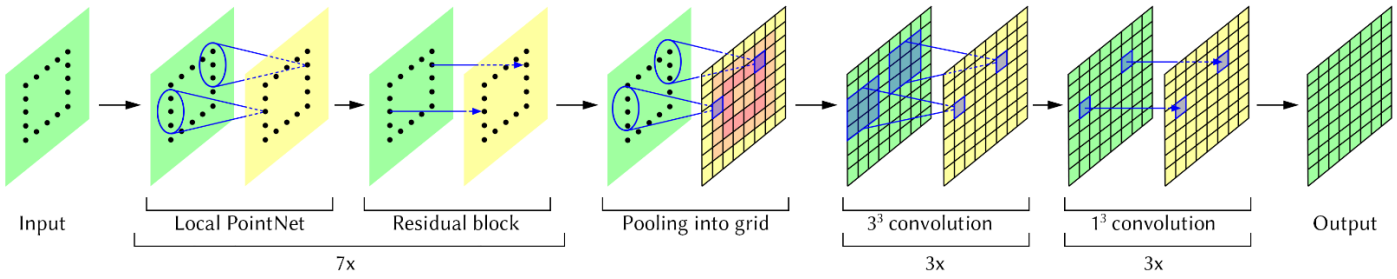


Figure 5: The architecture of our point cloud processing network for UNDC.

For point cloud inputs, we devise a local point cloud encoder network divided into two parts: 1 point cloud processing and 2 regular grid processing. The former is implemented as a dense PointNet++[14], while grid processing has three 33 convolution layers and three 13 convolution layers, hence of a similar architecture to the one used for inputs represented as grids.[5] The network architecture is shown in Figure 5.

The architecture of NDC consists of a multi-layer neural network that maps the input point cloud to a set of implicit surface functions. These implicit functions are then combined to form a scalar field that describes the density of the object surface. The NDC algorithm then uses a contouring technique to extract the zero-level set of the scalar field, which defines the 3D triangular mesh surface of the object.The NDC architecture can be trained end-to-end using supervised learning, with ground-truth meshes as the target outputs. The neural network is trained to predict the scalar field that best approximates the surface of the object, given the input point cloud.

### 3.3 Training losses

Given the ground truth data, it should be noted that all of the sub-networks within the NDC and UNDC models can be trained separately, resulting in a simpler training setup where there is no need for hyper-parameter tuning between the losses. NDC use a simple L2 reconstruction loss of pseudo ground-truth vertices:

$$\mathcal{L}_{\mathcal{V}}(\theta) = \mathbb{E}_{(I,\mathcal{M}_{\mathcal{V}},\mathcal{V}_{\text{gt}})\sim\mathcal{D}} \sum_{m,n,k} \left[\|\mathcal{M}_{\mathcal{V}} \odot (f_{\mathcal{V}}(\mathcal{I},\mathcal{G};\theta) - \mathcal{V}_{\text{gt}})\|_2^2\right] \tag{6}$$

While NDC supervise the prediction of signs via Binary Cross Entropy (BCE):

$$\mathcal{L}_{\mathcal{S}}(\theta) = \mathbb{E}_{(I,\mathcal{M}_{\mathcal{S}},\mathcal{S}_{\text{gt}})\sim\mathcal{D}} \sum_{m,n,k} \left[\mathcal{M}_{\mathcal{S}} \odot \text{BCE}\left(f_{\mathcal{S}}(\mathcal{I},\mathcal{G};\theta), \mathcal{S}_{\text{gt}}\right)\right] \tag{7}$$

## 4 Implementation details

### 4.1 Comparing with released source codes

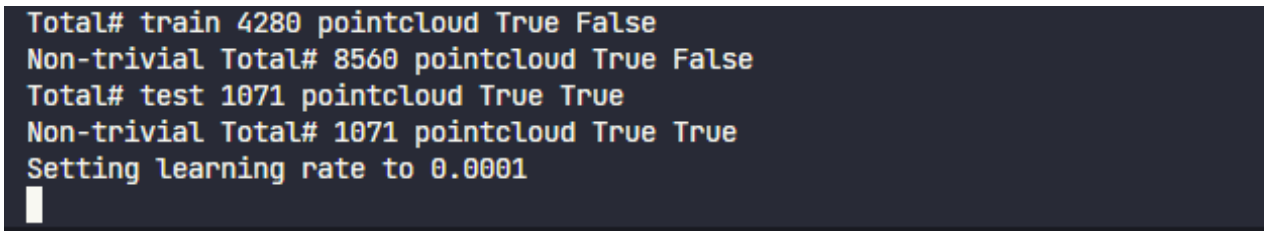


Figure 6: The interface of the code of UNDC while training.

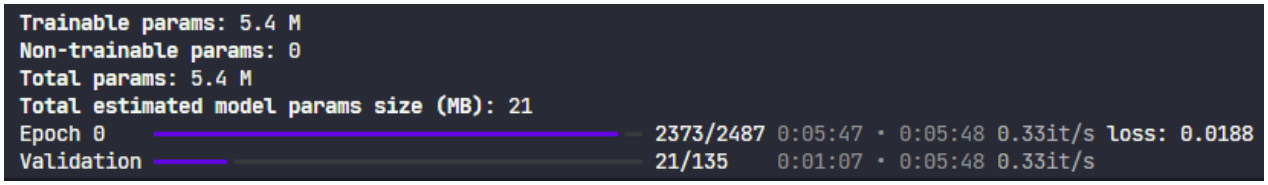


Figure 7: The interface of the code of our while training.

The code and dataset of ndc are available at https://github.com/czq142857/NDC. But the code released by ndc suffers from a number of engineering problems. Firstly, the code of ndc violates the open and close principle of the coding principles, it is very difficult to modify its code when you want to modify its neural network architecture or train in your own dataset. Secondly, the code of ndc does not have a good code organisation and is very difficult to read. What is more, it does not have a good visual interface to reflect the progress of the training and it is not possible to view the intermediate results of the training.

In our code (https://github.com/JYILY/NDC), we use high-level structure (see Figure 8) for PyTorch code to simplifies the management of large-scale experiments settings, we can easily modify any hyperparameters or the structure of the network to allow for more flexible experiments.Depending on the experimental device conditions, our code makes it very simple to set up the model for training on a CPU or GPU, or multiple GPUs. What is more, our code has very good visualisation (see Figure 6 7)and we can easily observe the training progress and intermediate results of the training (see Figure **??**), while being able to easily save the best model parameters.

```
Global seed set to 2023
[2023-02-06 05:01:43,052][__main__][INFO] - Instantiating datamodule <src.datamodules.undc_pc_datamodule_fight.UNDC_PC_DataModule>
[2023-02-06 05:01:43,496][__main__][INFO] - Instantiating model <src.models.undc_module_pre.UNDCLitModule>
Loading pretrain weight
Done
[2023-02-06 05:01:51,113][__main__][INFO] - Instantiating callbacks...
[2023-02-06 05:01:51,115][src.utils.utils][INFO] - Instantiating callback <pytorch_lightning.callbacks.ModelCheckpoint>
[2023-02-06 05:01:51,121][src.utils.utils][INFO] - Instantiating callback <pytorch_lightning.callbacks.EarlyStopping>
[2023-02-06 05:01:51,122][src.utils.utils][INFO] - Instantiating callback <pytorch_lightning.callbacks.RichModelSummary>
[2023-02-06 05:01:51,123][src.utils.utils][INFO] - Instantiating callback <pytorch_lightning.callbacks.RichProgressBar>
[2023-02-06 05:01:51,124][__main__][INFO] - Instantiating loggers...
[2023-02-06 05:01:51,124][src.utils.utils][WARNING] - Logger config is empty.
[2023-02-06 05:01:51,124][__main__][INFO] - Instantiating trainer <pytorch_lightning.Trainer>
Trainer already configured with model summary callbacks: [<class 'pytorch_lightning.callbacks.rich_model_summary.RichModelSummary'>]. S
allback.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
[2023-02-06 05:01:51,558][__main__][INFO] - Starting training!
[2023-02-06 05:01:51,596][src.datamodules.undc_pc_datamodule_fight][INFO] - Loading Train dataset...
[2023-02-06 05:01:51,680][src.datamodules.undc_pc_datamodule_fight][INFO] - Train dataset [ABC_pc_hdf5] of size 2352 had been created.
[2023-02-06 05:01:51,680][src.datamodules.undc_pc_datamodule_fight][INFO] - Loading Val dataset...
[2023-02-06 05:01:51,735][src.datamodules.undc_pc_datamodule_fight][INFO] - Val dataset [ABC_pc_hdf5] of size 135 had been created.
[2023-02-06 05:01:51,735][src.datamodules.undc_pc_datamodule_fight][INFO] - Loading Test dataset...
[2023-02-06 05:01:51,790][src.datamodules.undc_pc_datamodule_fight][INFO] - Test dataset [ABC_pc_hdf5] of size 135 had been created.
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0,1,2,3,4,5,6,7,8,9]
```

Figure 8: The log of our code.

## 4.2 Main contributions

On the one hand, we provided more robust code for ndc.While it is the idea that is at the heart of ndc, how to implement it more rationally is also a very important issue. Based on the original code of ndc, we have applied a more rational engineering knowledge to organise the code of ndc in a more rational way, making it more robust for more flexible settings and more detailed experiments.

On the other hand, We explored the uses of ndc. We experimented more carefully with ndc on our dataset and implemented new work using ndc. ndc played an important role in the interactive modelling project (see Figure 9) we carried out, improving the quality of the reconstructions and having sharper edges.

The most important contribution is that we have learned profoundly from the ideas of the ndc, explored new scientific findings based on the ndc, and continued to improve the ndc.

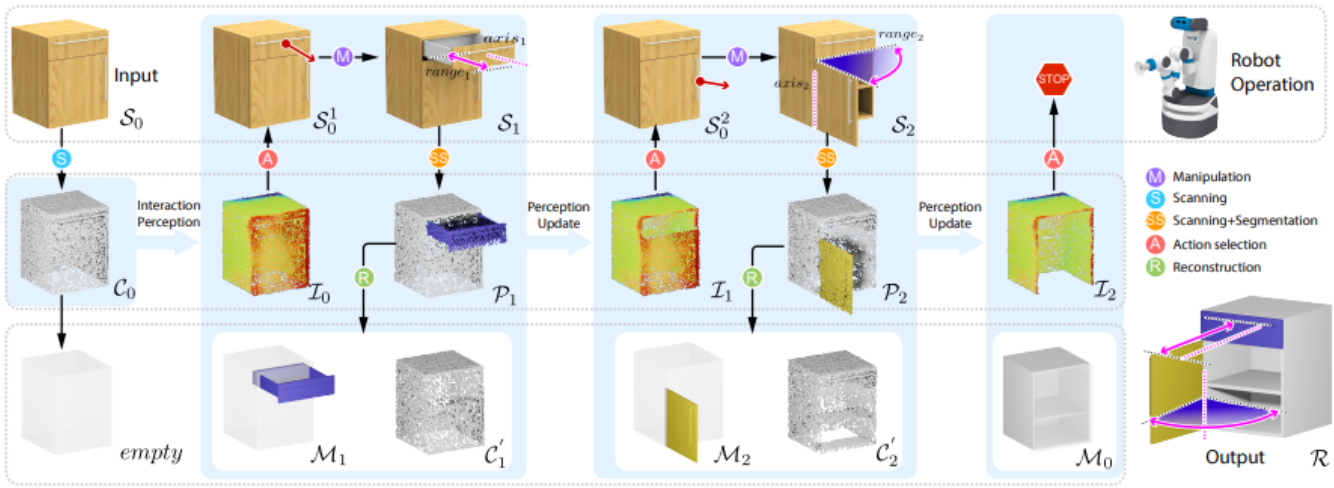


Figure 9: Interaction-Driven Active 3D Reconstruction with Object Interiors.

# 5 Results and analysis

| | CD↓ | F1↑ | NC↑ | ECD↓ | EF1↑ |
|---|---|---|---|---|---|
| **ConvONet** | 25.6 | 0.38 | 0.72 | 15.9 | 0.08 |
| **NMC** | 5.89 | 0.65 | 0.86 | 1.35 | 0.58 |
| **NDC** | 2.33 | **0.82** | 0.91 | 0.77 | 0.68 |
| **UNDC** | **1.57** | 0.81 | **0.95** | **0.51** | **0.76** |

Table 1: Quantitative results on test set of our dataset with point cloud input.

As the table and Figure 10 show, NDC has many advantages. Firstly, NDC has a higher accuracy rate, the deep learning component of Neural Dual Contouring allows for accurate surface representation, even in the presence of noise or incomplete data.Secondly, NDC can control the quality of reconstruction. The Dual Contouring component of Neural Dual Contouring provides control over the quality and structure of the generated mesh, ensuring that it meets the requirements of the application.In addition, NDC is really robust. Neural Dual Contouring is robust to noise and sparse data, which are common in real-world applications, such as medical imaging or computer vision.

In summary, Neural Dual Contouring offers a fast, accurate, and flexible approach to 3D mesh reconstruction, with the potential to significantly improve the quality and efficiency of various applications.

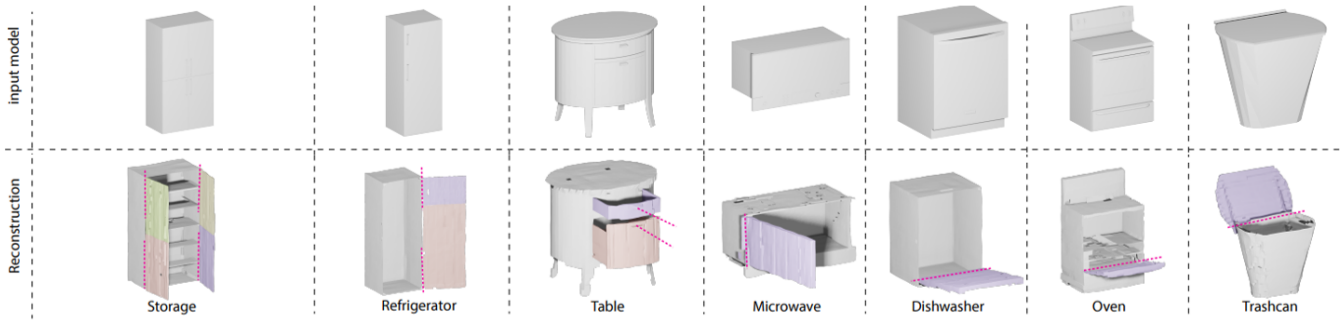


Figure 10: The results of the ndc of our work.

# 6 Conclusion and future work

The future work of Neural Dual Contouring is likely to focus on improving the accuracy and efficiency of the technique, as well as expanding its range of applications. Some potential areas of future work include:

Improving network architecture: Researchers may explore different network architectures and training strategies to improve the accuracy of the implicit surface representation learned by the neural network.

Combining with other techniques: Researchers may investigate ways to combine Neural Dual Contouring with other deep learning and geometry-based techniques to improve the quality and accuracy of the generated mesh.

Handling noisy or sparse data: Neural Dual Contouring may be extended to handle noisy or sparse data, which is common in many real-world applications, such as medical imaging or computer vision.

Scalability: Researchers may investigate methods to scale Neural Dual Contouring to handle large and complex datasets, making it more suitable for use in real-time applications.

Expanding applications: Neural Dual Contouring may be extended to a wider range of applications, such as robotics, virtual reality, and simulation, to enable more accurate and efficient representation of 3D objects and scenes.

# References


[1] Nina Amenta, Marshall Bern, and Manolis Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 415–421, 1998.

[2] Abhishek Badki, Orazio Gallo, Jan Kautz, and Pradeep Sen. Meshlet priors for 3d mesh reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2849–2858, 2020.

[3] Matthew Berger, Andrea Tagliasacchi, Lee M Seversky, Pierre Alliez, Gael Guennebaud, Joshua A Levine, Andrei Sharf, and Claudio T Silva. A survey of surface reconstruction from point clouds. In *Computer graphics forum*, volume 36, pages 301–329. Wiley Online Library, 2017.

[4] Matthew Berger, Andrea Tagliasacchi, Lee M Seversky, Pierre Alliez, Gael Guennebaud, Joshua A Levine, Andrei Sharf, and Claudio T Silva. A survey of surface reconstruction from point clouds. In *Computer graphics forum*, volume 36, pages 301–329. Wiley Online Library, 2017.

[5] Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. Neural dual contouring. *ACM Transactions on Graphics (TOG)*, 41(4):1–13, 2022.

[6] Zhiqin Chen and Hao Zhang. Neural marching cubes. *ACM Transactions on Graphics (TOG)*, 40(6):1–15, 2021.

[7] Evgeni Chernyaev. Marching cubes 33: Construction of topologically correct isosurfaces. Technical report, 1995.

[8] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.

[9] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.

[10] Bruno Rodrigues De Araújo, Daniel S Lopes, Pauline Jepp, Joaquim A Jorge, and Brian Wyvill. A survey on implicit surface polygonization. *ACM Computing Surveys (CSUR)*, 47(4):1–39, 2015.

[11] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2mesh: A self-prior for deformable meshes. *arXiv preprint arXiv:2005.11084*, 2020.

[12] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 339–346, 2002.

[13] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013.

[14] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[15] Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo. Deep geometric prior for surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10130–10139, 2019.