

# Learning and Meshing from Deep Implicit Surface Networks Using an Efficient Implementation of Analytic Marching

Jiabao Lei, Kui Jia, and Yi Ma, *Fellow, IEEE*

**Abstract**—Reconstruction of object or scene surfaces has tremendous applications in computer vision, computer graphics, and robotics. The topic attracts increased attention with the emerging pipeline of deep learning surface reconstruction, where implicit field functions constructed from deep networks (e.g., multi-layer perceptrons or MLPs) are proposed for generative shape modeling. In this paper, we study a fundamental problem in this context about recovering a surface mesh from an implicit field function whose zero-level set captures the underlying surface. To achieve the goal, existing methods rely on traditional meshing algorithms (e.g., the de-facto standard marching cubes); while promising, they suffer from loss of precision learned in the implicit surface networks, due to the use of discrete space sampling in marching cubes. Given that an MLP with activations of Rectified Linear Unit (ReLU) partitions its input space into a number of linear regions, we are motivated to connect this local linearity with a same property owned by the desired result of polygon mesh. More specifically, we identify from the linear regions, partitioned by an MLP based implicit function, the *analytic cells* and *analytic faces* that are associated with the function's zero-level isosurface. We prove that under mild conditions, the identified analytic faces are guaranteed to connect and form a *closed, piecewise planar surface*. Based on the theorem, we propose an algorithm of *analytic marching*, which marches among analytic cells to *exactly* recover the mesh captured by an implicit surface network. We also show that our theory and algorithm are equally applicable to advanced MLPs with shortcut connections and max pooling. Given the parallel nature of analytic marching, we contribute *AnalyticMesh*, a software package that supports efficient meshing of implicit surface networks via CUDA parallel computing, and mesh simplification for efficient downstream processing. We apply our method to different settings of generative shape modeling using implicit surface networks. Extensive experiments demonstrate our advantages over existing methods in terms of both meshing accuracy and efficiency. Codes are at <https://github.com/Karbo123/AnalyticMesh>.

**Index Terms**—Generative shape modeling, implicit surface representation, polygon mesh, deep learning, multi-layer perceptron.

## 1 INTRODUCTION

CREATION of 3D content prepares geometric data useful for analysis and processing in many scientific fields. For example, in computer vision and robotics, object or scene surface reconstruction via simultaneous localization and mapping [1] enables robotic manipulation, indoor navigation, and urban modeling; in computer graphics, reconstruction of continuous surface from discrete raw scanning is the first step in computer-aided design, virtual/augmented reality, and movie production. The geometric data created in these applications are of 2-dimensional manifold embedded in the 3D space. In this work, we are particularly interested in those data of closed manifolds representing, e.g., the boundary of a 3D solid.

As a mathematical notion of geometry, a continuous surface manifold is difficult to be modeled directly; in practice, it is approximated as different representations, such as spline surface, subdivision surface, or polygon mesh [2]. Among them, the polygon mesh is arguably the most popular representation proposed in the literature, which

approximates a smooth surface explicitly as a piecewise, linear function; for example, the most typical triangle mesh is defined as a collection of connected faces, each of which has three vertices that uniquely determine plane parameters of the face in the 3D space. Given the parametric mappings specified by planar faces of a polygon mesh, the representation is advantageous in surface evaluation and rendering; however, it is usually difficult to obtain a mesh directly, especially for topologically complex surface; queries of points inside or outside the surface are expensive as well. As an alternative, one may resort to implicit surface representations, such as signed distance function (SDF) [3], [4] or occupancy field (OF) [5], [6], which subsume a surface as the zero-level isosurface in the function field; other implicit representations include discrete volumes and those based on algebraic [7], [8], [9] and radial basis functions [10], [11], [12]. To obtain a surface mesh, the continuous field is often sampled discretely as a regular grid of voxels, followed by the de-facto standard algorithm of marching cubes [13]. Efficiency and precision of marching cubes can be improved on a hierarchically sampled structure of octree via algorithms such as dual contouring [14].

Implicit functions are traditionally implemented based on moving least squares [15]. More recently, methods of deep learning surface reconstruction [4], [5], [6] propose to leverage the great modeling capacities of deep networks (e.g., Multi-Layer Perceptrons (MLPs) based on Rectified Linear Unit (ReLU) [16]) to learn implicit fields. Given a

- J. Lei and K. Jia are with the School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China. E-mails: [eejblei@mail.scut.edu.cn](mailto:eejblei@mail.scut.edu.cn), [kuijia@scut.edu.cn](mailto:kuijia@scut.edu.cn).
- Y. Ma is with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720-1770 USA. E-mails: [yima@eecs.berkeley.edu](mailto:yima@eecs.berkeley.edu).
- Correspondence to: K. Jia.

learned field, they again take a final step of marching cubes to obtain the mesh result. While promising, the final step of marching cubes recovers a mesh that is only an approximation of the surface captured by the learned implicit network; more specifically, it suffers from a trade-off of sampling efficiency and recovery precision, due to the discretization nature of the marching cubes algorithm. The very recent deep models using soft ReLU [17] or sine/cosine activation functions [18] suffer from this limitation as well.

To address the limitation, we are motivated from the established knowledge that a ReLU based MLP partitions its input space into a number of linear regions [19]; this connects with the locally linear property of polygon mesh. Given an MLP based implicit function, we identify from its partitioned linear regions the *analytic cells* and *analytic faces* that are associated with the function’s zero-level isosurface. Assuming that such an implicit function learns its zero-level isosurface as a *closed, piecewise planar surface*, we characterize theoretical conditions under which analytic faces of the implicit function *connect and exactly form* the surface mesh. Based on our theorem, we propose an algorithm of *analytic marching*, which marches among analytic cells to recover the *exact mesh* of the closed, piecewise planar surface captured by a learned MLP. Our choices of MLPs also include those with shortcut connections and max pooling. The proposed analytic marching algorithm can be naturally implemented in parallel, for which we contribute *AnalyticMesh*, a software package that supports efficient meshing of implicit surface networks via CUDA parallel computing, and post-processing of mesh simplification. We apply our meshing algorithm in the contexts of either direct shape decoding of raw point observations, or learning to reconstruct novel shape instances using global or local decoders. Experiments on benchmark 3D object repositories show the advantages of our meshing algorithm over existing ones.

## 1.1 Relations with the Literature

The problem studied in this work is closely related to the following three lines of research.

**Implicit Surface Representations.** An implicit surface representation is defined as the zero-level set of an scalar-valued implicit function. Earlier methods take a divide-and-conquer strategy that represents the surface using atom functions. For example, blobby molecule [7] is proposed to approximate each atom by a gaussian potential, and a piecewise quadratic meta-ball [8] is used to approximate the gaussian, which is improved via a soft object model in [9] by using a sixth degree polynomial. Radial basis function (RBF) is an alternative to the above algebraic functions. RBF-based approaches [10], [11], [12] place the function centers near the surface and are able to reconstruct a surface from a discrete point cloud. It has been recently discovered that deep networks, owing to their great modeling capacities, are able to learn implicit surface fields very effectively. DeepSDF [4] trains ReLU based MLPs as signed distance functions. IMNet [5] and OccNet [6] learn similar types of networks as occupancy fields. Deep implicit surface networks are also used in [20] for surface reconstruction from as few as a single image. Other than ReLU based networks, smooth activations such as soft ReLU [17] or sine/cosine func-

tions [18] have been showing the new promise for learning smoother surfaces via implicit fields. We focus on ReLU based networks in the present work.

**Mesh Conversions from Implicit Fields.** The conversion from an implicit representation to an explicit surface mesh is called isosurface extraction. Probably the simplest approach is to directly convert an implicit volume via greedy meshing (GM). The de-facto standard algorithm of marching cubes (MC) [13] builds from the implicit function a discrete volume around the surface of interest, and then computes mesh vertices on the edges of the volume; due to its discretization nature, mesh results of the algorithm are often short of sharp surface details. Algorithms similar to MC include marching tetrahedra (MT) [21] and dual contouring (DC) [14]. MT divides a voxel into six tetrahedrons and calculates the vertices on edges of each tetrahedron; DC utilizes gradients to estimate positions of vertices in a cell and extracts meshes from adaptive octrees. All these methods suffer from a trade-off of precision and efficiency, due to their necessity to sample discrete points from the 3D space.

**Local Linearity of MLPs.** Among works studying representational complexities of deep networks, Montúfar et al. [19] and Pascanu et al. [22] investigate how a ReLU or maxout based MLP partitions its input space into a number of linear regions, and bound this number via quantities relevant to network depth and width. The region-wise linear mapping is explicitly established in [23] in order to analyze generalization properties of deep networks. A closed-form solution termed OpenBox is proposed in [24] that computes consistent and exact interpretations for piecewise linear deep networks. The present work leverages the locally linear properties of ReLU based MLPs and studies how the zero-level isosurface can be extracted from such an MLP based implicit function.

## 1.2 Contributions

A preliminary version of this work appears in [25], where for the first time, we establish the analytic relations between an MLP based implicit function and its captured zero-level isosurface; we present in [25] a theorem that guarantees exact meshing from deep implicit surface networks, and a corresponding meshing algorithm. We re-state its technical contributions as follows.

- 1) Given that an MLP with ReLU activation partitions its input space into a number of linear regions, we identify from these regions *analytic cells* and *analytic faces* that are associated with zero-level isosurface of an implicit function constructed from such an MLP; we characterize the theoretical conditions under which the identified analytic faces are guaranteed to connect and form a *closed, piecewise planar surface*.
- 2) Based on the above analytic meshing theorem, we propose an algorithm of *analytic marching*, which marches among analytic cells to *exactly* recover the mesh captured by an implicit surface network. We empirically verify that the proposed meshing algorithm achieves a precision upper-bounding those achieved by existing algorithms.

In the present paper, we extend the theoretical analysis in [25] for more advanced MLP architectures, and contribute

techniques to improve the efficiency of analytic marching. These extensions enable us to apply our proposed method to learning and meshing novel and complex shape instances. In addition, we augment the paper presentation with motivation of theory and intuitive illustrations. We finally summarize our new contributions as follows.

- 1) We present analyses that make the analytic meshing theorem in [25] applicable to more advanced MLP architectures, including those with shortcut connections and max pooling operations. These extensions support a richer set of architectural designs for learning and exactly meshing complex surface shapes with analytic marching.
- 2) We contribute techniques to improve the efficiency of analytic marching, including *parallel marching* with CUDA implementation, efficient initialization schemes respectively customized for signed distance field and occupancy field, and mesh simplification for efficient downstream processing. Implementations of these techniques are included in *AnalyticMesh*, a software package accessible at <https://github.com/Karbo123/AnalyticMesh>.
- 3) We apply our method to different contexts of generative shape modeling using implicit surface networks; we consider both direct shape decoding of raw point observations, and learning to reconstruct novel shape instances using global or local shape decoders. Extensive experiments demonstrate the advantages of our meshing algorithm over existing ones in terms of both accuracy and efficiency.

## 2 PROBLEM STATEMENT AND MOTIVATION

This paper studies the fundamental problem of recovering an *explicit* representation  $\mathcal{Z}$  of an underlying surface  $\mathcal{M}$  from some, possibly learned, *implicit* surface function. We focus our surface of interest on those representing the boundary of a non-degenerate 3D solid whose nature is a continuous and closed 2-dimensional manifold embedded in the Euclidean space  $\mathbb{R}^3$ ; such a solid has no infinitely thin parts and its boundary surface properly separates the interior and exterior of the solid (cf. Fig. 1.1 in [2] for an illustration). Among choices of explicit surface representation, a polygon mesh is the most popular one defined as  $\mathcal{Z} = \{\mathcal{V}, \{\mathcal{P}\}\}$ , where  $\mathcal{V} = \{v \in \mathbb{R}^3\}$  contains the mesh vertices and  $\{\mathcal{P} \subset \mathbb{P}^2\}$  denotes the collection of connected polygon faces, each of which contains a coplanar set of vertices<sup>1</sup>. Any planar face thus defines an explicit mapping  $g_{\mathcal{P}} : \Omega \rightarrow \mathbb{R}^3$  from the domain  $\Omega$  (e.g.,  $\Omega \subset \mathbb{R}^2$ ) to a plane  $\mathbb{P}^2 \subset \mathbb{R}^3$ ; consequently, the mesh  $\mathcal{Z}$  becomes a piecewise linear approximation of the underlying  $\mathcal{M}$ . Recent results [26], [27], [28], [29] show that the collection  $\{g_{\mathcal{P}}\}$  of explicit mapping functions can be effectively learned as one or several deep networks, which are trained to generate a surface mesh via vertex deformation. However, topologies of the resulting meshes are restricted by those defined on

1. For simplicity, we omit edges in the definition of polygon mesh. Edges can be inferred as boundaries of polygon faces. In this work, we consider the non-degenerate case that any edge has no more than two incident faces and any vertex is incident to no more than one fan of faces.

the input domain  $\Omega$ ; queries of points inside or outside the surface are expensive as well.

As an alternative, one may resort to implicit surface representations, such as signed distance function (SDF) [4] or occupancy field [5], [6]. Implicit representations enjoy the benefits of modeling smooth and topologically complex surfaces. Let  $F : \mathbb{R}^3 \rightarrow \mathbb{R}$  denote a scalar-valued, implicit field function, and a surface is formally defined as its zero-level isosurface  $\{x \in \mathbb{R}^3 | F(x) = 0\}$ .<sup>2</sup> While  $F$  can be realized using radial basis functions [10], [11], [12] or be approximated as a regular grid of voxels (i.e., a volume), in this work, we are particularly interested in implementing  $F$  using deep networks, e.g., Multi-Layer Perceptrons (MLPs) with Rectified Linear Units [16], which become an increasingly popular choice in recent works of deep learning surface reconstruction [4], [5], [20]. These methods achieve the state-of-the-art performance in terms of surface modeling, and they typically take a final step of marching cubes [13] to recover the surface meshes, by sampling and evaluating a regular grid of discrete points in the 3D space. As stated in Section 1, the final step of marching cubes recovers a mesh that is only an approximation of the surface captured by  $F$ ; more specifically, it suffers from a trade-off of sampling efficiency and recovery precision, due to the discretization nature of the marching cubes algorithm.

In this work, we aim to address this limitation by developing a meshing algorithm whose nature is completely different from the discrete meshing family of marching cubes [13], [14], [21]. As stated above, most of existing deep implicit surface functions are based on MLPs. We note that a ReLU based MLP partitions its input space into a number of linear regions [19]; consequently, the zero-level isosurface  $\{x \in \mathbb{R}^3 | F(x) = 0\}$  of such a function  $F$  is embedded in the input space  $\mathbb{R}^3$  and is intersected by (some of) the partitioned linear regions. Given the *piecewise planar/linear*  $\{g_{\mathcal{P}}\}$  of a polygon mesh  $\mathcal{Z}$  and the *locally linear mappings* defined by an MLP based  $F$ , we are motivated to connect the local linearities of the two worlds and *analytically* identify the linear regions intersected by the zero-level isosurface; we expect these intersections to form  $\mathcal{Z}$  that is an *exact* meshing solution from  $F$ . In the present section, we give formal statement of the problem and our motivation. Fig. 1 illustrates the intuition.

### 2.1 Polygon Mesh as a Piecewise Linear Surface Representation

Assume we have a polygon mesh  $\mathcal{Z} = \{\mathcal{V}, \{\mathcal{P}\}\}$  embedded in the space  $\mathbb{R}^3$ . For any planar face  $\mathcal{P}$ , let  $\{v_i \in \mathcal{V} | i = 1, \dots, n_{\mathcal{P}}\}$  be its defining vertices. Given any three

2. Given an implicit field function  $F : \mathbb{R}^3 \rightarrow \mathbb{R}$ , the surface of interest is more precisely defined as

$$\{x \in \mathbb{R}^3 | F(x) = 0, |\nabla_x F(x)| \neq 0\}, \quad (1)$$

where the constraint  $|\nabla_x F(x)| \neq 0$  ensures that (1) indeed defines a zero-crossing isosurface. When  $F$  implements a signed distance function,  $|F(x)|$  measures the distance of any  $x \in \mathbb{R}^3$  to the surface, and by convention we have  $F(x) < 0$  for points inside the surface and  $F(x) > 0$  for those outside. When  $F$  represents an occupancy field, it implements a mapping  $\mathbb{R}^3 \rightarrow \{0, 1\}$ , which assigns each  $x \in \mathbb{R}^3$  a binary occupancy value indicating the exterior ( $F(x) = 0$ ) or interior ( $F(x) = 1$ ) status of  $x$ .

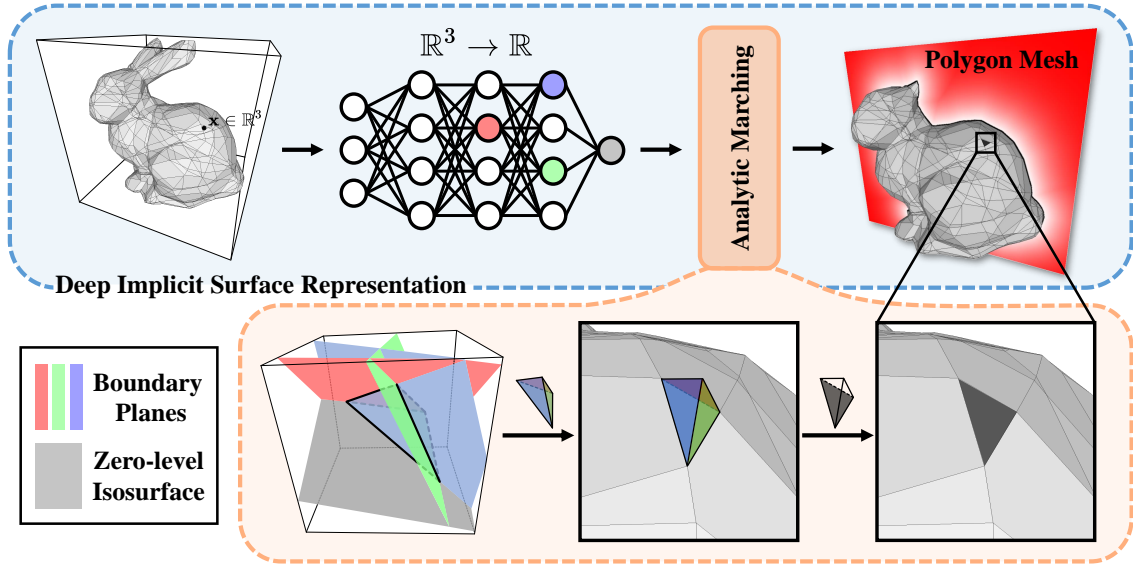


Fig. 1. An illustration on the intrinsic connection between a polygon mesh and the local linearities of its capturing deep implicit surface network. The polygon mesh is formed by the intersection between the zero-level isosurface of the implicit surface network and some of its partitioned linear regions in the input Euclidean space.

$\{v_i, v_j, v_k\}$  of these vertices, the plane on which the polygon segment  $\mathcal{P}$  resides can be written as

$$\mathbf{n}_{\mathcal{P}}^{\top}(\mathbf{x} - \mathbf{v}_i) = 0 \quad \text{s.t.} \quad \mathbf{n}_{\mathcal{P}} = \frac{(\mathbf{V}\mathbf{V}^{\top})^{-1}\mathbf{V} \cdot \mathbf{1}_3}{\|(\mathbf{V}\mathbf{V}^{\top})^{-1}\mathbf{V} \cdot \mathbf{1}_3\|_2}, \quad (2)$$

where the matrix  $\mathbf{V} = [v_i, v_j, v_k]$  collects coordinates of the three vertices,  $\mathbf{1}_3$  is a 3-dimensional vector with all its entries as the value of 1,  $\mathbf{n}_{\mathcal{P}} \in \mathbb{R}^3$  is the plane kernel or normal vector, and  $\mathbf{x} \in \mathbb{R}^3$  is any space point on the plane. The collection  $\{\mathbf{n}_{\mathcal{P}}\}$  thus gives a piecewise linear parameterization of  $\mathcal{Z}$ . We will show in the following that any ReLU based MLP  $F$  has its zero-level isosurface as polygon segments embedded in  $\mathbb{R}^3$ , which motivates a possible solution of analytic meshing from  $F$ .

## 2.2 The Local Linearity of Multi-Layer Perceptrons

We first discuss how a ReLU based MLP, as a nonlinear function, partitions its input space into linear regions via compositional structure. The discussion is put in a general form by assuming an MLP of  $L$  hidden layers that takes an input  $\mathbf{x} \in \mathbb{R}^{n_0}$  from the space  $\mathcal{X}$  and layer-wisely computes  $\mathbf{x}_l = \mathbf{g}(\mathbf{W}_l \mathbf{x}_{l-1})$ , where  $l \in \{1, \dots, L\}$  indexes the layer,  $\mathbf{x}_l \in \mathbb{R}^{n_l}$ ,  $\mathbf{x}_0 = \mathbf{x}$ ,  $\mathbf{W}_l \in \mathbb{R}^{n_l \times n_{l-1}}$ ,  $\mathbf{g}$  is the point-wise ReLU activation, and we omit the network biases for notational simplicity. We also denote the intermediate feature space  $\mathbf{g}(\mathbf{W}_l \mathbf{x}_{l-1})$  as  $\mathcal{X}_l$  and  $\mathcal{X}_0 = \mathcal{X}$ . In the context of present paper, we have  $\mathcal{X}_0 \subset \mathbb{R}^3$  and  $n_0 = 3$ .

The thus defined MLP can be compactly written as

$$\mathbf{T}\mathbf{x} = \mathbf{g}(\mathbf{W}_L \dots \mathbf{g}(\mathbf{W}_1 \mathbf{x})). \quad (3)$$

Any  $k^{\text{th}}$  neuron,  $k \in \{1, \dots, n_l\}$ , of an  $l^{\text{th}}$  layer of the MLP  $\mathbf{T}$  specifies a pre-activation functional defined as

$$a_{lk}(\mathbf{x}) = \pi_k \mathbf{W}_l \mathbf{g}(\mathbf{W}_{l-1} \dots \mathbf{g}(\mathbf{W}_1 \mathbf{x})),$$

where  $\pi_k$  denotes an operator that projects onto the  $k^{\text{th}}$  coordinate. All the neurons at layer  $l$  define a functional

$$\mathbf{a}_l(\mathbf{x}) = \mathbf{W}_l \mathbf{g}(\mathbf{W}_{l-1} \dots \mathbf{g}(\mathbf{W}_1 \mathbf{x})).$$

We define the support of  $\mathbf{T}$  as

$$\text{supp}(\mathbf{T}) = \{\mathbf{x} \in \mathcal{X} | \mathbf{T}\mathbf{x} \neq \mathbf{0}\}, \quad (4)$$

which are instances of practical interest in the input space  $\mathcal{X}$ .

For an intermediate feature space  $\mathcal{X}_{l-1} \in \mathbb{R}^{n_{l-1}}$ , each hidden neuron of layer  $l$  specifies a hyperplane  $\mathbf{H}$  that partitions  $\mathcal{X}_{l-1}$  into two halves, and the collection of hyperplanes  $\{\mathbf{H}_i\}_{i=1}^{n_l}$  specified by all the  $n_l$  neurons of layer  $l$  form a *hyperplane arrangement* [30]. These hyperplanes partition the space  $\mathcal{X}_{l-1}$  into multiple linear regions whose formal definition is as follows.

**Definition 1 (Region/Cell).** Let  $\mathcal{A}$  be an arrangement of hyperplanes in  $\mathbb{R}^m$ . A region of the arrangement is a connected component of the complement  $\mathbb{R}^m - \bigcup_{\mathbf{H} \in \mathcal{A}} \mathbf{H}$ . A region is a cell when it is bounded.

Classical result from [22], [31] shows that the arrangement of  $n_l$  hyperplanes gives at most  $\sum_{j=0}^{n_l-1} \binom{n_l}{j}$  regions in  $\mathbb{R}^{n_{l-1}}$ . Given fixed  $\{\mathbf{W}_l\}_{l=1}^L$ , the MLP  $\mathbf{T}$  partitions the input space  $\mathcal{X} \in \mathbb{R}^{n_0}$  by its layers' recursive partitioning of intermediate feature spaces, which can be intuitively understood as a successive process of space folding [19].

Let  $\mathcal{R}(\mathbf{T})$ , shortened as  $\mathcal{R}$ , denote the set of all linear regions/cells in  $\mathbb{R}^{n_0}$  that are possibly achieved by  $\mathbf{T}$ . To have a concept on the maximal size of  $\mathcal{R}$ , we introduce the following functionals about activation states of neuron, layer, and the whole MLP.

**Definition 2 (State of Neuron/MLP).** For a  $k^{\text{th}}$  neuron of an  $l^{\text{th}}$  layer of an MLP  $\mathbf{T}$ , with  $k \in \{1, \dots, n_l\}$  and  $l \in \{1, \dots, L\}$ , its state functional of neuron activation is defined as

$$s_{lk}(\mathbf{x}) = \begin{cases} 1 & \text{if } a_{lk}(\mathbf{x}) > 0 \\ 0 & \text{if } a_{lk}(\mathbf{x}) \leq 0, \end{cases} \quad (5)$$

3. Any instance  $\mathbf{x} \in \mathcal{X}$  nullified by an MLP  $\mathbf{T}$  of  $L$  hidden layers defined as (3) would be less useful for downstream tasks, e.g., an implicit function constructed from  $\mathbf{T}$ .



which gives the state functional of layer  $l$  as

$$\mathbf{s}_l(\mathbf{x}) = [s_{l1}(\mathbf{x}), \dots, s_{ln_l}(\mathbf{x})]^\top, \quad (6)$$

and the state functional of MLP  $\mathbf{T}$  as

$$\mathbf{s}(\mathbf{x}) = [\mathbf{s}_1(\mathbf{x})^\top, \dots, \mathbf{s}_L(\mathbf{x})^\top]^\top. \quad (7)$$

Let the total number of hidden neurons in  $\mathbf{T}$  be  $N = \sum_{l=1}^L n_l$ . Denote  $\mathbb{J} = \{1, 0\}$ , and we have the state functional  $\mathbf{s} \in \mathbb{J}^N$ . Considering that a region in  $\mathbb{R}^{n_0}$  corresponds to a realization of  $\mathbf{s} \in \mathbb{J}^N$ , it is clear that the maximal size of  $\mathcal{R}$  is upper bounded by  $2^N$ . This gives us the following labeling scheme.

- Any region  $r \in \mathcal{R}$  corresponds to a unique element in  $\mathbb{J}^N$ ; since  $\mathbf{s}(\mathbf{x})$  is fixed for all  $\mathbf{x} \in \mathcal{X}$  that fall in a same region  $r$ , we use  $\mathbf{s}(r) \in \mathbb{J}^N$  to label this region.

The following theorem from [19] gives a lower bound on the maximal size of  $\mathcal{R}$ .

**Theorem 3** ([19]). *For a ReLU based MLP  $\mathbf{T}$  of  $L$  hidden layers, whose layer widths satisfy  $n_l \geq n_0$  for any  $l \in \{1, \dots, L\}$ , the maximal size of  $\mathcal{R}(\mathbf{T})$  is lower bounded by  $\left(\prod_{l=1}^{L-1} \lfloor n_l/n_0 \rfloor^{n_0}\right) \sum_{j=0}^{n_0} \binom{n_L}{j}$ , where  $\lfloor \cdot \rfloor$  ignores the remainder. Assuming  $n_1 = \dots = n_L = n$ , the lower bound has an order of  $\mathcal{O}\left((n/n_0)^{(L-1)n_0} n^{n_0}\right)$ .*

The above theorem shows that the number of linear regions into which an MLP can partition the input space grows exponentially with the network depth and polynomially with the network width. We have the following lemma adapted from [23] to characterize the region-wise linear mappings.

**Lemma 4** (Linear Mapping of Region/Cell, an adaptation of Lemma 3.2 in [23]). *Given a ReLU based MLP  $\mathbf{T}$  of  $L$  hidden layers, for any region/cell  $r \in \mathcal{R}(\mathbf{T})$ , its associated linear mapping  $\mathbf{T}^r$  is defined as*

$$\mathbf{T}^r = \prod_{l=1}^L \mathbf{W}_l^r \quad (8)$$

$$\mathbf{W}_l^r = \text{diag}(\mathbf{s}_l(r)) \mathbf{W}_l, \quad (9)$$

where  $\text{diag}(\cdot)$  diagonalizes the state vector  $\mathbf{s}_l(r)$ .

Intuitively, the state vector  $\mathbf{s}_l$  in (9) selects a submatrix from  $\mathbf{W}_l$  by setting those inactive rows as zero.

### 2.3 Motivation to Connect the Local Linearities of the Two Worlds

We implement the implicit surface field function  $F$  by stacking on top of  $\mathbf{T}$  a regression function  $f : \mathbb{R}^{n_L} \rightarrow \mathbb{R}$ , giving rise to a functional

$$F(\mathbf{x}) = f \circ \mathbf{T}(\mathbf{x}) = \mathbf{w}_f^\top \mathbf{g}(\mathbf{W}_L \dots \mathbf{g}(\mathbf{W}_1 \mathbf{x})),$$

where  $\mathbf{w}_f \in \mathbb{R}^{n_L}$  is weight vector of the regressor. Since  $F$  is defined in  $\mathbb{R}^3$ , we have  $n_0 = 3$ . Given that the MLP  $\mathbf{T}$  partitions the input space  $\mathbb{R}^3$  into a set  $\mathcal{R}$  of linear regions, any region  $r \in \mathcal{R}$  satisfies  $\mathbf{x} \in \text{supp}(\mathbf{T}) \forall \mathbf{x} \in r$ , and can be uniquely indexed by its state vector  $\mathbf{s}(r)$  defined by (7). For such a region  $r$ , we have the following corollary from Lemma 4 that characterizes the associated linear mappings defined at neurons of  $\mathbf{T}$  and the final regressor.

**Corollary 5.** *Given an implicit surface field function  $F = f \circ \mathbf{T}$  built on a ReLU based MLP of  $L$  hidden layers, for any  $r \in \mathcal{R}(\mathbf{T})$ , the associated neuron-wise linear mappings and that of the final regressor are defined as*

$$\mathbf{a}_{lk}^r = \begin{cases} \pi_k \mathbf{W}_l \prod_{i=1}^{l-1} \mathbf{W}_i^r & \text{when } l > 1 \\ \pi_k \mathbf{W}_l & \text{when } l = 1 \end{cases} \quad (10)$$

$$\mathbf{a}_F^r = \mathbf{w}_f^\top \mathbf{T}^r = \mathbf{w}_f^\top \prod_{i=1}^L \mathbf{W}_i^r, \quad (11)$$

where  $l \in \{1, \dots, L\}$ ,  $k \in \{1, \dots, n_l\}$ , and  $\mathbf{T}^r$  and  $\mathbf{W}_i^r$  are defined as in Lemma 4 (equations (8) and (9) for  $\mathbf{T}^r$  and  $\mathbf{W}_i^r$ , respectively).

Assume that the implicit function  $F$  models a continuous and closed 2-dimensional surface manifold embedded in  $\mathbb{R}^3$ . Our problem of interest is to recover an explicit mesh  $\mathcal{Z} = \{\mathcal{V}, \{\mathcal{P}\}\}$  from  $F$ , by extracting its zero-level isosurface  $\{\mathbf{x} \in \mathbb{R}^3 | F(\mathbf{x}) = 0\}$ . Section 2.1 shows that  $\mathcal{Z}$  has the piecewise linear parameterization of  $\{\mathbf{n}_{\mathcal{P}} \in \mathbb{R}^3\}$ . On the other hand, Corollary 5 suggests that  $F = f \circ \mathbf{T}$  in fact implements a piecewise linear function defined by the collection  $\{\mathbf{a}_F^r \in \mathbb{R}^3 | r \in \mathcal{R}\}$ ; consequently, the zero-level isosurface  $\{\mathbf{x} \in \mathbb{R}^3 | F(\mathbf{x}) = 0\}$  can either be locally linear with polygon faces obtained by intersection with some of the linear regions  $\{r \in \mathcal{R}\}$ , as illustrated in Fig. 1, or in some special case coincide with hyperplane boundaries of some linear regions (detailed explanations are given in Section 3). We are interested in the former case and expect that the parameterization  $\{\mathbf{n}_{\mathcal{P}}\}$  of  $\mathcal{Z}$  can be analytically identified as some of the liner mappings  $\{\mathbf{a}_F^r | r \in \mathcal{R}\}$ . We prove in Section 3 that this is indeed the case, and present an efficient algorithm of exactly meshing  $\mathcal{Z}$  from  $F$  in Section 4, where we also show that  $\mathbf{T}$  can be extended to incorporate shortcut connections and max pooling, which supports advanced architectures of  $\mathbf{T}$ . Details are presented as follows.

## 3 ANALYTIC MESHING FROM DEEP IMPLICIT SURFACE NETWORKS

### 3.1 Analytic Cells and Analytic Faces Associated with a Deep Implicit Surface Network

Corollary 5 is useful to specify linear regions in  $\mathcal{R}$  and the zero-level isosurface of the implicit function  $F$ . For any  $r \in \mathcal{R}$ , its boundary planes must be among the set

$$\{\mathbf{H}_{lk}^r\} \text{ s.t. } \mathbf{H}_{lk}^r = \{\mathbf{x} \in \mathbb{R}^3 | \mathbf{a}_{lk}^r \mathbf{x} = 0\}, \quad (12)$$

where  $l = 1, \dots, L$  and  $k = 1, \dots, n_l$ . The zero-level isosurface of  $F$  in fact induces a set of region-associated planes in  $\mathbb{R}^3$ ; the induced plane  $\{\mathbf{x} \in \mathbb{R}^3 | \mathbf{a}_F^r \mathbf{x} = 0\}$  and the associated region  $r$  have the following relations, assuming that the plane does not happen to coincide with any boundary plane of  $\{\mathbf{H}_{lk}^r\}$ . For simplicity, we use the plane kernel/normal vector  $\mathbf{a}_F^r$  to represent the region-associated plane induced by the zero-level isosurface of  $F$ .

- Intersection  $\mathbf{a}_F^r$  splits the region  $r$  into two halves, denoted as  $r^+$  and  $r^-$ , such that  $\forall \mathbf{x} \in r^+$ , we have  $\mathbf{a}_F^r \mathbf{x} > 0$  and  $\forall \mathbf{x} \in r^-$ , we have  $\mathbf{a}_F^r \mathbf{x} \leq 0$ .

- *Non-intersection* We either have  $\mathbf{a}_F^r \mathbf{x} > 0$  or  $\mathbf{a}_F^r \mathbf{x} < 0$  for all  $\mathbf{x} \in r$ .

Let  $\{\tilde{r} \in \tilde{\mathcal{R}}\}$  denote the subset of regions in  $\mathcal{R}$  that have the above relation of intersection. It is clear that the zero-level isosurface  $\{\mathbf{x} \in \mathbb{R}^3 | F(\mathbf{x}) = 0\}$  defined on the support (4) of  $\mathbf{T}$  can be only in  $\tilde{\mathcal{R}}$ . Consider such a region  $\tilde{r} \in \tilde{\mathcal{R}}$ ; for any  $\mathbf{x} \in \tilde{r}$ , it must satisfy  $(2s_{lk}(\tilde{r}) - 1)\mathbf{a}_{lk}^{\tilde{r}} \mathbf{x} \geq 0$ , which gives the following system of inequalities

$$(\mathbf{I} - 2\text{diag}(\mathbf{s}(\tilde{r}))\mathbf{A}^{\tilde{r}})\mathbf{x} = \begin{bmatrix} (1 - 2s_{11}(\tilde{r}))\mathbf{a}_{11}^{\tilde{r}} \\ \vdots \\ (1 - 2s_{lk}(\tilde{r}))\mathbf{a}_{lk}^{\tilde{r}} \\ \vdots \\ (1 - 2s_{Ln_L}(\tilde{r}))\mathbf{a}_{Ln_L}^{\tilde{r}} \end{bmatrix} \mathbf{x} \preceq 0, \quad (13)$$

where  $\mathbf{I}$  is an identity matrix of compatible size,  $\mathbf{A}^{\tilde{r}} \in \mathbb{R}^{N \times 3}$  collects the coefficients of the  $N = \sum_{l=1}^L n_l$  inequalities, and the state functionals  $s_{lk}$  and  $\mathbf{s}$  are defined by (5) and (7). When the region is bounded, the system (13) of inequalities essentially forms a convex polyhedral cell defined as

$$\mathcal{C}_F^{\tilde{r}} = \{\mathbf{x} \in \mathbb{R}^3 | (\mathbf{I} - 2\text{diag}(\mathbf{s}(\tilde{r}))\mathbf{A}^{\tilde{r}})\mathbf{x} \preceq 0\}, \quad (14)$$

which we term as *analytic cell of an implicit function's zero-level isosurface*, shortened as *analytic cell*. We note that there could exist redundancy in the defining inequalities of (13); an analytic cell could also be a region open towards infinity in some directions.

Given the plane functional (11), we define the polygon face that is an intersection of analytic cell  $\tilde{r}$  and zero-level isosurface of  $F$  as

$$\mathcal{P}_F^{\tilde{r}} = \{\mathbf{x} \in \mathbb{R}^3 | \mathbf{a}_F^{\tilde{r}} \mathbf{x} = 0, (\mathbf{I} - 2\text{diag}(\mathbf{s}(\tilde{r}))\mathbf{A}^{\tilde{r}})\mathbf{x} \preceq 0\}, \quad (15)$$

which we term less precisely as *analytic face of an implicit function's zero-level isosurface*, shortened as *analytic face*, since it is possible that the face goes towards infinity in some directions. With the analytic form (15), we realize that a ReLU based MLP  $F$  defines a piecewise planar zero-level isosurface, which could be an approximation to an underlying surface  $\mathcal{M}$  when  $F$  is trained using techniques presented in Section 6.

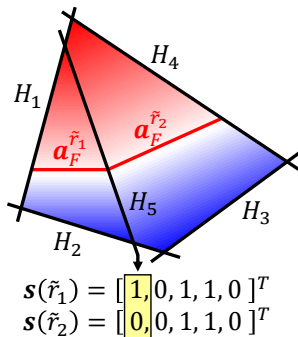


Fig. 2. An 2D illustration for analytic cells and neural states. Red lines are the analytic faces. The states of two neighboring cells that share a boundary plane are switched at one neuron.

### 3.2 A Closed Mesh via Connected Analytic Faces in Analytic Cells

We have stated in Section 2 that our problem of interest is to recover a surface mesh  $\mathcal{Z}$  from the implicit function  $F = f \circ \mathbf{T}$ , and the surface of interest has the property of being continuous and closed. A closed, piecewise planar mesh  $\mathcal{Z}$  means that every of its planar faces is connected with other faces via shared edges. Analysis in the preceding section shows that the zero-level isosurface of  $F$  is piecewise planar whose associated analytic cells and analytic faces respectively satisfy (14) and (15). We have the following theorem that characterizes the conditions under which analytic faces (15) in their respective analytic cells (14) guarantee to connect and form a closed, piecewise planar  $\mathcal{Z}$ .

**Theorem 6.** Assume that the zero-level isosurface  $\mathcal{Z}$  of an implicit field function  $F = f \circ \mathbf{T}$  defines a closed surface. If for any region/cell  $r \in \mathcal{R}(\mathbf{T})$ , its associated linear mapping  $\mathbf{T}^r$  (8) and the induced plane  $\mathbf{a}_F^r = \mathbf{w}_f^\top \mathbf{T}^r$  (11) are uniquely defined, i.e.,  $\mathbf{T}^r \neq \beta \mathbf{T}^{r'}$  and  $\mathbf{a}_F^r \neq \beta \mathbf{a}_F^{r'}$  for any region pair of  $r$  and  $r'$ , where  $\beta$  is an arbitrary scaling, then analytic faces  $\{\mathcal{P}_F^{\tilde{r}}\}$  defined by (15) connect and exactly form the surface  $\mathcal{Z}$  of polygon mesh.

*Proof.* The proof is given in Appendix A. Given the assumed conditions, the proof can be sketched by first showing that each planar face on  $\mathcal{Z}$  captured by the SDF  $F = f \circ \mathbf{T}$  uniquely corresponds to an analytic face of an analytic cell, and then showing that for any pair of planar faces connected on  $\mathcal{Z}$ , their corresponding analytic faces are connected via boundaries of their respective analytic cells.  $\square$

We note that the conditions assumed in Theorem 6 can be practically met up to a numerical precision of the weights in  $F = f \circ \mathbf{T}$ . The proof also suggests an algorithm to identify the analytic polygon faces of the surface captured by  $F$ , which is to be presented in Section 4.

## 4 PRACTICAL AND EFFICIENT IMPLEMENTATIONS OF THE ANALYTIC MARCHING ALGORITHM

In this section, we first present our proposed algorithm of *analytic marching* for extraction of the piecewise planar, zero-level isosurface captured by an implicit  $F = f \circ \mathbf{T}$ ; we then present its efficient implementations, including the CUDA version on GPUs, and customized strategies for triggering our algorithm when  $F$  respectively represents an SDF or occupancy field. We finally discuss how the mesh obtained by analytic marching can be simplified, with least sacrifice of precision, to support efficient downstream processing.

### 4.1 The Algorithm of Analytic Marching

Given an implicit function  $F = f \circ \mathbf{T}$  whose zero-level isosurface  $\mathcal{Z} = \{\mathbf{x} \in \mathbb{R}^3 | F(\mathbf{x}) = 0\}$  defines a closed, piecewise planar surface, Theorem 6 suggests that obtaining the mesh  $\mathcal{Z}$  concerns with identification of analytic faces  $\{\mathcal{P}_F^{\tilde{r}} | \tilde{r} \in \tilde{\mathcal{R}}\}$  in analytic cells  $\{\mathcal{C}_F^{\tilde{r}} | \tilde{r} \in \tilde{\mathcal{R}}\}$ . To this end, we propose an algorithm of *analytic marching* that marches among  $\{\mathcal{C}_F^{\tilde{r}} | \tilde{r} \in \tilde{\mathcal{R}}\}$  to identify vertices and edges of the polygon faces, where the name is indeed to show respect to the classical discrete algorithm of marching cubes [13].

Specifically, analytic marching is triggered by identifying at least one point  $\mathbf{x} \in \mathcal{Z}$  that satisfies  $F(\mathbf{x}) = 0$ ; its state vector  $\mathbf{s}(\mathbf{x})$  can be computed via (7), which specifies the analytic cell  $\mathcal{C}_F^{\tilde{\mathbf{x}}}$  (14) and analytic face  $\mathcal{P}_F^{\tilde{\mathbf{x}}}$  (15) where  $\mathbf{x}$  resides. Analytic marching then successively solves a system of equations to analytically obtain vertices of the polygon face inside each analytic cell, and marches to neighboring analytic cells via transition of cell states. Details are given in Algorithm 1, with an illustration shown in Fig. 2.

---

**Algorithm 1** The algorithm of Analytic Marching

---

**INPUT:** An implicit surface field function  $F = f \circ \mathbf{T}$  constructed from a ReLU based MLP

**OUTPUT:** The exact zero-level isosurface of  $F$  as a polygon mesh  $\mathcal{Z} = \{\mathcal{V}, \{\mathcal{P}\}\}$

---

- 1: Initialize an active set  $\mathcal{S}^\bullet = \emptyset$ , and an inactive set  $\mathcal{S}^\circ = \emptyset$ .
  - 2: Identify one point  $\mathbf{x} \in \mathbb{R}^3$  that satisfies  $F(\mathbf{x}) = 0$  (and  $\nabla_{\mathbf{x}} F(\mathbf{x}) \neq \mathbf{0}$ ).
  - 3: Compute the state  $\mathbf{s}(\mathbf{x})$  via (7), and push  $\mathbf{s}(\mathbf{x})$  into  $\mathcal{S}^\bullet$ .
  - 4: **while**  $\mathcal{S}^\bullet \neq \emptyset$  **do**
  - 5:   Take an active state  $\mathbf{s}(\tilde{\mathbf{r}})$  from  $\mathcal{S}^\bullet$ .
  - 6:   Let  $\mathcal{V}_{\tilde{\mathcal{P}}}^{\tilde{\mathbf{r}}}$  be the set of defining vertices for the polygon face  $\mathcal{P}_F^{\tilde{\mathbf{r}}}$ ; enumerate all the pair  $(\mathbf{H}_{l_k}^{\tilde{\mathbf{r}}}, \mathbf{H}_{l'_k}^{\tilde{\mathbf{r}}})$  of boundary planes  $\{\mathbf{H}_{l_k}^{\tilde{\mathbf{r}}}\}$  defined by (12), with  $l = 1, \dots, L$  and  $k = 1, \dots, n_l$ .
  - 7:   For each pair  $(\mathbf{H}_{l_k}^{\tilde{\mathbf{r}}}, \mathbf{H}_{l'_k}^{\tilde{\mathbf{r}}})$ , together with  $\mathcal{P}_F^{\tilde{\mathbf{r}}}$ , solve the following  $3 \times 3$  system of equations to have a  $\mathbf{v} \in \mathbb{R}^3$ 

$$[\mathbf{a}_{l_k}^{\tilde{\mathbf{r}}}; \mathbf{a}_{l'_k}^{\tilde{\mathbf{r}}}; \mathbf{a}_F^{\tilde{\mathbf{r}}}] \mathbf{x} = \mathbf{0}. \quad (16)$$
  - 8:   Confirm the validity of  $\mathbf{v} \in \mathcal{V}_{\tilde{\mathcal{P}}}^{\tilde{\mathbf{r}}}$  when it satisfies the boundary condition (13) of the cell  $\mathcal{C}_F^{\tilde{\mathbf{r}}}$ .
  - 9:   Form  $\mathcal{V}_{\tilde{\mathcal{P}}}^{\tilde{\mathbf{r}}}$  of the face  $\mathcal{P}_F^{\tilde{\mathbf{r}}}$  with all the valid vertices obtained by solving (16); push vertices of  $\mathcal{V}_{\tilde{\mathcal{P}}}^{\tilde{\mathbf{r}}}$  into  $\mathcal{V}$ , and the face  $\mathcal{P}_F^{\tilde{\mathbf{r}}}$  into  $\mathcal{Z}$ .
  - 10:   Record all the boundary planes  $\{\widehat{\mathbf{H}}_{l_k}^{\tilde{\mathbf{r}}}\}$  of  $\mathcal{C}_F^{\tilde{\mathbf{r}}}$  that give valid vertices; for each  $\widehat{\mathbf{H}}_{l_k}^{\tilde{\mathbf{r}}}$ , infer the state  $\mathbf{s}(\tilde{\mathbf{r}}_{\text{Neighbor}})$  of a neighboring analytic cell by switching  $\mathbf{s}(\tilde{\mathbf{r}})$  at  $\mathbf{s}(\tilde{\mathbf{r}}_{l_k})$ .
  - 11:   Push  $\mathbf{s}(\tilde{\mathbf{r}})$  out of the active set  $\mathcal{S}^\bullet$  and into the inactive set  $\mathcal{S}^\circ$ ; push  $\{\mathbf{s}(\tilde{\mathbf{r}}_{\text{Neighbor}}) | \mathbf{s}(\tilde{\mathbf{r}}_{\text{Neighbor}}) \notin \mathcal{S}^\circ\}$  activated in the preceding step into the active set  $\mathcal{S}^\bullet$ .
  - 12: **end while**
- 

Given  $F$  and an arbitrary space point, the zero-crossing point  $\mathbf{x}$  in Step 2 can be obtained simply by solving the following problem via stochastic gradient descent (SGD)

$$\min_{\mathbf{x} \in \mathbb{R}^3} |F(\mathbf{x})|. \quad (17)$$

In practice, it is not necessary for the obtained  $\mathbf{x}$  to exactly satisfy  $F(\mathbf{x}) = 0$ ; the algorithm works as long as  $F(\mathbf{x})$  is sufficiently small that ensures  $\mathbf{x}$  falls in an analytic cell.

**Algorithmic guarantee** Theorem 6 guarantees that when the zero-level isosurface  $\mathcal{Z}$  of the implicit function  $F = f \circ \mathbf{T}$  is closed, identification of all the analytic faces forms the

closed surface mesh. The proposed analytic marching algorithm is anchored at cell state transition whose success is of high probability due to a phenomenon similar to the blessing of dimensionality [32] — it is of low probability that edges connecting planar faces of  $\mathcal{Z}$  coincide with edges of analytic cells (cf. proof of Theorem 6 for detailed analysis).

#### 4.1.1 Analysis of Computational Complexity

Consider the implicit function  $F = f \circ \mathbf{T}$  built on an MLP of  $L$  hidden layers, each of which has  $n_l$  neurons,  $l = 1, \dots, L$ . Let  $N = n_1 + \dots, n_L$ . For ease of analysis, we assume  $n_1 = \dots = n_L = n$  and thus  $N = nL$ . The computations inside each analytic cell concern with computing the boundary planes, solving a maximal number of  $\binom{N}{2}$  equation systems (16), and checking the validity of resulting vertices, which give a complexity order of  $\mathcal{O}(n^3 L^3)$ . We know from [19] that the maximal size of the set  $\mathcal{R}(\mathbf{T})$  of linear regions in general has an order of  $\mathcal{O}\left((n/n_0)^{(L-1)n_0} n^{n_0}\right)$ , where  $n_0$  is the dimensionality of input space. Since our focus of interest is the 2-dimensional surface embedded in the 3D space, we have  $n_0 = 2$  and thus the maximal size of  $\mathcal{R}(\mathbf{T})$ , which bounds the maximal number of analytic cells, in general has an order of  $\mathcal{O}\left((n/2)^{2(L-1)} n^2\right)$ . Overall, the complexity of our analytic marching algorithm has an order of  $\mathcal{O}\left((n/2)^{2(L-1)} n^5 L^3\right)$ , which is exponential w.r.t. the MLP depth  $L$  and polynomial w.r.t. the MLP width  $n$ . Note that the result can be further improved by applying the efficient implementations to be presented in Section 4.2.

The above analysis shows that the complexity nature of analytic marching is the complexity of implicit function, which is completely different from those of existing algorithms, such as marching cubes [13], whose complexities are irrelevant to function complexities but rather depend on the discretized resolutions of the 3D space. Our algorithm thus provides an opportunity to recover highly precise mesh reconstruction by using networks of low complexities.

## 4.2 Efficient Implementations

### 4.2.1 Pivoting Enumeration in a Working Cell

Steps 6 - 11 in Algorithm 1 involve enumeration of all the pairs from the boundary planes  $\{\mathbf{H}_{l_k}^{\tilde{\mathbf{r}}}\}$ , with  $l = 1, \dots, L$  and  $k = 1, \dots, n_l$ , in order to find the valid vertices  $\{\mathbf{v} \in \mathcal{V}_{\tilde{\mathcal{P}}}^{\tilde{\mathbf{r}}}\}$  for the polygon face  $\mathcal{P}_F^{\tilde{\mathbf{r}}}$  in the working cell  $\mathcal{C}_F^{\tilde{\mathbf{r}}}$ . The process is less efficient and requires a postprocessing step to identify the traversal order of valid vertices. To improve the efficiency, we present a pivoting-based enumeration scheme [33] whose details are as follows.

Pivoting enumeration starts with identifying a point  $\mathbf{x} \in \mathcal{P}_F^{\tilde{\mathbf{r}}}$  and a hyperplane  $\mathbf{H}_{l_k}^{\tilde{\mathbf{r}}}$  among  $\{\mathbf{H}_{l_k}^{\tilde{\mathbf{r}}}\}$ , that is a true boundary of the working cell  $\mathcal{C}_F^{\tilde{\mathbf{r}}}$ ; we present how such an  $\mathbf{x}$  and  $\mathbf{H}_{l_k}^{\tilde{\mathbf{r}}}$  can be identified shortly. The scheme then establishes an index set  $\mathcal{I}^{\tilde{\mathbf{r}}} = \{(l_1, k_1), \dots, (l_N, k_N)\}$  by sorting, in an increasing order, the Euclidean distances from each of the  $N$  hyperplanes in  $\{\mathbf{H}_{l_k}^{\tilde{\mathbf{r}}}\}$  to the point  $\mathbf{x} \in \mathcal{P}_F^{\tilde{\mathbf{r}}}$ , where  $N = n_1 + \dots, n_L$  is the total number of neurons in  $\mathbf{T}$ . Introduce the auxiliary  $t, t', t'' \in \mathbb{N}$ , and we use  $(l_t, k_t)$  to index the boundary plane  $\mathbf{H}_{l_t k_t}^{\tilde{\mathbf{r}}}$  identified in the beginning, written as  $\mathbf{H}_{l_t k_t}^{\tilde{\mathbf{r}}}$ ; for a later reference, we also use  $(l_*, k_*)$  and  $\mathbf{H}_{l_* k_*}^{\tilde{\mathbf{r}}}$  to refer to this same boundary. Let

$\mathcal{I}_{/t}^{\tilde{r}} = \{(l_1, k_1), \dots, (l_{t-1}, k_{t-1}), (l_{t+1}, k_{t+1}), \dots, (l_N, k_N)\}$ , and we have  $|\mathcal{I}_{/t}^{\tilde{r}}| = N - 1$ . By initializing  $\mathcal{V}_{\tilde{P}}^{\tilde{r}} = \emptyset$  and  $t'' = 1$ , the scheme firstly repeats the following steps: 1) solves the system of equations  $[\mathbf{a}_{l_t k_t}^{\tilde{r}}; \mathbf{a}_{l_{t''} k_{t''}}^{\tilde{r}}; \mathbf{a}_F^{\tilde{r}}] \mathbf{x} = \mathbf{0}$  to have a vertex candidate  $\mathbf{v} \in \mathbb{R}^3$ ; 2) confirms the validity of  $\mathbf{v} \in \mathcal{V}_{\tilde{P}}^{\tilde{r}}$  by checking whether it satisfies the boundary condition (13) of the cell  $\mathcal{C}_F^{\tilde{r}}$ ; 3) if the above step is true, pushes  $\mathbf{v}$  into  $\mathcal{V}_{\tilde{P}}^{\tilde{r}}$ , lets  $(l_{t'}, k_{t'}) = (l_t, k_t)$  and updates  $(l_t, k_t) = (l_{t'}, k_{t'})$ , and then *exits*; otherwise updates  $t'' \leftarrow t'' + 1$  upon  $(l_{t''+1}, k_{t''+1}) \in \mathcal{I}_{/t}^{\tilde{r}}$  or  $t'' \leftarrow t'' + 2$  upon  $(l_{t''+2}, k_{t''+2}) \in \mathcal{I}_{/t}^{\tilde{r}}$ , and goes back to step 1. Given that  $\mathbf{H}_{l_t k_t}^{\tilde{r}}$  is a true boundary of the polyhedral cell  $\mathcal{C}_F^{\tilde{r}}$ , it is guaranteed for the above steps to find a valid  $\mathbf{v} \in \mathcal{V}_{\tilde{P}}^{\tilde{r}}$ . Let  $\mathcal{I}_{/t'}^{\tilde{r}} = \{(l_1, k_1), \dots, (l_{t'-1}, k_{t'-1}), (l_{t'+1}, k_{t'+1}), \dots, (l_{t-1}, k_{t-1}), (l_{t+1}, k_{t+1}), \dots, (l_N, k_N)\}$  and we have  $|\mathcal{I}_{/t'}^{\tilde{r}}| = N - 2$ , where we assume  $t' < t$  for notational simplicity, the scheme then repeats the following steps: 1) solves the system of equations  $[\mathbf{a}_{l_t k_t}^{\tilde{r}}; \mathbf{a}_{l_{t'} k_{t'}}^{\tilde{r}}; \mathbf{a}_F^{\tilde{r}}] \mathbf{x} = \mathbf{0}$  to have a vertex candidate  $\mathbf{v} \in \mathbb{R}^3$ ; 2) confirms the validity of  $\mathbf{v} \in \mathcal{V}_{\tilde{P}}^{\tilde{r}}$  by checking whether it satisfies the boundary condition (13) of the cell  $\mathcal{C}_F^{\tilde{r}}$ ; 3) if the above step is true, pushes  $\mathbf{v}$  into  $\mathcal{V}_{\tilde{P}}^{\tilde{r}}$ , and either *exits* when  $(l_{t'}, k_{t'})$  indexes the same boundary plane as the starting one  $\mathbf{H}_{l_t k_t}^{\tilde{r}}$ , or *sequentially* updates  $(l_{t'}, k_{t'}) = (l_t, k_t)$ ,  $(l_t, k_t) = (l_{t'}, k_{t'})$ , and  $\mathcal{I}_{/t'}^{\tilde{r}} = \{(l_1, k_1), \dots, (l_{t'-1}, k_{t'-1}), (l_{t'+1}, k_{t'+1}), \dots, (l_{t-1}, k_{t-1}), (l_{t+1}, k_{t+1}), \dots, (l_N, k_N)\}$ , and (re-)sets  $t'' = 1$ ; otherwise updates  $t'' \leftarrow t'' + 1$  upon  $(l_{t''+1}, k_{t''+1}) \in \mathcal{I}_{/t'}^{\tilde{r}}$  or  $t'' \leftarrow t'' + 2$  upon  $(l_{t''+2}, k_{t''+2}) \in \mathcal{I}_{/t'}^{\tilde{r}}$ , and goes back to step 1. The ending condition in the above step 3 suggests that the scheme has circled back and found all the valid vertices  $\{\mathbf{v} \in \mathcal{V}_{\tilde{P}}^{\tilde{r}}\}$  that form the polygon face  $\mathcal{P}_F^{\tilde{r}}$ . We summarize the algorithm in Appendix B.

Sorting planes according to Euclidean distances reduces the number of iterations required to find the valid vertices. We note that acquiring the starting boundary plane  $\mathbf{H}_{l_t k_t}^{\tilde{r}}$  and point  $\mathbf{x} \in \mathcal{P}_F^{\tilde{r}}$  does not require any extra effort;  $\mathbf{H}_{l_t k_t}^{\tilde{r}}$  can be set exactly as the switching plane that gives the current cell state (cf. Step 10 of Algorithm 1), and  $\mathbf{x}$  can be set as the middle point of the switching edge on the switching plane. The presented pivoting enumeration has an average-case complexity of  $\mathcal{O}(|\mathcal{V}_{\tilde{P}}|n^2L^2)$ , where  $|\mathcal{V}_{\tilde{P}}|$  is the average number of vertices per analytic face; it reduces the overall complexity of analytic marching to an order of  $\mathcal{O}\left((n/2)^{2(L-1)}|\mathcal{V}_{\tilde{P}}|n^4L^2\right)$ .

#### 4.2.2 Parallel Marching with CUDA Implementation

Our proposed analytic marching naturally supports parallel implementation. Instead of starting from a single initial point  $\mathbf{x} \in \mathcal{Z}$  obtained by solving (17) (Step 2 of Algorithm 1), one may initialize as many of such points as possible in parallel; the algorithm can then be fully parallelized by simultaneously marching towards all the analytic cells in the active set  $\mathcal{S}^*$  that are unsolved to get their respectively analytic faces. *Parallel marching* would improve the efficiency of analytic marching significantly.

Parallel marching can be practically achieved on parallel computing devices (e.g. GPUs). As a contribution to the community, we present a CUDA implementation of

analytic marching algorithm to support parallel marching. The implementation is incorporated in the package of `AnalyticMesh` publicly available at <https://github.com/Karbo123/AnalyticMesh>. With CUDA implementation on Nvidia GPUs, the efficiency of analytic marching can be improved at an order of 10. We report empirical running time comparisons in Section 7.1.

### 4.3 Customized Schemes for Triggering the Algorithm

The algorithm of analytic marching can be triggered by solving (17) via SGD to find  $\mathbf{x} \in \mathcal{Z}$ . Depending on whether the implicit function  $F$  is an SDF or an occupancy field (OF) function, one may leverage the defining properties of the respective fields to have customized triggering strategies. In this section, we present two such schemes respectively specialized for SDF and OF. Empirical results in Section 7.1 confirm the efficiency of the two schemes when compared with triggering by solving (17) via SGD.

#### 4.3.1 Sphere Tracing-based Triggering for SDF

By utilizing the property that SDF satisfies the Eikonal equation  $|\nabla_{\mathbf{x}} F| = 1$ , sphere tracing [34] greatly accelerates the rendering of implicit surface. This inspires us to have an efficient sphere tracing-based scheme to find  $\mathbf{x} \in \mathcal{Z}$  for SDF. More specifically, let  $\mathbf{x}_0 \in \mathbb{R}^3$  denote an arbitrary initial point in the implicit field of SDF  $F$ ; given an  $\mathbf{x}_t \in \mathbb{R}^3$  at a time step  $t$ , we use the following updating rule to have  $\mathbf{x}_{t+1}$

$$\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \eta F(\mathbf{x}_t) \nabla_{\mathbf{x}_t} F, \quad (18)$$

where  $\eta \leq 1$  is the step size. Given that  $F(\mathbf{x}_t)$  computes the signed distance to the surface  $\mathcal{Z}$  at  $\mathbf{x}_t$ , the rule (18) thus accelerates standard SGD updating when  $\mathbf{x}_t$  is far away from  $\mathcal{Z}$ , and it has a damping effect to prevent surface penetration when  $\mathbf{x}_t$  moves very close to  $\mathcal{Z}$ .

#### 4.3.2 Dichotomy-based Triggering for Occupancy Field

A binary OF is usually relaxed by using a sigmoid function  $f$  to construct  $F = f \circ T$ . As such, gradient-based optimization (e.g. SGD) is less effective to find  $\mathbf{x} \in \mathcal{Z}$  from an arbitrary field point, especially when the initial point is far away from the surface. Fortunately, Bolzano's theorem [35] states that a continuous function has a root in an interval if it has values of opposite signs inside that interval. Based on this, we propose a dichotomy-based scheme to find  $\mathbf{x} \in \mathcal{Z}$  for OF. More specifically, we first randomly sample seed points in the implicit OF until a pair  $\{\mathbf{x}_0^+, \mathbf{x}_0^-\}$  is obtained which satisfies  $F(\mathbf{x}_0^+) > 0$  and  $F(\mathbf{x}_0^-) < 0$ . According to Bolzano's theorem, we can assert that there must be at least one zero-crossing point  $\mathbf{x}$  satisfying  $F(\mathbf{x}) = 0$  on the line segment  $\mathbf{x}_0^+ - \mathbf{x}_0^-$ . To find a zero-crossing point, we repeatedly bisect the line segment, and then select the subinterval at an iteration  $t$  whose two end points  $\mathbf{x}_t^+$  and  $\mathbf{x}_t^-$  have opposite signs of occupancy. The process continues until  $F(\mathbf{x}_t^+) - F(\mathbf{x}_t^-) \leq \epsilon$  for a specified tolerance  $\epsilon$ .

### 4.4 Postprocessing for Mesh Simplification

The polygon mesh  $\mathcal{Z}$  obtained by analytic marching is an exact solution of the zero-level isosurface of an implicit function  $F = f \circ T$ . When the network  $T$  is large, there

would be a huge number of polygon faces in the obtained  $\mathcal{Z}$  (cf. Table 5 for a reference of the number of faces practically obtained). This would bring inconvenience for subsequent processing on mesh, e.g., rendering and texturing. To reduce the number of faces with least sacrifice of mesh precision, we adopt the quadric edge collapse decimation (QECD) algorithm [36] to simplify the obtained  $\mathcal{Z}$ ; QECD estimates the placement of vertices of collapsed edges by minimizing the distances to neighboring faces, and is hence able to preserve sharp features of the original  $\mathcal{Z}$ . In case that the implicit function  $F = f \circ T$  itself captures a less ideal surface  $\mathcal{Z}$ , e.g., a rugged or non-watertight surface, our analytic marching would also produce the exact but less desirable  $\mathcal{Z}$ . One could apply postprocessing steps, such as smoothing [37] or holes filling [38], to improve the visual quality. We also note that our obtained polygon meshes can be easily converted as triangular ones, simply by subdividing polygons into triangles along diagonals.

We incorporate the above postprocessing operations into the publicly released package *AnalyticMesh*, in which we also provide a handle to control the number of polygon faces. Fig. 3 shows the interface. Experiments in Section 7 confirm that the proposed postprocessing is able to effectively simplify the meshes obtained by analytic marching; in many cases, it produces visually more pleasant results.

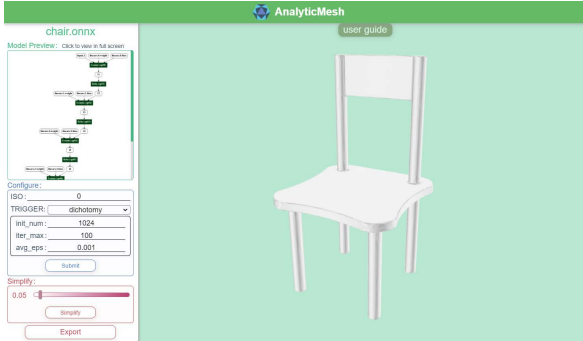


Fig. 3. The user interface that facilitates use of the publicly released *AnalyticMesh* package.

## 5 EXTENSIONS TO OTHER ARCHITECTURES

We have so far focused our analysis on implicit functions constructed from MLPs with ReLU activations. In this section, we show that our proposed analytic meshing is applicable to implicit functions constructed from more advanced architectures, including those with shortcut connections [39] and max pooling. These architectures are used in some of the recent deep learning surface reconstruction methods [4, [5], [40], [41]. To facilitate the discussion, we will override some of the previously introduced math notations, which are self clear in the respective contexts.

### 5.1 Multi-Layer Perceptrons with Shortcut Connections

We consider two prototypical residual blocks with shortcut connections to present the extension. Fig. 4 gives the illustration. The first residual block aggregates two paths of forward signal propagation before a final ReLU activation, where one path is a shortcut connection and the other

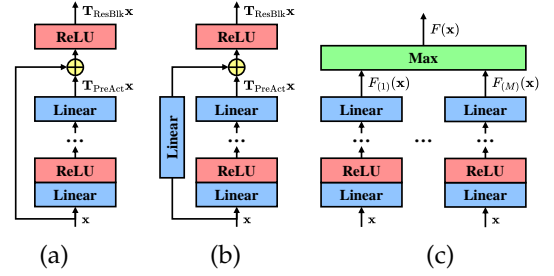


Fig. 4. Illustration of the three network architectures discussed in Sections 5.1 and 5.2. (a) A residual block with a shortcut connection. (b) A residual block with a shortcut connection of linear mapping. (c) A network with max pooling as the final aggregation of the outputs from multiple subnetworks.

path stacks  $L$  hidden layers respectively of  $n_l$  neurons,  $l \in \{1, \dots, L\}$ . Denote an input  $x \in \mathcal{X} \subset \mathbb{R}^{n_0}$ ; the residual block thus defines a mapping  $T_{\text{ResBlk}}x = g(x + T_{\text{PreAct}}x)$ , with the mapping  $T_{\text{PreAct}}x = W_L g(\dots g(W_1 x))$ , which also implies  $n_0 = n_L$ . Analysis in Section 2.2 shows that  $T_{\text{PreAct}}$  in fact partitions the space  $\mathcal{X}$  into a number of linear regions. To understand how the residual block  $T_{\text{ResBlk}}$  partitions the space, we first note from Definition 2 that the state functional of layer  $l$  in  $T_{\text{PreAct}}$ ,  $l \in \{1, \dots, L-1\}$ , is  $s_l(x) = [s_{l1}(x), \dots, s_{ln_l}(x)]^\top$ , where  $s_{lk}(x)$ ,  $k \in \{1, \dots, n_l\}$ , is defined by (5); we then extend Definition 2 and define the state functional for the output of the final layer of  $T_{\text{ResBlk}}$  as

$$s_L(x) = [s_{L1}(x), \dots, s_{Ln_L}(x)]^\top, \quad (19)$$

$$\text{s.t. } s_{Lk}(x) = \begin{cases} 1 & \text{if } \pi_k(x + T_{\text{PreAct}}x) > 0 \\ 0 & \text{if } \pi_k(x + T_{\text{PreAct}}x) \leq 0, \end{cases}$$

where  $k \in \{1, \dots, n_L\}$ . We thus have the state functional of  $T_{\text{ResBlk}}$  as  $s_{\text{ResBlk}}(x) = [s_1(x)^\top, \dots, s_L(x)^\top]^\top$ . Let  $\mathcal{R}_{\text{ResBlk}}$  denote the set of linear regions in  $\mathbb{R}^{n_0}$  that are partitioned by  $T_{\text{ResBlk}}$ . With definition (19), we can label any region  $r \in \mathcal{R}_{\text{ResBlk}}$  as  $s_{\text{ResBlk}}(r) \in \mathbb{J}^N$ , where  $N = \sum_{l=1}^L n_l$ . The region-wise linear mapping analogous to Lemma 4 can thus be defined as

$$T_{\text{ResBlk}}^r = \text{diag}(s_L(r)) \left( I + W_L \prod_{l=1}^{L-1} W_l^r \right) \quad (20)$$

$$\text{s.t. } W_l^r = \text{diag}(s_l(r)) W_l \quad l \in \{1, \dots, L-1\},$$

where  $I$  is an identity matrix of compatible size. Consequently, the neuron-wise linear mapping analogous to Corollary 5 is defined as

$$a_{lk}^r = \begin{cases} \pi_k \left( I + W_L \prod_{i=1}^{L-1} W_i^r \right) & \text{when } l = L \\ \pi_k \left( W_l \prod_{i=1}^{l-1} W_i^r \right) & \text{when } l \in \{2, \dots, L-1\} \\ \pi_k W_l & \text{when } l = 1, \end{cases} \quad (21)$$

where  $k \in \{1, \dots, n_l\}$  and  $W_i^r$  is defined the same as  $W_l^r$  in (20). One may use  $T_{\text{ResBlk}}$  as building blocks to construct an MLP  $T$  with shortcut connections, and consequently an implicit function  $F = f \circ T$ . Given the definitions (19), (20), and (21), the theoretical analysis in Section 3 and the analytic marching algorithm in Section 4 can be readily

applied to such an  $F$ . This extends our proposed method to architectures incorporating  $\mathbf{T}_{\text{ResBlk}}$  as building blocks.

The second residual block simply replaces the path of shortcut connection with a linear mapping  $\mathbf{V}\mathbf{x}$ , giving rise to a mapping of the residual block as  $\mathbf{T}_{\text{ResBlk}}\mathbf{x} = \mathbf{g}(\mathbf{V}\mathbf{x} + \mathbf{T}_{\text{PreAct}}\mathbf{x})$ , where  $\mathbf{V} \in \mathbb{R}^{n_L \times n_0}$  and  $\mathbf{T}_{\text{PreAct}}$  is the same as for the first residual block. This second residual block supports  $n_0 \neq n_L$ . Definitions similar to (19), (20), and (21) can be derived correspondingly; this extends our method to architectures incorporating the second type of residual blocks as building blocks.

## 5.2 Max pooling as a Final Aggregation of Deep Implicit Surface Networks

A few recent methods [40], [41] model a solid surface using constructive solid geometry [2]. They technically implement a union operation via a final max pooling over multiple deep implicit surface (sub-)networks. Fig. 4 gives an illustration. In this section, we show that our proposed method can also be extended to achieve analytic meshing from such a union of subnetworks.

Let  $F_{(i)} = f_{(i)} \circ \mathbf{T}_{(i)}$ ,  $i = 1, \dots, M$ , denote the individual implicit functions constructed from  $M$  subnetworks respectively of  $L_{(i)}$  hidden layers. Each  $F_{(i)}$  takes as input a same  $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^{n_0}$ ;  $n_0 = 3$  in the context of interest. Their union via max pooling computes

$$F(\mathbf{x}) = \max_{i \in \{1, \dots, M\}} F_{(i)}(\mathbf{x}). \quad (22)$$

Analysis in Sections 2 and 3 suggests that each  $\mathbf{T}_{(i)}$  partitions the input space  $\mathcal{X}$  into a set  $\mathcal{R}_{(i)}$  of linear regions/cells (convex polyhedrons), of which  $\tilde{\mathcal{R}}_{(i)}$  is the set of analytic cells relevant to the zero-level isosurface  $\{\mathbf{x} \in \mathbb{R}^3 | F_{(i)}(\mathbf{x}) = 0\}$ . For any  $\mathbf{x} \in \mathcal{X}$ , without loss of generality we assume that it falls in the cells  $r_{(i)} \in \mathcal{R}_{(i)}$ ,  $i = 1, \dots, M$ , respectively partitioned by the  $M$  subnetworks. Given the neuron-wise linear mappings defined in Corollary 5, we can spell out (22) for a local region around  $\mathbf{x}$  as

$$F(\mathbf{x}) = \max\{\mathbf{a}_{F_{(1)}}^{r_{(1)}}\mathbf{x}, \dots, \mathbf{a}_{F_{(M)}}^{r_{(M)}}\mathbf{x}\} \quad \forall \mathbf{x} \in r_{(1)} \cap \dots \cap r_{(M)}, \quad (23)$$

which is a point-wise maximum of linear functions. Let  $r := r_{(1)} \cap \dots \cap r_{(M)} \subset \mathbb{R}^3$ ; classical linear algebra suggests that  $r$  is a convex polyhedron as well.

To achieve analytic meshing from  $r$ , the key is to identify (possibly overlapped) subcells in  $r$  by specifying different  $j \in \{1, \dots, M\}$  as the indices, each of which satisfies  $F_{(j)} \geq F_{(i)} \quad \forall i \in \{1, \dots, M\}/j$ . More specifically, for any of such an index  $j$ , we assume that  $\mathbf{x}$  is in an analytic cell  $\tilde{r}_{(j)} \in \tilde{\mathcal{R}}_{(j)}$ , and override the notation  $\tilde{r} = r = r_{(1)} \cap \dots \cap \tilde{r}_{(j)} \cap \dots \cap r_{(M)}$ , which is the cell of interest relevant to extraction of zero-level isosurface — we note that some of  $r_{(i)}$ ,  $i \in \{1, \dots, M\}/j$ , may also be analytic cells achieved by their respective subnetworks, and the subsequent analysis holds without explicit specification of their analytic cell status. Define  $M - 1$  inequalities  $(\mathbf{a}_{F_{(i)}}^{r_{(i)}} - \mathbf{a}_{F_{(j)}}^{\tilde{r}_{(j)}})\mathbf{x} \leq 0$ ,  $i \in \{1, \dots, M\}/j$ ; the kernels  $\mathbf{a}_{F_{(i)}}^{r_{(i)}} - \mathbf{a}_{F_{(j)}}^{\tilde{r}_{(j)}}$  specify planes in  $\mathbb{R}^3$  that partition  $\tilde{r}$  as a convex polyhedral subcell, de-

noted as  $\tilde{r}_{\text{Sub}}$ . To specify the subcell, we compactly write the inequalities as

$$(\mathbf{A}^{/(j)} - \mathbf{1}_{M-1}\mathbf{a}_{F_{(j)}}^{\tilde{r}_{(j)}})\mathbf{x} \preceq 0, \quad (24)$$

where  $\mathbf{A}^{/(j)} = [\mathbf{a}_{F_{(1)}}^{r_{(1)}}; \dots; \mathbf{a}_{F_{(j-1)}}^{r_{(j-1)}}; \mathbf{a}_{F_{(j+1)}}^{r_{(j+1)}}; \dots; \mathbf{a}_{F_{(M)}}^{r_{(M)}}] \in \mathbb{R}^{(M-1) \times 3}$  and  $\mathbf{1}_{M-1}$  is a vector with all its  $M - 1$  entries as the value 1. Given that the cell  $\tilde{r}$  can be explicitly determined by the set of boundary planes  $\{\mathbf{H}_{l(i)k(i)}^{r(i)}\}$  with  $\mathbf{H}_{l(i)k(i)}^{r(i)} = \{\mathbf{x} \in \mathbb{R}^3 | \mathbf{a}_{l(i)k(i)}^{r(i)}\mathbf{x} = 0\}$ , where  $l(i) = 1, \dots, L_{(i)}$ ,  $k(i) = 1, \dots, n_{l(i)}$ , and  $i \in \{1, \dots, M\}/j$ , and the similarly defined set  $\{\mathbf{H}_{l(j)k(j)}^{\tilde{r}_{(j)}}\}$ , the subcell of interest for the specified index  $j$  can be written as

$$\mathbf{B}^{\tilde{r}_{\text{Sub}}}\mathbf{x} = \begin{bmatrix} \mathbf{A}^{/(j)} - \mathbf{1}_{M-1}\mathbf{a}_{F_{(j)}}^{\tilde{r}_{(j)}} \\ (\mathbf{I}_{(1)} - 2\text{diag}(\mathbf{s}(r_{(1)}))\mathbf{A}^{r_{(1)}} \\ \vdots \\ (\mathbf{I}_{(j)} - 2\text{diag}(\mathbf{s}(\tilde{r}_{(j)}))\mathbf{A}^{\tilde{r}_{(j)}} \\ \vdots \\ (\mathbf{I}_{(M)} - 2\text{diag}(\mathbf{s}(r_{(M)}))\mathbf{A}^{r_{(M)}} \end{bmatrix} \mathbf{x} \preceq 0, \quad (25)$$

where  $\mathbf{A}^{r(i)}$ ,  $i \in \{1, \dots, M\}/j$ , and  $\mathbf{A}^{\tilde{r}_{(j)}}$  are defined similarly as (13), and  $\mathbf{I}_{(i)}$  is an identity matrix of compatible size. Given the system (25), we have the analytic subcell associated with the union function  $F$  defined as

$$\mathcal{C}_F^{\tilde{r}_{\text{Sub}}} = \{\mathbf{x} \in \mathbb{R}^3 | \mathbf{B}^{\tilde{r}_{\text{Sub}}}\mathbf{x} \preceq 0\}. \quad (26)$$

Since  $\mathbf{a}_{F_{(j)}}^{\tilde{r}_{(j)}}\mathbf{x} \geq \mathbf{a}_{F_{(i)}}^{r_{(i)}}\mathbf{x} \quad \forall i \in \{1, \dots, M\}/j$  in the present subcell, we have the corresponding analytic face defined as

$$\mathcal{P}_F^{\tilde{r}_{\text{Sub}}} = \{\mathbf{x} \in \mathbb{R}^3 | \mathbf{a}_{F_{(j)}}^{\tilde{r}_{(j)}}\mathbf{x} = 0, \mathbf{B}^{\tilde{r}_{\text{Sub}}}\mathbf{x} \preceq 0\}. \quad (27)$$

In practice, to implement the analytic marching algorithm proposed in Section 4, for the present subcell  $\tilde{r}_{\text{Sub}}$  specified by the index  $j$ , we define an additional state functional for the final max pooling operation of  $F$  as

$$\mathbf{s}_{\text{MaxPool}}(\mathbf{x}) = \mathbf{e}_j, \quad (28)$$

which is an  $M$ -dimensional one-hot vector with the entry of 1 at the  $j^{\text{th}}$  index. Given the definitions (26), (27), and (28), our theoretical analysis in Section 3 and the analytic marching algorithm in Section 4 can be readily applied to such an architecture  $F$  with a final max pooling aggregation. The state functional (28) is used, together with the state functionals of the  $M$  subnetworks, to transit among analytic (sub-)cells during the analytic marching process.

## 6 LEARNING IMPLICIT SURFACE NETWORKS FOR GENERATIVE SHAPE MODELING

The analysis and algorithm presented in the previous sections assume that an implicit function  $F = f \circ \mathbf{T}$  has been given. In this section, we present different manners to construct and learn  $F$  for shape modeling and reconstruction. These manners have their respective advantages when  $F$  represents an implicit field of SDF or an occupancy field.



### 6.1 Learning as a Direct Shape Encoding

We first show the usefulness of analytic marching by directly fit individual 3D shapes to a model  $F = f \circ T$  constructed from a ReLU based MLP. A similar strategy is taken in [42] that demonstrates the compactness of neural network as a shape representation. Take a field of SDF as the example. Assume that a surface  $\mathcal{M}$  to be encoded is given. Following [17], we train the network with a regularized objective

$$\min_{F=f \circ T} \ell_{\mathcal{M}}(F) + \lambda_1^{\text{Direct}} \mathbb{E}_{\mathbf{x} \sim \mathbb{R}^3} \left| \|\nabla_{\mathbf{x}} F(\mathbf{x})\|_2 - 1 \right|, \quad (29)$$

with

$$\ell_{\mathcal{M}}(F) = \mathbb{E}_{\mathbf{z} \sim \mathcal{M}} [ |F(\mathbf{z})| + \lambda_2^{\text{Direct}} \|\nabla_{\mathbf{z}} F(\mathbf{z}) - \mathbf{n}_{\mathbf{z}}\|_2 ], \quad (30)$$

where  $\lambda_1^{\text{Direct}}$  and  $\lambda_2^{\text{Direct}}$  are penalty parameters, and  $\mathbf{n}_{\mathbf{z}}$  denotes the normal vector at a surface point  $\mathbf{z} \in \mathcal{M}$ ; the second term of (30) is optionally used when surface normals are available [17].

To improve the training efficiency, one may also replace the ReLU nonlinearity with a smooth version  $g_{\alpha}(x) = xe^{\alpha x} / (1 + e^{\alpha x})$  [43] during training, where  $\alpha > 0$  is a parameter controlling the degree of approximation, which is gradually increased until the training convergence. Note that after training, we still use the standard ReLU nonlinearity for analytic marching.

### 6.2 Global Decoding for Reconstruction of Novel Surface Shapes

Learning deep models to reconstruct novel shape instances gains popularity in recent research of deep learning surface reconstruction [4], [6]. Given training shapes, these methods usually train an encoder-decoder architecture for the purpose. Considering that the training shapes are point clouds sampled from ground-truth object surfaces, these methods train a point set encoder (e.g., a PointNet [44]) that outputs latent shape representation for a testing point cloud, which, together with sampled points in the 3D implicit space, are then fed into the decoder for inference of the implicit surface. Heavy MLP decoders are usually used in order for learning to generalize to novel shape instances. Given that the computation complexity of our analytic marching depends on the network capacities (cf. Section 4.1.1), we choose to use a hypernetwork [45] for shape decoding, instead of directly using an MLP decoder. More specifically, the hypernetwork can be chosen as a heavy MLP, which takes as input a latent shape representation from the encoder, and outputs weights of another light MLP, and the resulting light MLP is used as the implicit model  $F = f \circ T$  for surface inference via analytic marching. Such a hypernetwork based pipeline enjoys the benefit of precisely modeling and decoding novel shapes, while keeping an efficient process of shape inference.

Let  $E$  and  $H$  respectively denote the encoder and hypernetwork. Given a training set of ground-truth surfaces  $\{\mathcal{M} \in \mathfrak{M}\}$ , each of which is sampled from the distribution  $\mathfrak{M}$ , we use the following objective to train  $E$  and  $H$

$$\min_{E, H} \mathbb{E}_{\mathcal{M} \sim \mathfrak{M}} \ell_{\mathbb{R}^3}(E, H; \mathcal{M}) + \lambda^{\text{Global}} \|E(\mathcal{M})\|_2^2, \quad (31)$$

with

$$\ell_{\mathbb{R}^3}(E, H; \mathcal{M}) = \mathbb{E}_{\mathbf{x} \sim \mathbb{R}^3} |F(\mathbf{x}; H(E(\mathcal{M}))) - d(\mathbf{x}; \mathcal{M})|,$$

where  $H(E(\mathcal{M}))$  outputs network weights of an implicit  $F(\cdot; H(E(\cdot)))$ , and  $d(\mathbf{x}; \mathcal{M})$  is the ground-truth signed distance of  $\mathbf{x} \in \mathbb{R}^3$  to the surface  $\mathcal{M}$ ; the latent representation  $E(\mathcal{M})$  in (31) is regularized with its  $L_2$  norm, and  $\lambda^{\text{Global}}$  is a penalty parameter.

### 6.3 Improved Reconstruction via an Ensemble of Local Decoders

Global decoding either via a direct MLP or indirectly via a hypernetwork is limited in reconstructing surface shapes of complex topologies [40], [41]. To remedy, one strategy is to rely on local models and reconstruct a topologically complex surface as a union of local surface parts, i.e., a typical strategy in constructive solid geometry [2].

In this work, we technically implement this strategy using an ensemble of local decoders, as illustrated in Fig. 4-(c). Let  $E$  be the shape encoder, and  $F_{(i)}$ ,  $i = 1, \dots, M$ , denote the local decoders. Given the latent representation  $E(\mathcal{M})$  for a surface  $\mathcal{M}$ , we again use a hypernetwork  $H$  to estimate weights of  $\{F_{(i)}\}_{i=1}^M$ . As indicated by (22), the global implicit function  $F$  can be constructed as a maximum over outputs of the local decoders, i.e.,  $F(\mathbf{x}) = \max_{i \in \{1, \dots, M\}} F_{(i)}(\mathbf{x})$  for  $\mathbf{x} \in \mathbb{R}^3$ . Indeed, constructive solid geometry suggests that any boolean operation of occupancy fields can be realized by maximum function, and therefore composition of several components is the maximal value of individual occupancy values. The max function used in (22) can also be replaced as a soft version [43] to improve the training efficiency, i.e.,

$$F(\mathbf{x}) = \frac{\sum_{i=1}^M F_{(i)}(\mathbf{x}) e^{\beta F_{(i)}(\mathbf{x})}}{\sum_{i=1}^M e^{\beta F_{(i)}(\mathbf{x})}}, \quad (32)$$

where  $\beta > 0$  is a parameter controlling the degree of softness, which is gradually increased during training. Given a training set of ground-truth surfaces  $\{\mathcal{M} \in \mathfrak{M}\}$ , we use the following regularized objective to train such a model on occupancy fields

$$\min_{E, H} \mathbb{E}_{\mathcal{M} \sim \mathfrak{M}} \ell_{\mathbb{R}^3}(E, H; \mathcal{M}) + \lambda_1^{\text{Local}} \|E(\mathcal{M})\|_2^2 + \lambda_2^{\text{Local}} \|H(E(\mathcal{M}))\|_2^2, \quad (33)$$

with

$$\ell_{\mathbb{R}^3}(E, H; \mathcal{M}) = \mathbb{E}_{\mathbf{x} \sim \mathbb{R}^3} \text{CE}(F(\mathbf{x}; H(E(\mathcal{M}))), o(\mathbf{x}; \mathcal{M})),$$

where  $\text{CE}(\cdot, \cdot)$  denotes a binary cross-entropy loss,  $o(\mathbf{x}; \mathcal{M}) \in \{0, 1\}$  indicates the occupancy status of  $\mathbf{x} \in \mathbb{R}^3$ , and  $\lambda_1^{\text{Local}}$  and  $\lambda_2^{\text{Local}}$  are penalty parameters. In (33), we also use an  $L_2$ -norm regularization to constrain weights of the local decoders estimated from  $H$ , which is effective to regularize the learning.

## 7 EXPERIMENTS

**Datasets** We use two datasets of 3D solid objects for our experiments. The first dataset consists of object instances from five categories of the ShapeNet [46] (namely, “Rifle”, “Chair”, “Airplane”, “Sofa”, and “Table”). We construct our second, Richly Detailed (RD), dataset by collecting 5 geometrically and topologically complex shapes from Stanford 3D Scanning Repository [47], Artec3D [48], and Free3D [49]; Fig. 8 and Appendix C show these shapes. We normalize



object mesh models of the two datasets in the unit sphere of the 3D space. We use both the two datasets for our experiments of direct shape encoding (cf. Section 6.1). For experiments of learning to reconstruct novel object shapes (cf. Sections 6.2 and 6.3), we use the first dataset from ShapeNet where we split object instances of each category as training or test ones by a ratio of 4:1, and obtain input point clouds for the encoder  $E(\cdot)$  by sampling 2,048 points from each ground-truth mesh; when implementing the training objectives (31) and (33), ground-truth values of SDF or occupancy are calculated by linear interpolation from a dense grid obtained by [50], [51].

**Implementation Details** For direct shape encoding, we use an MLP of width 60 and depth 8. The network is trained for 1,500 epochs, with learning rates starting from  $1 \times 10^{-3}$  and dropping by a factor of 0.3 at epochs 1100, 1200, 1350 and 1450; we set  $\lambda_1^{\text{Direct}} = 0.2$ ,  $\lambda_2^{\text{Direct}} = 1.0$ , and the slope  $\alpha$  used in soft ReLU is initialized to 10 and gradually increased to 10,000. For experiments of learning to reconstruct novel shapes, we directly use PointNet [44] as the encoder  $E(\cdot)$ . We use a hypernetwork of MLP with depth 2 and width 1,024 for global decoding, which gives a decoding MLP of width 60 and depth 6; we set  $\lambda^{\text{Global}} = 0.01$ ; learning rates start from  $3 \times 10^{-4}$  and drop at the epoch 2,400, 4,200, and 5,400 respectively by a factor of 0.2, until a total of 6,000 epochs. For local decoding, we again use a hypernetwork of MLP with depth 2 and width 1,024, which results in an ensemble of  $M = 4$  local decoders; each local decoder is an MLP of width 32 and depth 4; we set  $\lambda_1^{\text{Local}} = 0.01$  and  $\lambda_2^{\text{Local}} = 5 \times 10^{-5}$ ; the hypernetwork is trained for 3,000 epochs, and learning rates start from  $3 \times 10^{-4}$  and are reduced by a factor of 0.2 at the epoch 1,200, 2,100, and 2,700; during inference, individual shape components are independently extracted and then integrated using [52]. All experiments are conducted on a single Nvidia Tesla K80.

**Comparative Methods and Evaluation Metrics** We compare our proposed meshing algorithm of Analytic Marching (AM) with existing ones, which have already been the standard meshing choices given that implicit functions are provided; the comparative methods include Greedy Meshing (GM), Marching Cubes (MC) [13], Marching Tetrahedra (MT) [21], and Dual Contouring (DC) [14]. All these methods are based on discrete sampling of the 3D space for evaluation of signed distances or occupancies for the sampled points, which are then used for extraction of the zero-level isosurfaces. They are thus by nature different from our AM. *We emphasize that the comparisons are made on meshing algorithms themselves, independent of how the implicit functions have been constructed or learned; our presented methods of learning implicit surface networks in Section 6 are mainly to set contexts for such a meshing comparison.* As such, we use evaluation metrics of (approximate) distances between each ground-truth mesh and those extracted by different meshing algorithms, including Chamfer Distance (CD) and Earth Mover Distance (EMD), each of which computes symmetric, pairwise Euclidean distance between sampled point sets, Intersection over Union (IoU) that measures how the two meshes overlap and whose values range in  $[0, 1]$ , and F-score (F) that measures the symmetrical percentage of reachable surface areas within a given distance  $\tau$ ; we use

a default  $\tau = 5 \times 10^{-3}$ . In addition, we report relevant attributes of meshing algorithms and results, such as the number of triangular faces per mesh (#TriFace) and running time, where triangular faces of the results from our AM are converted from polygonal ones (cf. Section 4.4 for how the conversion can be conducted).

## 7.1 Analysis of Analytic Marching

In this section, we analyze various properties of analytic marching. Experiments are conducted on instances of ShapeNet by directly fitting implicit MLP functions.

**Effects of Network Capacities** Section 4.1 suggests that the meshing precision of our AM depends on the (maximal) number of linear regions partitioned by an MLP, whose order is exponential to network depth and polynomial to network width. To verify empirically, we design experiments by using two groups of MLPs that respectively have the same numbers of 360 and 900 neurons. The first group distributes their neurons as D4-W90, D6-W60, and D8-W45, where “D” is for depth and “W” is for width, and the second group distributes their neurons as D10-W90, D15-W60, and D20-W45. Results in Table 1 confirm that mesh accuracies increase consistently with the increased network capacities, but at a cost of much increased face numbers per mesh. Given the same number of neurons, it seems that a balanced depth-width neuron distribution is more advantageous at the studied regime of relatively lower network capacities.

TABLE 1

Meshing accuracies of analytic marching by using MLPs of different capacities. “D” stands for network depth and “W” stands for network width. Results are obtained by averaging over 200 instances of the “Rifle” category from ShapeNet.

Architecture	CD ↓	EMD ↓	IoU ↑	F@ $\tau$ ↑	#TriFace
D4-W90	0.616	0.00860	0.865	0.848	195,658
D6-W60	0.529	0.00800	0.869	0.855	200,263
D8-W45	0.567	0.00812	0.857	0.842	186,903
D10-W90	0.364	0.00585	0.902	0.881	844,114
D15-W60	0.385	0.00623	0.889	0.874	671,752
D20-W45	0.421	0.00689	0.861	0.866	507,742

**Results of Different Categories** To investigate how AM performs on instances of different categories whose surface complexities may vary, we conduct experiments on five categories of ShapeNet, using the MLP of D6-W60 as mentioned above. Table 2 shows that the category of “Airplane” has the best meshing accuracies in terms of 3 of the total 4 metrics, which is in accordance with the common intuition on the simplicity of its shapes.

TABLE 2

Meshing accuracies of analytic marching on five categories of ShapeNet, using an MLP of depth 6 and width 60 (the same D6-W60 network as in Table 1). Results are obtained by averaging over 200 instances of each category.

Category	CD ↓	EMD ↓	IoU ↑	F@ $\tau$ ↑	#TriFace
Rifle	0.529	0.00800	0.869	0.855	200,263
Chair	0.727	0.00801	0.896	0.529	365,231
Airplane	0.325	0.00532	0.894	0.898	262,116
Sofa	0.678	0.00665	0.966	0.532	313,718
Table	0.698	0.00716	0.907	0.516	326,808

**Efficiencies of Customized Triggering Schemes** We present different triggering schemes of AM in Section 4.3 as improvements over the simple one of solving (17) via SGD, including Sphere Tracing (ST) and Dichotomy (DICH). To verify their efficiencies, we conduct experiments of finding 1,024 points on the surface for each instance of all the five categories of ShapeNet, using the aforementioned MLP of D6-W60. We compare the three schemes in terms of their required numbers of iterations (#Iter) to converge, the corresponding running time (second), and the rate of success (SR). Table 3 tells that for signed distance field (SDF), all the schemes can successfully converge to find the points on the surface and thus trigger the AM algorithm, and ST and DICH are much faster than SGD; for occupancy field (OF), DICH is the only scheme that can trigger AM successfully and efficiently. We thus recommend DICH as the default triggering scheme of AM.

**Parallel Marching with CUDA Implementation** Our AM algorithm supports parallel marching, which simultaneously marches the analytic cells to recover the mesh. In this work, we implement parallel marching with CUDA implementation. Table 4 shows that on Nvidia GPUs (64-bit floating point), parallel marching with the ST triggering of 1,024 points on the surface is nearly an order-of-magnitude faster than solving AM on CPUs using SGD triggering of a same number of points. These experiments are conducted on all the five categories of ShapeNet, using the MLP of D6-W60 as mentioned above.

**Mesh Simplification in AnalyticMesh** As indicated by the results in Tables 1 and 2, in spite of the exactness, AM tends to produce polygon meshes with huge numbers of polygon faces. This brings inconvenience to downstream processing on meshes. It is desirable to simplify the meshes at no or less sacrifice of mesh precisions. As stated in Section 4.4, we have incorporated such postprocessing operations into our publicly released package of AnalyticMesh. Quantitative results in Table 5 show that by mesh simplification, the loss of precision is negligible even when preserving only 1% of the original numbers of mesh faces. A corresponding visualization is shown in Fig. 5. *Except mentioning otherwise, we present all results of AM in the subsequent sections after mesh simplification at a ratio of 10%.*

## 7.2 Direct Shape Encoding

In this section, we compare our proposed AM with existing methods in the context of direct shape encoding, where an implicit network of SDF is trained to fit each shape instance (cf. Section 6.1 for the details). Experiments are conducted on both the five categories of ShapeNet and the RD dataset. For ShapeNet, we use an MLP of depth 6 and width 60 (the same D6-W60 model as in the preceding section); for RD, we use an MLP of depth 8 and width 60.

The plotting of numerical results on ShapeNet instances is shown in Fig. 6. Under different evaluation metrics, mesh accuracies of existing methods are upper bounded by our proposed AM. These methods of marching cubes family recover a mesh by firstly sampling a grid of 3D points at a specified resolution, followed by identifying the intersections with the zero-level isosurface; as such, their

accuracies depend on the sampling resolutions. Instead, our AM recovers the mesh exactly captured by the fitted MLP.

Fig. 7 shows the plotting of numerical results on the five shape instances of RD dataset. Example results for one of the shapes are given in Fig. 8. Mesh accuracies of existing methods are still upper bounded by our proposed AM. We also notice from Fig. 8 that existing methods may give geometrically and/or topologically wrong recoveries, even when their sampling resolutions are increased to  $512^3$ . In contrast, AM gives much better and visually pleasant results. Results of other shape instances from the RD dataset are given in Appendix C.

## 7.3 Global Learning for Novel Shape Reconstruction

In this section, we compare our proposed AM with existing meshing methods in the context of learning a global model for decoding of novel shape instances. We train a hypernetwork of MLP with depth 2 and width 1,024, which learns weights of a light MLP with depth 6 and width 60 for SDF modeling. Experiments are conducted on the five ShapeNet categories by training the hypernetwork on the training shapes and testing on novel ones. Other learning setups are given in the beginning of Section 7. Quantitative results in Table 6 show that AM, *after mesh simplification at a ratio of 10%*, achieves mesh reconstruction accuracies nearly identical to those of MC512 (i.e., marching cubes with sampling of  $512^3$  3D points), while running much faster with much fewer numbers of mesh faces; this is due to the better accuracies of the original meshes exactly recovered by AM. The qualitative comparisons given in Appendix D are consistent to those quantitative ones.

## 7.4 Novel Shape Reconstruction by Learning an Ensemble of Local Decoders

In this section, we investigate whether the results of AM can be improved by learning an ensemble of local decoders for novel shape reconstruction. We train a hypernetwork of MLP with depth 2 and width 1,024, which learns weights for an ensemble of 4 subnetworks for occupancy modeling; each subnetwork is of depth 4 and width 32; thus in total the ensemble has 512 neurons, slightly more than the D6-W60 model used in global decoding. We again conduct experiments on the five ShapeNet categories by training the hypernetwork on the training shapes and testing on novel ones. Other learning setups are given in the beginning of Section 7. Quantitative results in Table 7 show that the ensemble model indeed improves the accuracies of mesh reconstructions over those obtained by a single, global model, at the cost of a slightly slower inference. On all the five categories, AM, *after mesh simplification at a ratio of 10%*, gives more accurate reconstructions at a much faster inference and much fewer numbers of mesh faces than MC512 does. The qualitative comparisons given in Appendix D are consistent with the quantitative ones, where the recovered components by individual decoders of the ensemble suggest that the ensemble learns the shapes naturally in a constructive manner.

## 8 CONCLUSION AND FUTURE RESEARCH

We have presented in this paper an exact meshing solution from deep implicit surface networks. Our analytic marching

TABLE 3

Comparisons among different triggering scheme of analytic marching (cf. Section 4.3). Results are obtained by finding 1,024 points on the surface for each instance of all the five categories of ShapeNet, using an MLP of depth 6 and width 60. We use measures of the required number of iterations (#Iter) to converge, the corresponding running time (second), and the success rate (SR). The mark “-” indicates the failure of triggering.

Method	SGD			ST			DICH		
Metric	#Iter	Time(sec.)	SR	#Iter	Time(sec.)	SR	#Iter	Time(sec.)	SR
SDF	531	3.47	1.0	3	0.0267	1.0	11	0.0657	1.0
OF	—	—	0.0	—	—	0.0	22	0.0933	1.0

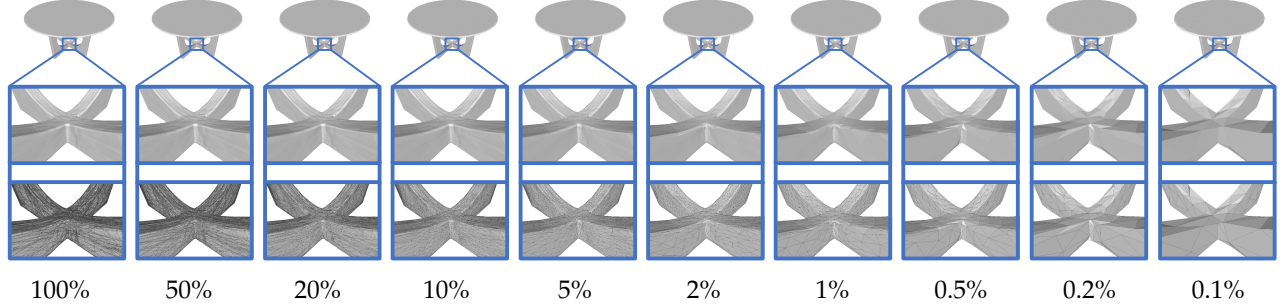


Fig. 5. Example results of analytic marching after mesh simplification at different ratios. Better viewing by zooming in the electronic version.

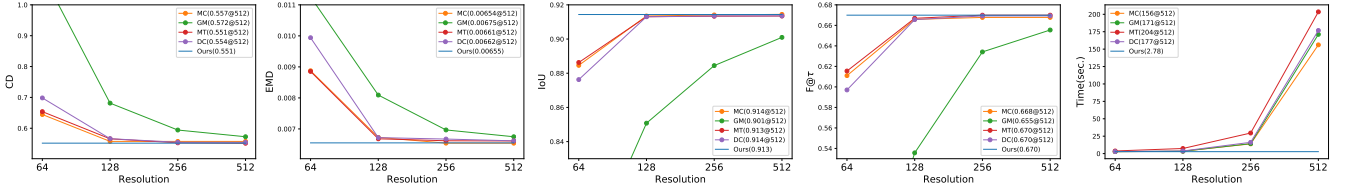


Fig. 6. Quantitative comparisons of direct shape encoding under metrics of recovery precision and inference time. For greedy meshing (GM), marching cubes (MC), marching tetrahedra (MT), and dual contouring (DC), results under a resolution range of discrete point sampling from  $64^3$  to a GPU memory limit of  $512^3$  are presented. Experiments are conducted on shape instances of five categories from ShapeNet, using an MLP of depth 6 and width 60 for SDF modeling.

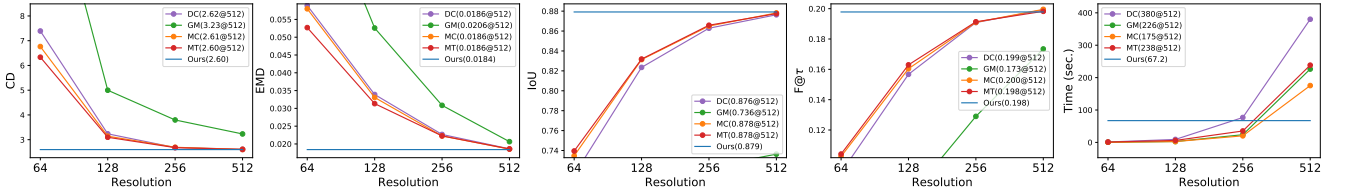


Fig. 7. Quantitative comparisons of direct shape encoding under metrics of recovery precision and inference time. For greedy meshing (GM), marching cubes (MC), marching tetrahedra (MT), and dual contouring (DC), results under a resolution range of discrete point sampling from  $64^3$  to a GPU memory limit of  $512^3$  are presented. Experiments are conducted on five geometrically and topologically complex shape instances, using an MLP of depth 8 and width 60 for SDF modeling.

TABLE 4

Efficiency of parallel marching with CUDA implementation. Experiments are conducted on a Nvidia Tesla K80 (64-bit floating point) using an MLP of depth 6 and width 60 for SDF modeling. Results are averaged over instances of all the five categories of ShapeNet. SGD, ST, and DICH are the three triggering schemes discussed in Section 4.3.

Implementation of Parallel Marching	Time (second)
CPU + SGD	20.8
CUDA + SGD	6.22
CUDA + DICH	2.82
CUDA + ST	2.78

algorithm works by marching along the analytic cells in the input 3D space and exactly recovering the analytic faces. We have proved that under mild, numerical conditions, the recovered analytic faces are guaranteed to connect and form a

TABLE 5

Quantitative results of mesh simplification in *AnalyticMesh*. Mesh results are obtained by first fitting an MLP of depth 6 and width 60 to each instance of the five categories of ShapeNet, and then recovering the mesh via AM and simplifying the mesh using the corresponding operations in *AnalyticMesh*. Note that the simplification is controlled by quadric error metrics [36], and a specified ratio does not translate exactly as a correspondingly reduced number of mesh faces.

Ratio	CD ↓	EMD ↓	IoU ↑	F@τ ↑	#TriFace	Mesh Size
100%	0.591	0.00703	0.906	0.666	293,627	4,161 KB
50%	0.591	0.00704	0.906	0.666	148,686	2,093 KB
20%	0.591	0.00704	0.906	0.666	59,218	835 KB
10%	0.591	0.00705	0.905	0.666	29,342	417 KB
5%	0.592	0.00713	0.905	0.665	14,762	209 KB
2%	0.594	0.00722	0.903	0.665	5,909	83.7 KB
1%	0.599	0.00730	0.901	0.664	2,964	42.0 KB
0.5%	0.608	0.00768	0.899	0.663	1,480	21.1 KB
0.2%	0.797	0.00996	0.882	0.653	586	8.54 KB
0.1%	1.59	0.0134	0.853	0.634	294	4.37 KB

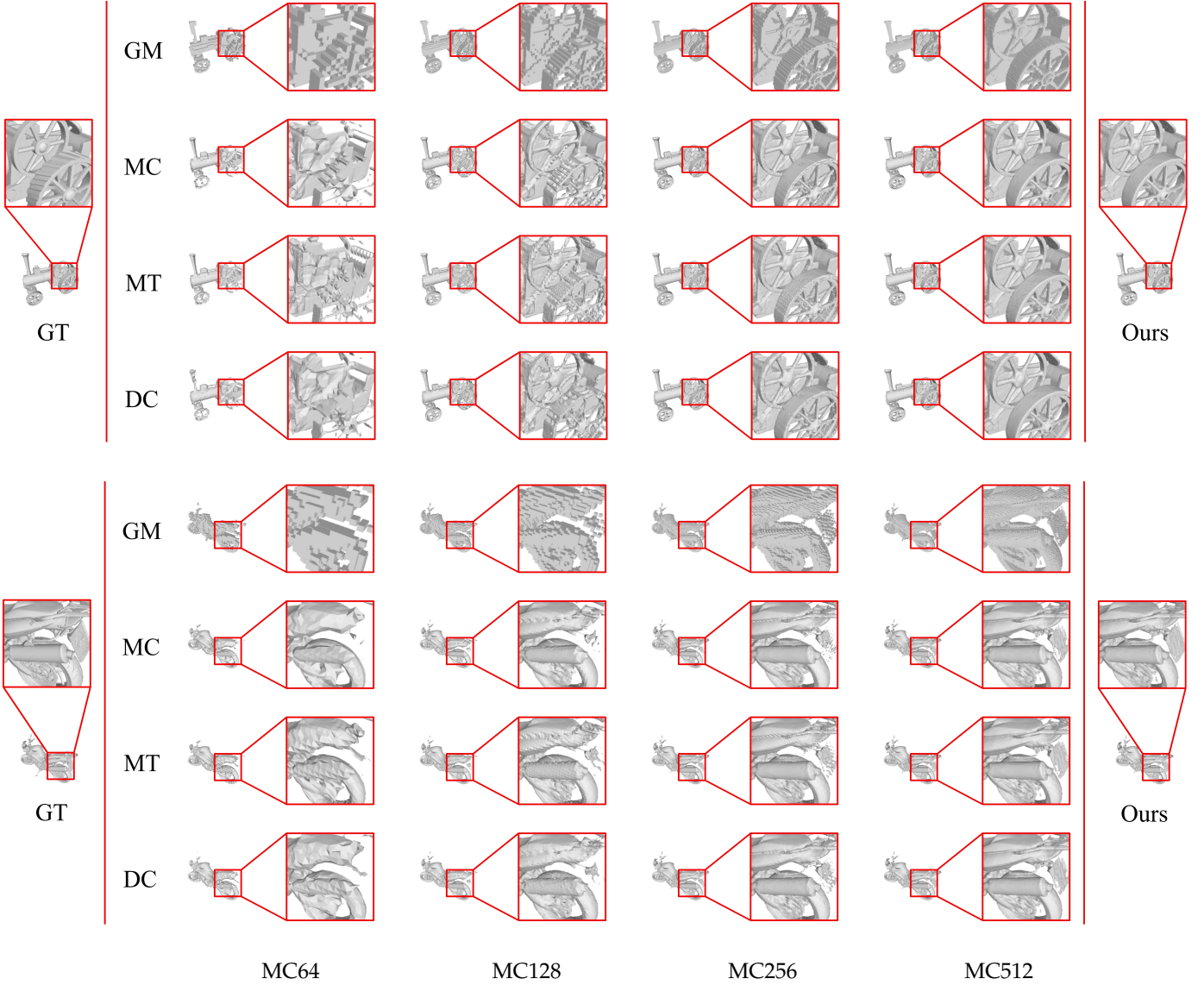


Fig. 8. Qualitative comparisons of direct shape encoding. The shown shape instance is fitted to an MLP of depth 8 and width 60 for SDF modeling. For greedy meshing (GM), marching cubes (MC), marching tetrahedra (MT), and dual contouring (DC), results under a resolution range of discrete point sampling from  $64^3$  to a GPU memory limit of  $512^3$  are presented. Better viewing by zooming in the electronic version.

TABLE 6

Quantitative comparisons between analytic marching (AM) and marching cubes (MC) in the context of learning a global model for decoding of novel shape instances. Experiments are conducted on shape instances of five categories from ShapeNet, by training a hypernetwork that gives weights of a light MLP with depth 6 and width 60 for SDF modeling. *Results of AM are after simplification of the originally recovered meshes at a ratio of 10%.* For MC, the sampling resolutions of 3D points range from  $64^3$  to  $512^3$ .

Metric Method	CD					#TriFace					Time (sec.)				
	MC64	MC128	MC256	MC512	AM	MC64	MC128	MC256	MC512	AM	MC64	MC128	MC256	MC512	AM
Airplane	18.0	5.00	2.91	<b>2.86</b>	<b>2.86</b>	<b>3601</b>	16511	70051	284293	5948	2.54	3.79	14.3	159	<b>2.04</b>
Chair	28.4	18.7	<b>15.4</b>	<b>15.4</b>	<b>15.4</b>	9850	41265	168071	677782	<b>9465</b>	2.58	3.86	13.1	139	<b>2.57</b>
Rifle	12.3	7.82	6.61	<b>5.34</b>	5.35	<b>2109</b>	8885	36183	146300	4230	2.51	3.65	13.7	157	<b>1.72</b>
Sofa	3.10	3.06	3.05	3.05	<b>3.04</b>	11860	48433	196007	788885	<b>5214</b>	2.59	3.99	14.9	169	<b>1.85</b>
Table	27.1	11.4	9.35	<b>9.34</b>	9.35	<b>9692</b>	42270	172320	695810	10157	2.69	4.28	14.2	147	<b>2.46</b>

closed, piecewise planar surface. Our theory and algorithm also support advanced MLPs with shortcut connections and max pooling. We have empirically demonstrated the advantages of our method over existing meshing algorithms in the contexts of either direct shape encoding or learning to decode novel shape instances.

Our current theory and algorithm assume the MLPs

with ReLU activations. For other variants of the ReLU family, such as leaky ReLU [53] or parametric ReLU [54], we note that it is possible to develop the corresponding theory based on the recent result in [55], which extends [19] and bounds the numbers of linear regions partitioned by general, piece-wise linear networks. For softer activations such as Softplus [43], ELU [56], or Swish [57], one may em-

TABLE 7

Quantitative comparisons between analytic marching (AM) and marching cubes (MC) in the context of learning an ensemble of local decoders for reconstructions of novel shape instances. Experiments are conducted on shape instances of five categories from ShapeNet, by training a hypernetwork that gives weights of the ensemble for occupancy modeling; the ensemble is formed by aggregating, via max pooling, outputs of 4 subnetworks, each of which is of depth 4 and width 32. *Results of AM are after simplification of the originally recovered meshes at a ratio of 10%.* For MC, the sampling resolutions of 3D points range from  $64^3$  to  $512^3$ .

Metric Method	CD					#TriFace					Time (sec.)				
	MC64	MC128	MC256	MC512	AM	MC64	MC128	MC256	MC512	AM	MC64	MC128	MC256	MC512	AM
Airplane	2.17	0.803	0.608	<b>0.607</b>	<b>0.607</b>	<b>4881</b>	19626	78375	311699	6775	<b>3.97</b>	6.75	28.6	198	4.62
Chair	3.95	3.82	3.55	<b>3.51</b>	<b>3.51</b>	12538	47807	184007	725064	<b>5625</b>	3.97	6.72	27.2	186	<b>3.61</b>
Rifle	2.48	1.55	1.48	<b>1.42</b>	<b>1.42</b>	<b>2683</b>	10132	38553	150220	3609	3.91	6.57	26.8	186	<b>3.16</b>
Sofa	2.03	1.97	1.91	1.87	<b>1.86</b>	14237	52950	203497	800701	<b>4100</b>	3.99	6.82	28.5	195	<b>3.49</b>
Table	6.36	3.89	3.66	3.63	<b>3.62</b>	10389	43442	174667	700780	<b>4951</b>	4.10	7.62	34.5	247	3.71

ploy them in an asymptotic manner during training of the implicit networks, similar to the efficient training scheme used in Section 6.1; after training, the networks approach ReLU based ones, whose zero-level isosurfaces can then be recovered by analytic marching. For more general smooth activations, including the very recent sinusoid one [18], it remains unclear to develop their exact meshing theories. One may consider using piecewise linear functions to approximate the smooth activations, or employing a strategy of distillation [58] to transfer the knowledge learned in the teacher implicit networks of general smooth activations into the student, ReLU based ones. We leave these extended studies as future research.

Direct shape encoding presented in Section 6.1 suggests a possibly more compact way of coding, storing, and transmitting 3D shapes as implicit surface networks, as pointed out firstly in [42]. However, the neurally encoded shapes in [42] are still to be decoded via meshing algorithms such as marching cubes, which, as we have argued, would cause loss of the precisions already encoded in the implicit networks. As an algorithm of exact meshing, our proposed analytic marching makes a solid step towards making implicit surface networks truly as a new promise of compact shape representations.

## REFERENCES

- [1] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *ISMAR*, 2011, pp. 127–136.
- [2] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Levy, *Polygon Mesh Processing*. CRC Press, 2010.
- [3] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *SIGGRAPH*, 1996.
- [4] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *CVPR*, Jun 2019.
- [5] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *CVPR*, 2019.
- [6] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *CVPR*, 2019.
- [7] J. F. Blinn, "A generalization of algebraic surface drawing," in *TOG*, vol. 1, no. 3, Jul. 1982, p. 235–256.
- [8] H. Nishimura, M. Hirai, T. Kawai, T. Kawata, I. Shirkawa, and K. Omura, "Object modeling by distribution function and a method of image generation," in *Trans. Inst. Electron Commun. Eng. Japan*, vol. 68, 1985, p. 718.
- [9] G. Wyvill, C. McPheeters, and B. Wyvill, "Data structure for soft objects," in *The Visual Computer*, vol. 2, Aug 1986, pp. 227–234.
- [10] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3d objects with radial basis functions," in *SIGGRAPH*. Association for Computing Machinery, 2001, p. 67–76.
- [11] J. C. Carr, W. R. Fright, and R. K. Beatson, "Surface interpolation with radial basis functions for medical imaging," in *IEEE Transactions on Medical Imaging*, vol. 16, no. 1, 1997, pp. 96–107.
- [12] G. Turk and J. F. O'Brien, "Shape transformation using variational implicit functions," in *SIGGRAPH*, 1999, p. 335–342.
- [13] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *SIGGRAPH*, vol. 21, no. 4, Aug. 1987, p. 163–169.
- [14] T. Ju, F. Losasso, S. Schaefer, and J. Warren, "Dual contouring of hermite data," in *TOG*, vol. 21, no. 3, Jul. 2002, p. 339–346.
- [15] R. Kolluri, "Provably good moving least squares," in *ACM Transactions on Algorithms*, vol. 4, 2008.
- [16] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *AISTATS*, vol. 15, 01 2011, pp. 315–323.
- [17] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit geometric regularization for learning shapes," in *Proceedings of Machine Learning and Systems*, 2020, pp. 3569–3579.
- [18] V. Sitzmann, J. N. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," in *NIPS*, 2020.
- [19] G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *ICANIPS*, 2014.
- [20] Q. Xu, W. Wang, D. Ceylan, R. Mech, and U. Neumann, "Disn: Deep implicit surface network for high-quality single-view 3d reconstruction," in *ICANIPS*, vol. 32, 2019.
- [21] A. Doi and A. Koide, "An efficient method of triangulating equivalent surfaces by using tetrahedral cells," in *IEICE Transactions on Information and Systems*, vol. 74, Jan 1991.
- [22] R. Pascanu, G. Montúfar, and Y. Bengio, "On the number of response regions of deep feed forward networks with piece-wise linear activations," in *ICLR*, 2014.
- [23] K. Jia, S. Li, Y. Wen, T. Liu, and D. Tao, "Orthogonal deep neural networks," in *TPAMI*, vol. 43, no. 4, 2021, pp. 1352–1368.
- [24] L. Chu, X. Hu, J. Hu, L. Wang, and J. Pei, "Exact and consistent interpretation for piecewise linear neural networks: A closed form solution," in *ACM SIGKDD*, 2018, pp. 1244–1253.
- [25] J. Lei and K. Jia, "Analytic marching: An analytic meshing solution from deep implicit surface networks," in *ICML*, 2020.
- [26] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "A papier-mâché approach to learning 3d surface generation," in *CVPR*, Jun 2018.
- [27] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2mesh: Generating 3d mesh models from single rgb images," in *ECCV*, 2018.
- [28] J. Tang, X. Han, J. Pan, K. Jia, and X. Tong, "A skeleton-bridged deep learning approach for generating meshes of complex topologies from single rgb images," in *CVPR*, 2019, pp. 4536–4545.
- [29] J. Pan, X. Han, W. Chen, J. Tang, and K. Jia, "Deep mesh reconstruction from single rgb images via topology modification networks," in *ICCV*, 2019, pp. 9964–9973.
- [30] P. Orlik, H. Teraso, M. Berger, B. Eckmann, and S. Varadhan, *Arrangements of Hyperplanes*, ser. Grundlehren der mathematischen Wissenschaften: a series of comprehensive studies in mathematics. U.S. Government Printing Office, 1992.
- [31] T. Zaslavsky, "Facing up to arrangements: Face-count formulas for partitions of space by hyperplanes," in *Memoirs of the American Mathematical Society*. American Mathematical Society, 1975.
- [32] A. Gorban and I. Tyukin, "Blessing of dimensionality: Mathematical foundations of the statistical physics of data," in *Philos. Trans. R. Soc. A*, vol. 376, Mar 2018.



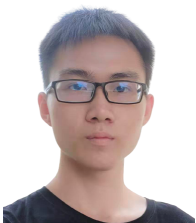
- [33] D. Avis and K. Fukuda, "A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra," in *Annual Symposium on Computational Geometry*, 1991, p. 98–104.
- [34] J. Hart, "Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces," in *The Visual Computer*, 1996.
- [35] S. Russ, "A translation of bolzano's paper on the intermediate value theorem," *Historia Mathematica*, vol. 7, pp. 156–185, 1980.
- [36] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *SIGGRAPH*, 1997, p. 209–216.
- [37] G. Guennebaud and M. Gross, "Algebraic point set surfaces," in *SIGGRAPH*, 2007.
- [38] P. Liepa, "Filling holes in meshes," in *Eurographics Symposium on Geometry Processing*. The Eurographics Association, 2003.
- [39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.
- [40] Z. Chen, K. Yin, M. Fisher, S. Chaudhuri, and H. Zhang, "Bae-net: Branched autoencoder for shape co-segmentation," in *ICCV*, 2019.
- [41] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Stoll, and C. Theobalt, "Patchnets: Patch-based generalizable deep implicit 3d shape representations," in *ECCV*, 2020.
- [42] T. Davies, D. Nowrouzezahrai, and A. Jacobson, "On the effectiveness of weight-encoded neural implicit 3d shapes," in *arXiv:2009.09808*, 2021.
- [43] M. Lange, D. Zühlke, O. Holz, and T. Villmann, "Applications of lp-norms and their smooth approximations for gradient based learning vector quantization," in *ESANN*, 2014.
- [44] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017, pp. 77–85.
- [45] G. Littwin and L. Wolf, "Deep meta functionals for shape representation," in *ICCV*, Oct 2019.
- [46] A. X. Chang, T. Funkhouser, L. Guibas, 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:1512.03012*, 2015.
- [47] The stanford 3d scanning repository. [Online]. Available: <http://graphics.stanford.edu/data/3Dscanrep/>
- [48] Artec3d. [Online]. Available: <https://www.artec3d.com/>
- [49] Free3d. [Online]. Available: <https://free3d.com/>
- [50] H. Xu and J. Barbič, "Signed distance fields for polygon soup meshes," in *Proceedings of Graphics Interface*, 2014, p. 35–41.
- [51] F. S. Sin, D. Schroeder, and J. Barbic, "Vega: Non-linear fem deformable object simulator," in *Computer Graphics Forum*, 2013.
- [52] M. Douze, J.-S. Franco, and B. Raffin, "Quickcsg: Fast arbitrary boolean combinations of n solids," in *arXiv:1706.01558*, 2017.
- [53] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML*, 2013.
- [54] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *ICCV*, 2015, pp. 1026–1034.
- [55] Q. Hu, H. Zhang, F. Gao, C. Xing, and J. An, "Analysis on the number of linear regions of piecewise linear neural networks," in *IEEE Trans. Neural Netw. Learn. Syst.*, 2020, pp. 1–10.
- [56] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," in *ICLR*, 2016.
- [57] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," in *ICLR*, 2018.
- [58] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Workshop*, 2015.



**Kui Jia** received the Ph.D. degree in computer science from the Queen Mary University of London, London, U.K., in 2007. He was with the Shenzhen Institute of Advanced Technology of the Chinese Academy of Sciences, Shenzhen, China, Chinese University of Hong Kong, Hong Kong, the Institute of Advanced Studies, University of Illinois at Urbana-Champaign, Champaign, IL, USA, and the University of Macau, Macau, China. He is currently a Professor with the School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China, and is the Director of Geometric Perception and Intelligence Research Lab. His recent research focuses on theoretical deep learning and its applications in vision and robotic problems, including deep learning of 3D data and deep transfer learning. He has been serving as Associate Editors for TIP and TSMC.



**Yi Ma** received the bachelor's degree in automation and applied mathematics from Tsinghua University, Beijing, China, in 1995, and the master's degree in EECS and mathematics and the Ph.D. degree in EECS from the University of California at Berkeley, Berkeley, in 2000. From 2000 to 2011, he has served on the Faculty of the ECE Department, University of Illinois at Urbana-Champaign. From 2009 to 2014, he has served as a Principal Researcher and a Research Manager for the Visual Computing Group, Microsoft Research Asia, Beijing, China. From 2014 to 2017, he was a Professor and the Executive Dean of the School of Information Science and Technology, ShanghaiTech University, China. Since January 2018, he has been with the Faculty of the EECS Department, University of California at Berkeley. He has written two textbooks *An Invitation to 3D Vision* (Springer) and *Generalized Principal Component Analysis* (Springer). He is a fellow of ACM. He has received best paper awards from ICCV, ECCV, and ACCV. He received the CAREER Award from the NSF and the YIP Award from the ONR. He has served as an Associate Editor for IJCV, SIIMS, SIMODS, the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (PAMI), and the IEEE TRANSACTIONS ON INFORMATION THEORY.



**Jiabao Lei** is currently working toward the master degree in the School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China. Recently his research interests mainly include 3D geometric representation and surface reconstruction in the field of deep learning.

## APPENDIX A

### PROOF OF THEOREM 6

*Proof.* Since  $F = f \circ T$  is constructed from a ReLU based MLP  $T$ , its zero-level isosurface  $\mathcal{Z}$  is piecewise planar. The proof proceeds by first showing that each planar face on  $\mathcal{Z}$  uniquely corresponds to an analytic face in an analytic cell, and then showing that for any pair of planar faces connected on  $\mathcal{Z}$  (since  $\mathcal{Z}$  is closed by assumption), their corresponding analytic faces are connected via boundaries of their respective analytic cells.

Let  $\mathcal{P}_1$  denote a planar face on the surface  $\mathcal{Z}$ , and  $\mathbf{n}_1 \in \mathbb{R}^3$  be its normal. We have  $\mathbf{n}_1^\top \mathbf{x} = 0 \forall \mathbf{x} \in \mathcal{P}_1$ . Equation (11) tells that  $\mathbf{n}_1$  must be proportional at least to one of  $\{\mathbf{a}_F^r | r \in \mathcal{R}\}$ . By the unique plane condition, i.e., each of  $\{\mathbf{a}_F^r | r \in \mathcal{R}\}$  is uniquely defined, we have  $\mathbf{n}_1 \propto \mathbf{a}_F^{r_1}$  of a certain region  $r_1$ . Assume  $r_1$  is not an analytic cell, which suggests that there exists no intersection between  $\mathbf{a}_F^{r_1}$  and  $r_1$  and we have  $\mathbf{a}_F^{r_1} \mathbf{x} = \mathbf{n}_1^\top \mathbf{x} \neq 0$  for all  $\mathbf{x} \in r_1$ , and thus  $\mathcal{P}_1 \cap r_1 = \emptyset$ ; it suggests that the normal  $\mathbf{n}_1$  of  $\mathcal{P}_1$  is induced in a different region  $r'_1$  by  $\mathbf{n}_1 \propto \mathbf{a}_F^{r'_1} = \mathbf{w}_f^\top T^{r'_1}$ , which contradicts with the assumed unique plane condition. We thus have that  $r_1$  must be an analytic cell.

Let  $\mathbf{n}_1 \propto \mathbf{a}_F^{\tilde{r}_1}$  of a certain analytic cell  $\tilde{r}_1 \in \tilde{\mathcal{R}}$  (or  $\mathcal{C}_F^{\tilde{r}_1}$ ), and we have the analytic face  $\mathcal{P}_F^{\tilde{r}_1} \subseteq \mathcal{P}_1$ . Assume there exist  $\mathcal{P}_1 - \mathcal{P}_F^{\tilde{r}_1} = \{\mathbf{x} \in \mathcal{Z} | \mathbf{x} \in \mathcal{P}_1, \mathbf{x} \notin \mathcal{P}_F^{\tilde{r}_1}\}$ , which means that for any  $\mathbf{x} \in \mathcal{P}_1 - \mathcal{P}_F^{\tilde{r}_1}$ , it resides in an analytic face  $\mathcal{P}_F^{\tilde{r}'_1}$  of a different cell  $\tilde{r}'_1$ ; since  $\mathbf{x} \in \mathcal{P}_1$ , we have  $\mathbf{n}_1 \propto \mathbf{a}_F^{\tilde{r}'_1}$  and thus  $\mathbf{a}_F^{\tilde{r}_1} \propto \mathbf{a}_F^{\tilde{r}'_1}$ , which contradicts with the unique plane condition of  $\mathbf{a}_F^{\tilde{r}_1} \not\propto \mathbf{a}_F^{\tilde{r}'_1}$ . We thus have  $\mathcal{P}_1 = \mathcal{P}_F^{\tilde{r}_1}$  and  $\mathcal{P}_1 \subset \mathcal{C}_F^{\tilde{r}_1}$ . By the definition (15) of analytic face, the above argument also tells that planar faces on  $\mathcal{Z}$  and analytic faces  $\{\mathcal{C}_F^{\tilde{r}} | \tilde{r} \in \tilde{\mathcal{R}}\}$  are one-to-one corresponded.

Assume  $\mathcal{P}_1$  connects with another planar face  $\mathcal{P}_2$  on a shared edge segment  $\mathcal{E} = \{\mathbf{x} \in \mathcal{Z} | \mathbf{x} \in \mathcal{P}_1, \mathbf{x} \in \mathcal{P}_2\}$ . Define the normal of  $\mathcal{P}_2$  as  $\mathbf{n}_2 \in \mathbb{R}^3$ , we have  $\mathbf{n}_1 \not\propto \mathbf{n}_2$ . Let  $\mathcal{P}_2 \subset \mathcal{C}_F^{\tilde{r}_2}$ , and we thus have  $\mathcal{E} \subset \mathcal{C}_F^{\tilde{r}_1}$  and  $\mathcal{E} \subset \mathcal{C}_F^{\tilde{r}_2}$ , which tells that the two cells  $\mathcal{C}_F^{\tilde{r}_1}$  and  $\mathcal{C}_F^{\tilde{r}_2}$  connect at least on  $\mathcal{E}$ . Due to the monotonous and convex nature of linear analytic cells  $\{\mathcal{C}_F^{\tilde{r}} | \tilde{r} \in \tilde{\mathcal{R}}\}$ ,  $\mathcal{E}$  must be on the boundaries of both  $\mathcal{C}_F^{\tilde{r}_1}$  and  $\mathcal{C}_F^{\tilde{r}_2}$ , and the boundaries of  $\mathcal{C}_F^{\tilde{r}_1}$  and  $\mathcal{C}_F^{\tilde{r}_2}$  share at least on  $\mathcal{E}$ . There exist two cases for the connection of cell boundaries on  $\mathcal{E}$ : 1) in the general case,  $\mathcal{C}_F^{\tilde{r}_1}$  and  $\mathcal{C}_F^{\tilde{r}_2}$  share a boundary  $\mathcal{B}_F^{\tilde{r}_1 \tilde{r}_2}$  defined by a hyperplane  $\mathbf{H}_{lk}^{\tilde{r}_1 \tilde{r}_2} = \{\mathbf{x} \in \mathbb{R}^3 | \mathbf{a}_{lk}^{\tilde{r}_1 \tilde{r}_2} \mathbf{x} = 0\}$ , and we have  $\mathcal{E} \in \mathcal{B}_F^{\tilde{r}_1 \tilde{r}_2}$ , which, based on Corollary 5 and Definition 2, suggests that the two cells have a switching neuron state  $s_{lk}(\mathbf{x}) \forall \mathbf{x} \in \mathcal{B}_F^{\tilde{r}_1 \tilde{r}_2}$ , and consequently a switching neuron state  $s_{lk}(\mathbf{x}) \forall \mathbf{x} \in \mathcal{E}$ ; 2) in some rare case,  $\mathcal{E}$  coincides with a cell edge of  $\mathcal{C}_F^{\tilde{r}_1}$  defined by  $\{\mathbf{x} \in \mathbb{R}^3 | \mathbf{a}_{l_1 k_1}^{\tilde{r}_1} \mathbf{x} = 0, \mathbf{a}_{l'_1 k'_1}^{\tilde{r}_1} \mathbf{x} = 0\}$ , and a cell edge of  $\mathcal{C}_F^{\tilde{r}_2}$  defined by  $\{\mathbf{x} \in \mathbb{R}^3 | \mathbf{a}_{l_2 k_2}^{\tilde{r}_2} \mathbf{x} = 0, \mathbf{a}_{l'_2 k'_2}^{\tilde{r}_2} \mathbf{x} = 0\}$ , and it is not necessary that  $l_1 k_1$  and  $l_2 k_2$  specify the same neuron, and  $l'_1 k'_1$  and  $l'_2 k'_2$  specify another same neuron. Due to a phenomenon similar to the blessing of (high) dimensionality [32], the second case of coincidence is expected to happen with a low probability. In any of the two cases, the boundaries  $\mathcal{C}_F^{\tilde{r}_1}$  and  $\mathcal{C}_F^{\tilde{r}_2}$  respectively associated with  $\mathcal{P}_1$  and  $\mathcal{P}_2$  connect on  $\mathcal{E}$ .

Since for any pair of planar faces  $\mathcal{P}_1$  and  $\mathcal{P}_2$  connected

on  $\mathcal{Z}$ , we prove that they are uniquely corresponded to a pair of analytic faces  $\mathcal{P}_F^{\tilde{r}_1}$  and  $\mathcal{P}_F^{\tilde{r}_2}$ , which are polygon faces connected via boundaries of their respective analytic cells  $\mathcal{C}_F^{\tilde{r}_1}$  and  $\mathcal{C}_F^{\tilde{r}_2}$ . The theorem is proved.  $\square$

## APPENDIX B

### ALGORITHMIC DETAILS OF PIVOTING ENUMERATION

In this section, we present the Algorithm 2 of pivoting enumeration that greatly improves the efficiency of analytic marching.

## APPENDIX C

### ADDITIONAL RESULTS OF THE RICHLY DETAILED DATASET

We show additional results of direct shape encoding for the Richly Detailed (RD) dataset in Fig. 9.

## APPENDIX D

### QUALITATIVE RESULTS OF NOVEL SHAPE RECONSTRUCTIONS

We present in Fig. 10 and Fig. 11 the qualitative results of analytic marching respectively by learning a global decoder or an ensemble of local decoders for reconstruction of novel shapes. Quantitative results are respectively presented in Section 7.3 and Section 7.4 in the main text.



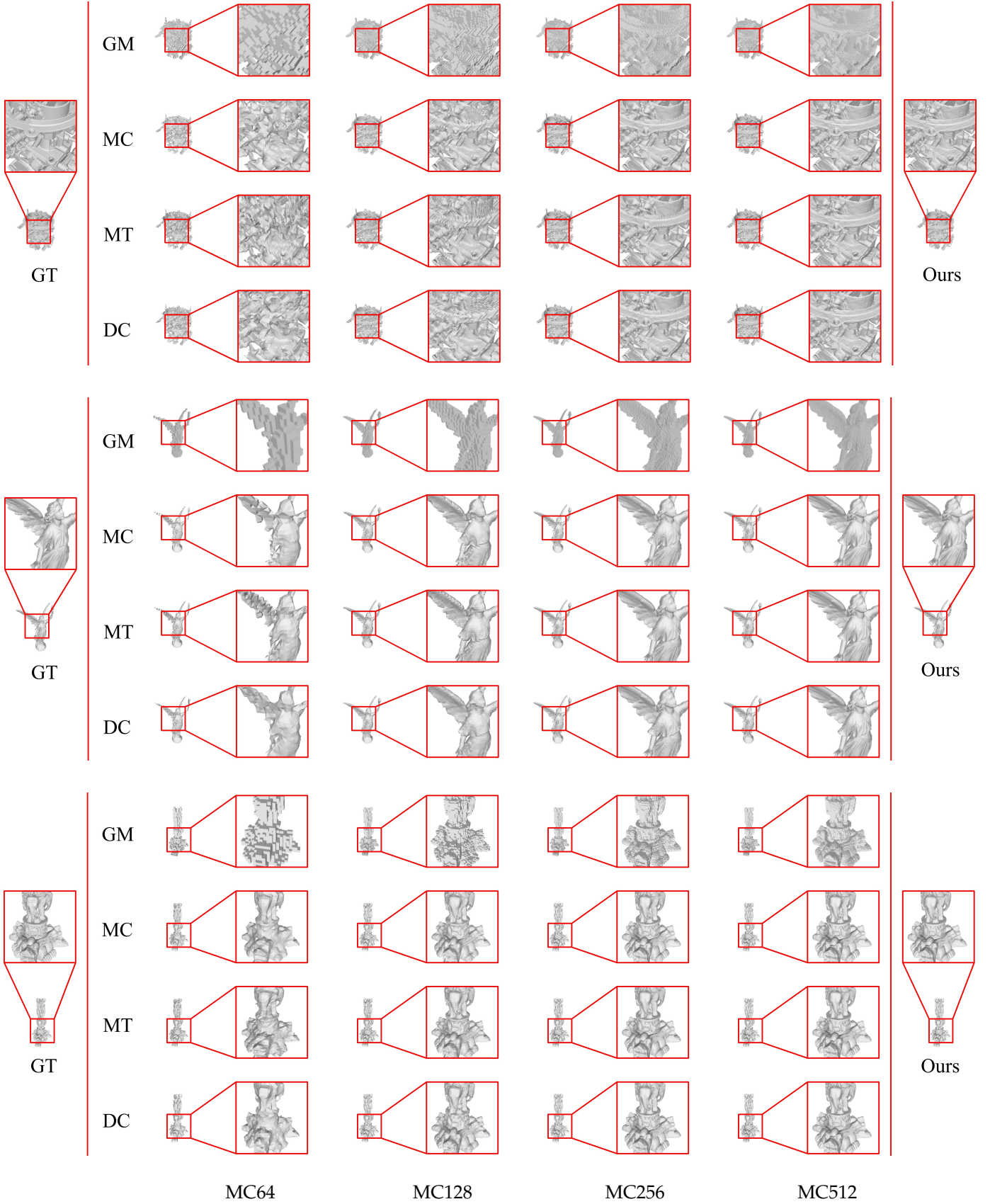


Fig. 9. Qualitative comparisons of direct shape encoding. The shown shape instances are fitted to an MLP of depth 8 and width 60 for SDF modeling. For greedy meshing (GM), marching cubes (MC), marching tetrahedra (MT), and dual contouring (DC), results under a resolution range of discrete point sampling from  $64^3$  to a GPU memory limit of  $512^3$  are presented.

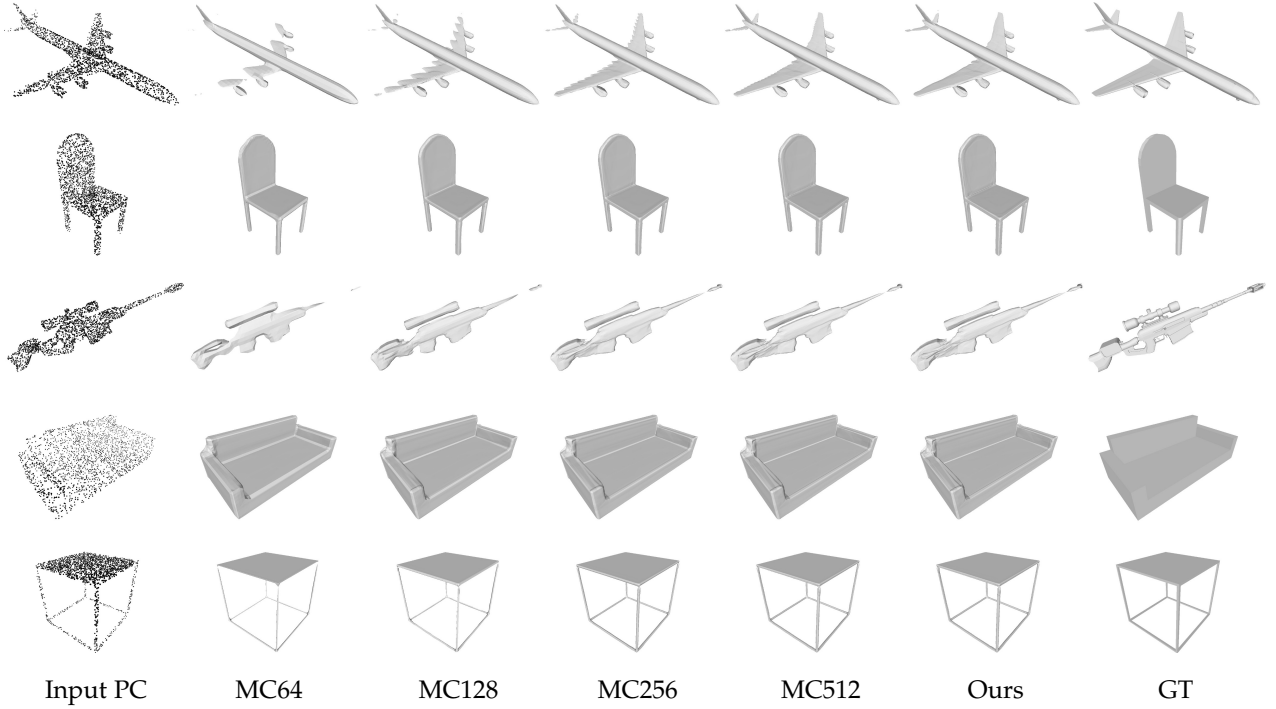


Fig. 10. Qualitative comparisons between analytic marching (AM) and marching cubes (MC) in the context of learning a global model for decoding of novel shape instances. Experiments are conducted on shape instances of five categories from ShapeNet, by training a hypernetwork that gives weights of a light MLP with depth 6 and width 60 for SDF modeling. We show an example from each of the five categories. *Results of AM are after simplification of the originally recovered meshes at a ratio of 10%.* For MC, the sampling resolutions of 3D points range from  $64^3$  to  $512^3$ .

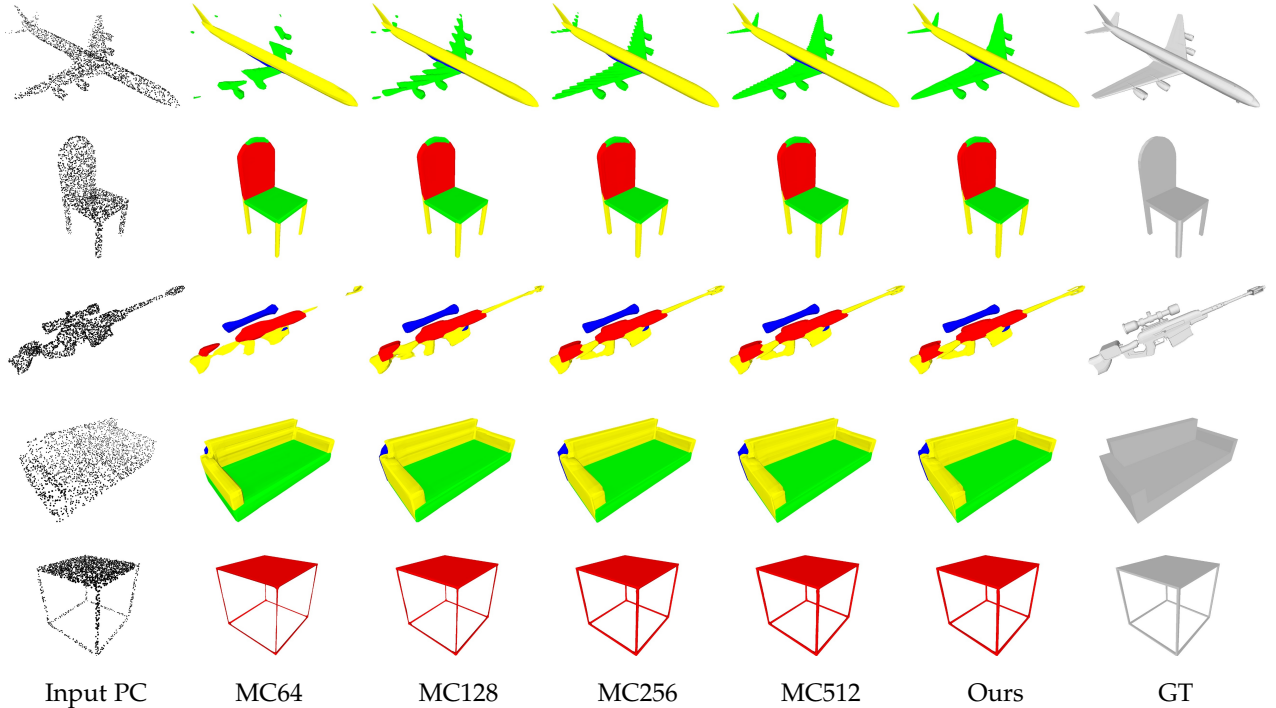


Fig. 11. Qualitative comparisons between analytic marching (AM) and marching cubes (MC) in the context of learning an ensemble of local decoders for reconstructions of novel shape instances. Experiments are conducted on shape instances of five categories from ShapeNet, by training a hypernetwork that gives weights of the ensemble for occupancy modeling; the ensemble is formed by aggregating, via max pooling, outputs of 4 subnetworks, each of which is of depth 4 and width 32. We show an example from each of the five categories. *Results of AM are after simplification of the originally recovered meshes at a ratio of 10%.* For MC, the sampling resolutions of 3D points range from  $64^3$  to  $512^3$ . Different colors indicate the shape components recovered by individual subnetworks.

---

**Algorithm 2** Pivoting enumeration in analytic marching.

---

**INPUT:** A point  $\mathbf{x} \in \mathcal{P}_F^{\tilde{r}}$ ; a hyperplane  $\mathbf{H}_{lk}^{\tilde{r}} \in \{\mathbf{H}_{lk}^{\tilde{r}}\}$  that is a true boundary of the working cell  $\mathcal{C}_F^{\tilde{r}}$ .

**OUTPUT:** A set  $\mathcal{V}_P^{\tilde{r}}$  of ordered vertices.

- 1: Establish an index set  $\mathcal{I}^{\tilde{r}} = \{(l_1, k_1), \dots, (l_N, k_N)\}$  by sorting, in an increasing order, the Euclidean distances from each of the  $N$  hyperplanes in  $\{\mathbf{H}_{lk}^{\tilde{r}}\}$  to the point  $\mathbf{x} \in \mathcal{P}_F^{\tilde{r}}$ , where  $N = n_1 + \dots + n_L$  is the total number of neurons in  $\mathbf{T}$ .
  - 2: Introduce the auxiliary  $t, t', t'' \in \mathbb{N}$ ; use  $(l_t, k_t)$  to index the firstly identified boundary plane  $\mathbf{H}_{lk}^{\tilde{r}}$ , written as  $\mathbf{H}_{l_t k_t}^{\tilde{r}}$  (this same boundary is denoted as  $(l_*, k_*)$  and  $\mathbf{H}_{l_* k_*}^{\tilde{r}}$  for a later reference). Let  $\mathcal{I}_{/t}^{\tilde{r}} = \{(l_1, k_1), \dots, (l_{t-1}, k_{t-1}), (l_{t+1}, k_{t+1}), \dots, (l_N, k_N)\}$ , and  $\mathcal{I}_{/t'/t}^{\tilde{r}} = \{(l_1, k_1), \dots, (l_{t'-1}, k_{t'-1}), (l_{t'+1}, k_{t'+1}), \dots, (l_{t-1}, k_{t-1}), (l_{t+1}, k_{t+1}), \dots, (l_N, k_N)\}$ .
  - 3: Initialize  $\mathcal{V}_P^{\tilde{r}} = \emptyset$  and  $t'' = 1$ .
  - 4: **while**  $\mathcal{V}_P^{\tilde{r}} = \emptyset$  **do**
  - 5:   Solve the system of equations  $[\mathbf{a}_{l_t k_t}^{\tilde{r}}; \mathbf{a}_{l_{t''} k_{t''}}^{\tilde{r}}; \mathbf{a}_F^{\tilde{r}}] \mathbf{x} = \mathbf{0}$  to have a vertex candidate  $\mathbf{v} \in \mathbb{R}^3$ .
  - 6:   **if**  $\mathbf{v}$  satisfies condition (13) **then**
  - 7:     Push  $\mathbf{v}$  into  $\mathcal{V}_P^{\tilde{r}}$ .
  - 8:     Update  $(l_{t'}, k_{t'}) = (l_t, k_t)$ .
  - 9:     Update  $(l_t, k_t) = (l_{t''}, k_{t''})$ .
  - 10:   **else**
  - 11:     Update  $t'' \leftarrow t'' + 1$  upon  $(l_{t''+1}, k_{t''+1}) \in \mathcal{I}_{/t}^{\tilde{r}}$  or  $t'' \leftarrow t'' + 2$  upon  $(l_{t''+2}, k_{t''+2}) \in \mathcal{I}_{/t}^{\tilde{r}}$ .
  - 12:   **end if**
  - 13: **end while**
  - 14: **while true do**
  - 15:   Solve the system of equations  $[\mathbf{a}_{l_t k_t}^{\tilde{r}}; \mathbf{a}_{l_{t''} k_{t''}}^{\tilde{r}}; \mathbf{a}_F^{\tilde{r}}] \mathbf{x} = \mathbf{0}$  to have a vertex candidate  $\mathbf{v} \in \mathbb{R}^3$ .
  - 16:   **if**  $\mathbf{v}$  satisfies condition (13) **then**
  - 17:     Push  $\mathbf{v}$  into  $\mathcal{V}_P^{\tilde{r}}$ .
  - 18:     **if**  $(l_{t''}, k_{t''})$  indexes  $\mathbf{H}_{l_* k_*}^{\tilde{r}}$  **then**
  - 19:       **break**
  - 20:     **else**
  - 21:       Update  $(l_{t'}, k_{t'}) = (l_t, k_t)$ .
  - 22:       Update  $(l_t, k_t) = (l_{t''}, k_{t''})$ .
  - 23:       Update  $\mathcal{I}_{/t'/t}^{\tilde{r}} = \{(l_1, k_1), \dots, (l_{t'-1}, k_{t'-1}), (l_{t'+1}, k_{t'+1}), \dots, (l_{t-1}, k_{t-1}), (l_{t+1}, k_{t+1}), \dots, (l_N, k_N)\}$ .
  - 24:       Reset  $t'' = 1$ .
  - 25:     **end if**
  - 26:     **else**
  - 27:       Update  $t'' \leftarrow t'' + 1$  upon  $(l_{t''+1}, k_{t''+1}) \in \mathcal{I}_{/t'/t}^{\tilde{r}}$  or  $t'' \leftarrow t'' + 2$  upon  $(l_{t''+2}, k_{t''+2}) \in \mathcal{I}_{/t'/t}^{\tilde{r}}$ .
  - 28:     **end if**
  - 29: **end while**
-