Implicit Conversion of Manifold B-Rep Solids by Neural Halfspace Representation

HAO-XIANG GUO, Tsinghua University, P.R. China YANG LIU, Microsoft Research Asia, P. R. China HAO PAN, Microsoft Research Asia, P. R. China BAINING GUO, Microsoft Research Asia, P. R. China



Fig. 1. Left: Conversion of a manifold B-Rep solid model S to the neural halfspace representation. NH-Rep is a function Boolean tree, in which each leaf node corresponds to a B-Rep (sub)patch and associates with a neural implicit function f_i . The zero isosurface of the Boolean expression h(x) at the tree root separates the interior and the exterior space of S and represents the boundary surface of S and its sharp features faithfully. h(x) also approximates the signed distance field of S, as seen from the sliced images of h(x). **Right**: Sample applications supported by NH-Rep: sharp shape offsetting using different blending radii (middle); CSG operation on two NH-Reps (bottom).

We present a novel implicit representation — *neural halfspace representation* (NH-Rep), to convert manifold B-Rep solids to implicit representations. NH-Rep is a Boolean tree built on a set of implicit functions represented by the neural network, and the composite Boolean function is capable of representing solid geometry while preserving sharp features. We propose an efficient algorithm to extract the Boolean tree from a manifold B-Rep solid and devise a neural network-based optimization approach to compute the implicit functions. We demonstrate the high quality offered by our conversion algorithm on ten thousand manifold B-Rep CAD models that contain various curved patches including NURBS, and the superiority of our learning approach over other representative implicit conversion algorithms in terms of surface reconstruction, sharp feature preservation, signed distance field approximation, and robustness to various surface geometry, as well as a set of applications supported by NH-Rep.

$\label{eq:CCS} Concepts: \bullet Computing methodologies \to Volumetric models; Parametric curve and surface models; Neural networks; Representation of Boolean functions.$

Additional Key Words and Phrases: B-Rep solid, neural halfspace representation, Boolean tree, implicit conversion

Hao-Xiang Guo (work done during internship at Microsoft Research Asia), ghx17@mails.tsinghua.edu.cn; Yang Liu (corresponding author), Hao Pan, Baining Guo, {yangliu,haopan,bainguo}@microsoft.com.

© 2022 Association for Computing Machinery.

ACM Reference Format:

Hao-Xiang Guo, Yang Liu, Hao Pan, and Baining Guo. 2022. Implicit Conversion of Manifold B-Rep Solids by Neural Halfspace Representation. *ACM Trans. Graph.* 41, 6, Article 276 (December 2022), 16 pages. https://doi.org/10.1145/3550454.3555502

1 INTRODUCTION

Implicit solid representations such as constructive solid geometry (CSG) and halfspace representations have gained popularity in computer graphics and CAD/CAM due to many desirable characteristics: it is easy to perform inside/outside queries; it is unbreakable in the sense that challenging operations (e.g., Boolean, round, offset) can be applied without numerical failure; they support field-driven design that integrates simulation and manufacturing conditions into shape optimization, thus significantly improving the design to manufacturing cycle consistency [Allen 2021]; and it is suitable for leveraging fast-growing multicore CPU and GPU architectures for parallelizable computation. Their ability to model signed distance fields is also demanded by many applications [Chen et al. 2022a]. However, existing CAD solid models are often in the form of boundary representations (B-Reps), in which the solid boundary is composed of a set of parameterized surface patches, such as NURBS. To benefit from the aforementioned merits of implicit solid representations, it is necessary to convert B-Reps to usable implicit representations with high fidelity.

This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *ACM Transactions on Graphics*, https://doi.org/10.1145/3550454.3555502.

An early attempt to convert B-Rep solids to halfspace CSG models is limited to convex solids and B-Reps composed of linear and quadratic patches with linear boundary edges [Shapiro and Vossler 1993]. High-order solid patches and nonlinear boundary edges are not supported due to their complexity. Inverse CSG approaches, such as Du *et al.*'s work [2018], are restricted by the limited types of solid primitives; thus, it is not easy to represent free-form geometry and sharp features. Classic implicit reconstruction methods, such as Poisson reconstruction [Kazhdan et al. 2006] and recent neural implicit methods, such as DeepSDF [Park et al. 2019], lack the ability to faithfully represent sharp features. The requirements of supporting freeform geometry, preserving sharp features, and modeling SDFs, are still challenges for the implicit conversion of B-Reps.

To address the above challenges, we propose a novel halfspace representation - Neural Halfspace Representation, abbreviated to NH-Rep, is defined by a Boolean tree-based hierarchy of implicit functions, where the implicit function at a leaf node corresponds to a B-rep (sub)patch, and all implicit functions are modeled via a multilayer perceptron (MLP) network. The composite function $h(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^3 \longmapsto \mathbb{R}$ by NH-Rep is an implicit representation of a manifold B-Rep solid. Here, the specific construction of the Boolean tree offers the ability to model sharp features in an implicit form, and the use of MLP to represent implicit functions provides a great fitting capability to various surface geometry. NH-Rep has nice properties: (1) the zero isosurface h(x) = 0 separates the interior and exterior of the input B-Rep solid; (2) h(x) = 0 fits the freeform geometry of the input B-Rep solid tightly with normal agreement and sharp feature preservation; (3) the SDF of the B-Rep solid can be approximated by h(x) with good quality, by imposing the eikonal equation on h(x). An example is illustrated in Fig. 1-left.

Based on the theoretical proof of the existence of NH-Rep for a given manifold B-Rep solid, we develop a method for computing NH-Reps, including an efficient Boolean tree construction algorithm and a learning approach to compute implicit functions. We performed a large-scale benchmark on 10,000 manifold B-Rep CAD models to validate the efficacy and robustness of our method. We verified the superiority of NH-Rep over other representative implicit conversion and reconstruction methods in terms of solid approximation fidelity, sharp feature preservation, and robustness to various solid geometries. Our code is available at https://github.com/guohaoxiang/NH-Rep for facilitating future research and applications.

2 RELATED WORK

Solid representation. Boundary representation (B-Rep) and implicit representation (Imp-Rep) are two common solid representations [Shapiro 2002]. B-Rep represents a solid as a collection of surface, curve, and point elements. The surface and curve elements can be in parametric forms like NURBS, or polygonal forms like triangle facets and polylines. It is easy to model and edit B-Rep models due to their explicit formulation. Imp-Reps such as Constructive Solid Geometry (CSG) [Ricci 1973] and Function Representation (F-Rep) [Pasko et al. 1995] provide a constructive modeling scheme and possess the advantages mentioned in Section 1. The former applies Boolean operations to primitive solids; the latter can employ more general R-functions [Shapiro 2007] based on halfspace functions for modeling and is also suitable for modeling complex shapes in a procedural manner. Our NH-Rep belongs to the category of F-Rep.

Representation conversion. To take advantage of both B-Rep and implicit representations, various applications require conversion between these two representations. The study of converting implicit representations to B-Reps is mature and many techniques are available, such as polygonalization [Lorensen and Cline 1987] and parameterization of algebraic surfaces. However, the opposite conversion is challenging. The difficulties of converting a B-Rep to a halfspace representation were explored by [Buchele and Crawford 2003; Shapiro and Vossler 1991a, b, 1993]: additional separating halfspaces are required in many circumstances. Shapiro et al. convert B-Reps bounded with linear and quadratic patches by using linear separators, but cannot handle high-order patches and nonlinear patch edges. For B-Rep models consisting of simple primitives such as cubes, spheres, and cylinders, a series of CSG inversion works are proposed: binary optimization [Wu et al. 2018], program synthesizer [Du et al. 2018; Xu et al. 2021], evolutionary algorithm [Friedrich et al. 2019], and learning from data [Kania et al. 2020; Ren et al. 2021; Sharma et al. 2018; Yu et al. 2022]; but they cannot accurately represent freeform geometry due to their restricted primitive types. Boundary-sampled halfspace (BSH) [Du et al. 2021] defines the shape as sparsely placed samples on the halfspace boundaries and provides greater agility and expressiveness than CSG for shape modeling, and it also simplifies the conversion process but has challenges in handling tangentially contacted surface patches, which are common in CAD models. A simple failure case is provided in Appendix A.

Reconstruction-based implicit conversion. By treating a B-Rep model as a collection of densely sampled points, many implicit-based reconstruction techniques [Berger et al. 2017; Carr et al. 2001; Ohtake et al. 2003] can be used to convert point clouds into implicit representations. For recovering shape features, Kazhdan et al. [2013] use an indicator field to represent a 3D shape and approximate features by penalizing the difference between the surface gradients and the oriented point normals. Oztireli et al. [2009] uses implicit moving least squares (MLS) to represent 3D shapes while estimating adaptive functional weights to handle features and outliers. However, these methods cannot model C^1 discontinuous features accurately, as the underlying surface representation is globally smooth, not piecewise smooth. Recent neural implicit approaches [Chabra et al. 2020; Chen and Zhang 2019; Chibane et al. 2020; Jiang et al. 2020; Liu et al. 2021; Park et al. 2019; Peng et al. 2020; Yang et al. 2021] represent the shape geometry as a signed distance field or an occupancy field, and some works [Gropp et al. 2020; Sitzmann et al. 2020; Williams et al. 2021] overfit the input point cloud to achieve high accuracy, but the learned function is either piecewise linear when using MLPs with ReLU [Lei and Jia 2020] or globally smooth; therefore, it cannot model (curved) sharp features faithfully. The works of BSP-Net [Chen et al. 2020] and CVXNet [Deng et al. 2020] infer a Boolean CSG tree of a set of planes to represent a 3D shape in a simple and compact form. However, their approximation quality to the input is restricted by the plane geometry and the number of

planes. By utilizing the B-Rep patch information explicitly and composing the learned neural implicit functions with a Boolean tree, we obtain faithful implicit conversion with sharp feature preservation.

Neural B-Rep representation. The complex data structure of B-Rep representations poses a challenge in integrating B-Rep with modern machine learning techniques. Several approaches have been developed to address this challenge. UV-Net [Jayaraman et al. 2021] converts B-Rep to a face-adjacency graph in which the face and edge correspond to a parametric surface and curve, respectively. 2D CNN and 1D CNN are performed in the UV domain of each face and edge, and the learned features are aggregated by graph convolution. BRepNet [Lambourne et al. 2021] learns the features on B-Rep faces, edges, loops, coedges, and vertices by graph convolution. These works are mainly designed for shape analysis tasks, such as classification and segmentation. Recently Willis et al. [2022] associate the one-hot feature (element type) for B-Rep faces and edges, and pass messages through the B-Rep graph to predict the joint information for part assembly. Wu et al. [2021] proposes a generative network to infer a series of CAD commands to create or reconstruct CAD models. However, its CAD reconstruction capability from point clouds is limited by the model complexity and the predefined commands. Our approach utilizes the patch adjacency of B-Rep to deduce the Boolean tree for converting B-Rep to implicit representation and has good scalability to deal with complex models.

Feature-sensitive distance fields. Storing distance field values in discrete grids [Jones et al. 2006] is a popular way to convert explicit models to implicit representation. Adaptive octree [Frisken et al. 2000] and hp-refinement [Koschier et al. 2016] are common techniques to make storage compact while maintaining sufficient accuracy. To preserve sharp corners and sharp features, additional information and operations are required to store, such as *directed* distances in the x-, y- and z-direction [Kobbelt et al. 2001], exact intersection points and normals [Ju et al. 2002], the nearest triangle information of the grid points to the input mesh [Huang et al. 2001], offset distance fields [Qu et al. 2004], and density gradient [Novotný and Srámek 2005]. To represent high-quality surface details and curved sharp features, high-resolution (adaptive) grids are needed in all these approaches and take up large storage spaces. In contrast, the neural implicit representation can approximate the signed distance field and achieves a good balance between storage efficiency and approximation accuracy; furthermore, the Boolean operations of NH-Rep ensure faithful feature preservation.

3 OVERVIEW

Our paper is organized as follows. In Section 4.2, we first introduce the terminologies of boundary representation and halfspace representation, then present our solution for converting manifold B-Rep solids to halfspace representation — Neural Halfspace Representation (NH-Rep), which is built on a special Boolean tree of a set of implicit functions. We develop a Boolean tree construction algorithm (Section 5) and a neural network-based optimization algorithm to determine the implicit functions associated with the Boolean tree (Section 6). In Section 7, we provide extensive experimental analysis and ablation studies to verify the efficiency and superiority of our approach. In Section 8, we demonstrate a series of applications supported by NH-Rep.

4 NEURAL HALFSPACE REPRESENTATION

4.1 Boundary representation and halfspace representation

Boundary representation. A B-Rep solid is a 3D volume surrounded by a collection of non-overlapped manifold surface patches. For a solid S with L patches, we denote these patches as $\mathcal{P}_1, \ldots, \mathcal{P}_L$, and the boundary of S as ∂S which satisfies $\partial S = \bigcup_i \mathcal{P}_i$. Any surface patch of a B-Rep solid can be in the form of a parametric surface or a set of polygonal facets. Without loss of generality, we assume that all surface patches have a consistent normal orientation, pointing outside of the volume. For a B-Rep solid S, ∂S , I(S) and $\mathcal{E}(S)$ denote the boundary surface, the interior space and the exterior space of S, respectively. For convenience, we define $\overline{I}(S) = I(S) \cup \partial S$. In our paper, we assume that ∂S is manifold and the solid volume is not empty.

Feature curves of B-Rep. The boundary of a B-Rep patch is formed by a set of 3D curves, each of which is shared by two adjacent B-Rep patches. We call these curves *feature curves*. The endpoints of a feature curve are called *feature corners* and the other points on the feature curve are called *feature points*. We call a feature point p *con-*



vex, if the interior dihedral angle at p, denoted by $\gamma_{\rm p}$, is smaller than 180°; *concave* if $\gamma_{\rm p} > 180°$; and *smooth* if $\gamma_{\rm p} = 180°$. "Convex" or "concave" means that ∂S is only C^0 smooth at p. Here $\gamma_{\rm p}$ is called *feature angle* at p. We call a feature curve *C convex* if all feature points on *C* are not concave; *concave* if all feature points on *C* are not concave; *concave* if all feature points on *C* are smooth; *hybrid* if *C* contains both convex and concave feature points. We say a feature curve *C sharp* if $\max_{p \in C} |180° - \gamma_p| \ge \delta_s, \delta_s > 0$ is the user-defined sharp angle threshold. The right inset illustrates a B-Rep solid with different types of feature curves: convex in black, concave in red, and smooth in blue.

Halfspace representation. An implicit function $f : x \in \mathbb{R}^3 \mapsto \mathbb{R}$ defines a halfspace: $\{x \in \mathbb{R}^3 : f(x) \le 0\}$. For simplicity, we abbreviate the notation as $f \leq 0$. Boolean operations (union, intersection, subtraction) can be defined over halfspaces using max and min functions as follows. The union of a set of halfspaces $\{f_i \leq 0\}_{i=1}^m$ is $\{\mathbf{x} \in \mathbb{R}^3 : \min(f_1(\mathbf{x}), \cdots, f_m(\mathbf{x})) \leq 0\}$; their intersection is $\{\mathbf{x} \in \mathbb{R}^3 : \max(f_1(\mathbf{x}), \dots, f_m(\mathbf{x})) \le 0\};$ the difference of two halfspaces $f \le 0$ and $g \le 0$ is $\{x \in \mathbb{R}^3 : \max(f(x), -g(x)) \le 0\}$. By compositing some Boolean operations on halfspaces, a Boolean expression, or called CSG expression, is obtained. A Boolean expression can be rewritten in Boolean tree format: (1) each leaf node stores an implicit function; (2) each non-leaf node stores a Boolean operation: max or min, which is applied to its child nodes; (3) the composite function at the root node, denoted by h(x), is the Boolean expression. For convenience, we denote the Boolean operation on the non-leaf tree node r as op(r), and use $op^+(r)$ to denote the opposite operation of op(r), *i.e.*, $op^+(r) := max$, if op(r) = min; $op^+(r) := min$ if op(r) = max.



Fig. 2. (a): A 2D B-Rep solid (shaded region) surrounded by eight curve segments $\{\mathcal{P}_i\}_{i=1}^8$. \mathcal{P}_i corresponds to an implicit function f_i whose zero isocurve c_i is depicted as a dash line. (b): The Boolean tree of (a). $h = \max(f_1, f_2, f_3, f_4, \min(f_5, f_8, \max(f_6, f_7)))$. The red curves in (c), (d), and (e) are the isocurves of the Boolean expression g_1, g_2 and h, respectively. $h(\mathbf{x}) = 0$ recovers the boundary of the B-Rep solid exactly.

A Boolean tree is a halfspace representation of a B-Rep solid if its composite function at the tree root can classify the point membership of S correctly, *i.e.*, holding the following properties:

276:4

• Guo, et al.

$$\begin{cases} h(\mathbf{x}) < 0, & \forall \mathbf{x} \in I(\mathcal{S}); \\ h(\mathbf{x}) > 0, & \forall \mathbf{x} \in \mathcal{E}(\mathcal{S}); \\ h(\mathbf{x}) = 0, & \forall \mathbf{x} \in \partial \mathcal{S}. \end{cases}$$
(1)

Fig. 2-(b) provides a Boolean tree example that can represent a 2D B-Rep solid shown in Fig. 2-(a). In Fig. 2-(c,d,e), the zero isocurves of the Boolean expressions on the tree nodes are also illustrated.

B-Rep to halfspace representation. Any surface patch of a B-Rep solid can be represented as a subset of the zero isosurface of an implicit function. All implicit functions associated with B-Rep patches define a set of halfspaces. For a solid S with L patches, all halfspaces partition \mathbb{R}^3 into 2^L cells (some are possibly empty). The describability theorem [Shapiro and Vossler 1993] states that there exists a Boolean expression on these halfspaces that can represent S if and only if for each nonempty cell, all points in the cell have the same point membership classification with respect to the solid volume. This condition implies Eq. (1), but is not easy to satisfy in general, except for convex solids. Shapiro and Vossler [1993] proposed adding separators, i.e., additional halfspaces, to partition those cells violating the condition until the mixed regions are separated. Linear separators for B-Rep solids composed of linear and quadratic patches were studied in their work. Nonlinear separators are essential for complicated B-Rep solids but their constructions are unknown. Our work provides an explicit way to compute the Boolean expression and the halfspace functions for general B-Rep solids and introduces nonlinear separators via patch decomposition (Section 5.2) when necessary.

4.2 Neural halfspace representation

We present a special halfspace representation — *neural halfspace representation* to convert an arbitrary B-Rep solid to an implicit presentation. To facilitate the definition of *neural halfspace representation*, we first introduce the following useful terminologies.

Definition 4.1. For a B-Rep solid S with L patches, a set of subpatches $\{s_1, \ldots, s_n\}$ is called a *decomposed patch set* of S if (1) $\bigcup_{i=1}^n s_i = \partial S$; (2) $\forall i, \emptyset \neq s_i \subseteq \mathcal{P}_k, \exists k \in \{1, \cdots, L\}$; (3) $\operatorname{area}(s_i \cap s_j) = 0, \forall i \neq j$.

Definition 4.2. A Boolean tree \mathcal{T} built on a set of implicit functions $\{f_1, \dots, f_n\}$ is called a *patch-based Boolean tree*, if there exists a

decomposed patch set $\{s_1, \ldots, s_n\}$ of S such that s_i lies on the zero surface of f_i , *i.e.*, $s_i \subseteq \{x \in \mathbb{R}^3 : f_i(x) = 0\}, \forall i$.

The following theorem guarantees that any manifold B-Rep solid can be converted to a halfspace representation.

THEOREM 4.3. For a B-Rep solid S, there exists a patch-based Boolean tree T to represent S in half-space representation, i.e., the Boolean expression of T satisfies Eq. (1).

The proof of the above theorem includes two parts, sketched as follows. We first provide an explicit way to build the tree structure of \mathcal{T} and determine the decomposed patch set explicitly, then prove that there exists a set of implicit functions of \mathcal{T} whose composition function at the root node fulfills Eq. (1). We detail the first part in Section 5 and leave the proof to Appendix B.

With the guarantee provided by the above theorem and the built patch-based Boolean tree structure, the computation of implicit functions of \mathcal{T} is formulated as an optimization problem. To maximize the fitting ability of implicit functions to various and complex B-Rep patch geometry, we represent all implicit functions by a neural network, which maps a 3D point to an *n*-dimensional vector in which each entry is the value of an implicit function evaluated at the given point. The optimization details are presented in Section 6.

As we use neural networks to represent implicit functions of the patch-based Boolean tree, we name the halfspace representation under this setup by *neural halfspace representation*.

5 BOOLEAN TREE STRUCTURE CONSTRUCTION

5.1 Algorithm overview

The design of Boolean tree structures is based on the following intuitive idea. Notice that the Boolean operations on halfspaces have clear geometry meaning: union, intersection, and subtraction; we propose to partition the space via these operations defined over halfspaces progressively, until one of the partition cells matches the volume of the input B-Rep, *i.e.*, satisfying the describability theorem [Shapiro and Vossler 1993].

We sketch the above idea as follows, using a 2D example shown in Fig. 3. The initial partition space, denoted by S_p , is the full space. We first select a maximal group of patches, each of which shares convex feature curves with its neighbor patches, *i.e.*, c_1, c_2, c_3, c_4 . We assume that by defining the appropriate implicit functions for these selected patches, the intersection of their halfspaces can contain $\overline{I}(S)$. We update S_p with this intersected region. We then select a maximal group of patches from the rest of the unprocessed patches,



Fig. 3. 2D illustration of Boolean tree construction. (a): An input B-Rep solid. The zero isocurves of the implicit functions are illustrated, same as Fig. 2-(a). (b)-(d) are the intermediate partition results. (b): The intersection of $\{f_1 \leq 0, f_2 \leq 0, f_3 \leq 0, f_4 \leq 0\}$. The corresponding Boolean expression is $\max(f_1, f_2, f_3, f_4)$. (c): The union of $f_5 \geq 0$ and $f_8 \leq 0$ is intersected with the space defined in (b). The corresponding Boolean expression is $\max(f_1, f_2, f_3, f_4, \min(f_5, f_8))$. (d): The region surrounded by c_6, c_7 is added to the partition. The corresponding Boolean expression is $\max(f_1, f_2, f_3, f_4, \min(f_5, f_8))$. (d): The region surrounded by c_6, c_7 is added to the partition. The corresponding Boolean expression is $\max(f_1, f_2, f_3, f_4, \min(f_5, f_8, \max(f_6, f_7)))$, yielding the Boolean tree shown in Fig. 2-(b).

each of which shares concave feature curves with its unprocessed neighbor patches, *i.e.*, c_5 , c_8 . We also assume that, by defining the appropriate implicit functions on c_5 , c_8 , the union of their corresponding halfspaces can be intersected with S_p , and these patches become part of the boundary of the intersected region. S_p is then updated by this new intersected region. Union and intersection operations can be executed alternatively until all surface patches are processed. Finally, S_p matches the volume of the input B-Rep solid, as S_p is tightly surrounded by all B-Rep patches. Fig. 3-(b-d) illustrates each step. In the next subsection, we turn the above idea into a rigorous and executable algorithm.

5.2 Tree structure construction

For a B-Rep solid S, we provide a recursive approach to construct a patch-based Boolean tree. It includes two steps: *tree initialization* and *tree node creation*, and it relies on the following graph structure.

Patch graph. We define an undirected multigraph according to the adjacency of all surface patches of S. Each graph vertex corresponds to a surface patch; each graph edge corresponds to a feature curve shared by two adjacent patches, and each graph edge e is labeled with its corresponding feature curve type, denoted by type(e). This multigraph is called *patch graph*, denoted by G. Fig. 4-(b) depicts the patch graph of the B-Rep model shown in Fig. 4-(a).

Tree initialization. If \mathcal{G} contains multiple *maximal connected subgraphs* (in short, MC subgraph), *i.e.*, the volume of \mathcal{S} contains multiple disconnected components, a tree root node r is created, and each MC subgraph will be used to create child nodes under the root. The Boolean operation on r is set to min, to combine all components. If \mathcal{G} contains only one MC subgraph, we set r as a virtual node: $r = \emptyset$ and op(r) = min. The virtual node does not appear in the final Boolean tree structure.

We take each MC-subgraph and the tree root as input to the following recursive tree node creation algorithm.

Tree node creation. This step takes an MC-subgraph \mathcal{G}' and a parent tree node r as input. From \mathcal{G}' , we select graph vertices that have no connected graph edges with label edgetype(op⁺(r)), to form a vertex set Q. Here, we define edgetype(max) := convex,

edgetype(min) := concave. As the graph vertex is dual to a surface patch, this selection does the following job: (1) when op(r) = min, Q collects those surface patches in G' that have no adjacent patches in G' or only share convex or smooth feature curves with other adjacent patches in G'; (2) when op(r) = max, Q collects those surface patches in G' that have no adjacent patches in G' or only share concave or smooth feature curves with other adjacent patches in G'. The formed Q is processed as follows.

Case 1: $Q = \emptyset$. There are two subcases that Q can be empty. The first subcase occurs when r is the root node associated with op(r) = max operation, but *S* does not contain any convex feature edges. For this special subcase (depicted in Fig. 5), we only need to set op(r) = min and construct the Boolean tree directly. The second subcase occurs when any patch in \mathcal{G}' has both convex and concave features shared with adjacent patches. Fig. 6-(b) depicts such an MC-subgraph in this situation. For this subcase, we randomly select one patch from \mathcal{G}' , denoted by \mathcal{P} , and decompose it into a set of subpatches, such that the boundary curves of any subpatch do not contain convex and concave feature curve segments of ${\mathcal P}$ simultaneously. We call this step patch decomposition. Fig. 6-(c) illustrates the patch decomposition step. We replace $\mathcal P$ with these subpatches and update \mathcal{G}' accordingly. Here, any graph edge connecting with two adjacent subpatches is labeled *smooth*. We then recompute Qfrom the updated G' and execute the tree node creation algorithm.

Case 2: $Q \neq \emptyset$. A child node r' is created under r, and the Boolean operator at r' is set to the opposite operation of r, *i.e.*, op(r') := op⁺(r). For each graph vertex $v \in Q$, a child node is created under r'. These added child nodes are tree leaf nodes because each of them corresponds to a surface (sub)patch. G' is updated by removing the vertices of Q and their connected edges from G'. Then we process every MC-subgraph in the updated G' by feeding it and r to our tree node creation algorithm.

The pseudocode of the above recursive node creation algorithm is provided in Algorithm 1. Fig. 4 illustrates how the Boolean tree is created for a 2D B-Rep solid, step by step.

Termination of tree construction. Since \mathcal{G} has finite graph vertices and patch decomposition always reduces the number of graph nodes that connect with convex and concave edges, the recursive algorithm ends in finite steps. The subpatches from patch decomposition and the original non-decomposed patches form the final decomposed patch set, and each leaf node corresponds to one of the (sub)patches. The tree node creation step ensures that max and min appear alternately in different layers of tree nodes. Thus, the Boolean tree created is a patch-based Boolean tree.

Implementation of patch decomposition. We implement patch decomposition as follows. For a surface patch in polygonal mesh format, we first employ face-splitting to ensure that no triangle contains feature edge segments with different feature types. Then, on the dual graph of the mesh, we formulate a facet labeling problem such that the facets with the same label do not contain convex and concave feature edges simultaneously. The problem is solved using the Min-Cut algorithm. The labeling result induces different subpatches. For a patch in parametric surface format such as NURBS, we need to discrete it first as a triangular mesh and then compute 276:6 • Guo, et al.



Fig. 4. Illustration of Boolean tree construction for a 2D B-Rep solid. (a): An input 2D B-Rep solid with eight curve patches. A patch graph is created first as shown in (b). Graph vertex v_i corresponds to patch \mathcal{P}_i , and the color of graph edges encodes the edge types: red (convex), blue (concave). (c)-(g) illustrate the tree node creation step by step. (c), (e), (g) are the Q set in each step; (d) and (f) are the updated patch graph by removing vertices of Q created from previous step. (h), (i) and (j) are the Boolean tree during construction. f_i corresponds to v_i and \mathcal{P}_i .



Fig. 5. A 2D B-Rep solid formed by four patches contains concave features only (left). The operation at the root node should be set to min.



Fig. 6. Illustration of patch decomposition. **Left**: A star-shape B-Rep that contains 12 patches (in different colors). **Middle**: 10 vertical patches form a graph vertex set, triggering the occurrence of the second subcase of $Q = \emptyset$. The yellow patch is chosen for decomposition. **Right**: Subpatches obtained by patch decomposition.

the subpatches. Here, we record the parametric coordinates of each mesh vertex, so that we can access surface points on the parametric surfaces exactly in the later NH-Rep computation stage, without suffering discretization artifacts. Fig. 6 illustrates patch decomposition on a simple example.

6 NH-REP COMPUTATION

With a built patch-based Boolean tree structure, our objective is to compute a set of implicit functions $\{f_i\}_{i=1}^n$ such that the conditions of Eq. (1) can be met. We rewrite Eq. (1) with respect to the decomposition patch set $\{s_1, \ldots, s_n\}$ as follows.

$h(\mathbf{x}) = 0,$	$\forall \mathbf{x} \in \mathbf{s}_i, \forall i;$	(2)

$$h(\mathbf{x}) > 0, \qquad \forall \mathbf{x} \in \mathcal{E}(\mathcal{S});$$
 (3)

$$h(\mathbf{x}) < 0, \qquad \forall \mathbf{x} \in I(\mathcal{S});$$
 (4)

$$f_i(\mathbf{x}) = 0, \qquad \forall \mathbf{x} \in \mathbf{s}_i, \forall i;$$
 (5)

$$\nabla_{\mathbf{x}} f_i(\mathbf{x}) = \mathbf{n}(\mathbf{x}), \qquad \forall \mathbf{x} \in \mathbf{s}_i, \forall i.$$
 (6)

ACM Trans. Graph., Vol. 41, No. 6, Article 276. Publication date: December 2022.

Algorithm 1: ConstructTreeNode

Input :patch graph G', parent tree node r Result: updated Boolean tree structure Create Q; if $Q \neq \emptyset$ then create child node r' under r and set $op(r') := op^+(r)$; $\forall v \in Q$, create a leaf node under r'; graph update: $G' := G' \setminus Q$; **for** each MC-subgraph $G'' \subseteq G'$ **do** ConstructTreeNode($\mathcal{G}^{\prime\prime}$, r'); end end else if No convex edges in G' then set op(r') = min; $\forall v \in Q$, create a leaf node under r'; end else Pick $v \in \mathcal{G}'$, decompose its corresponding patch \mathcal{P} ; Update G' based on the updated patch layout; ConstructTreeNode(\mathcal{G}' , r); end end

Here, n(x) is the oriented surface normal at x. We impose Eqs. (5) and (6) to ensure that the zero surface of f_i contains s_i and is the first-order approximation of s_i .

We design a learning approach to compute the implicit functions that satisfy the above conditions. We use a multilayer perceptron (MLP) with three hidden layers of size 256 as our network, shown in Fig. 7. It takes a 3D point coordinate x as the input and output *n* values, each of which is the value of f_i at x. These *n* values are passed to the Boolean tree to obtain the composite function value $h(x; \theta)$



Fig. 7. Our network architecture.

(θ denotes the network parameters). Here, we use SoftPlus as the activation function to ensure that MLPs have sufficient smoothness to represent smooth patches. The weight of the network is initialized with geometric initialization [Atzmon and Lipman 2020].

Our loss function consists of the following terms.

Position loss. The position loss E_p is derived from Eqs. (2) and (5):

$$E_p := \frac{1}{N} \sum_{i=1}^{n} \sum_{\mathbf{x} \in \mathbf{s}_i} |f_i(\mathbf{x}; \theta)| + |h(\mathbf{x}; \theta)|.$$
(7)

Here, N is the total number of sample points.

Normal loss. The normal loss E_n is derived from Eq. (6), penalizing the deviation of ∇f_i at the sample points from their ground-truth normals.

$$E_n \coloneqq \frac{1}{N} \sum_{i=1}^n \sum_{\mathbf{x} \in \mathbf{s}_i} \|\nabla_{\mathbf{x}} f_i(\mathbf{x}; \theta) - \mathbf{n}(\mathbf{x})\|.$$
(8)

Eikonal equation loss. We introduce the eikonal equation to approximate the signed distance field of S. The loss E_{eik} [Gropp et al. 2020] encourages h to satisfy the eikonal equation almost everywhere.

$$E_{eik} := \mathbb{E}_{\mathbf{x}}(\|\nabla_{\mathbf{x}}h(\mathbf{x},\theta)\| - 1)^2.$$
(9)

Off-surface loss. E_o is defined to penalize off-surface points whose function values are close to 0 [Sitzmann et al. 2020].

$$E_o := \mathbb{E}_{\mathbf{x}} \left(\exp(-\alpha |h(\mathbf{x}; \theta)|) \right), \ \alpha \gg 1.$$
(10)

Consistency loss. To ensure that $f_i(x; \theta)$ is activated in the Boolean tree when evaluating $h(x; \theta)$ on s_i , we introduce the following term to improve this consistency.

$$E_{cons} := \frac{1}{N} \sum_{i=1}^{n} \sum_{\mathbf{x} \in \mathbf{S}_i} |f_i(\mathbf{x}; \theta) - h(\mathbf{x}; \theta)|.$$
(11)

Correction loss. During training, some sample points may still severely disobey the consistency loss. We define a correction loss E_c to suppress this inconsistency with a high penalty.

$$E_{c} := \frac{1}{|\mathcal{D}|} \sum_{i} \sum_{\mathbf{x} \in \mathbf{s}_{i} \cap \mathcal{D}} \beta |h(\mathbf{x}; \theta) - f_{i}(\mathbf{x}; \theta)|, \ \beta \gg 1.$$
(12)

Here, \mathcal{D} is the point set in which any point violates the constraint: $|f_i(\mathbf{x}; \theta) - h(\mathbf{x}; \theta)| < 10^{-5}$.

The total loss is the sum of the above loss terms, and we set $\alpha = \beta = 100$ empirically. We found that there is no need to add the constraints of Eqs. (3) and (4) as loss terms to avoid spurious zero isosurfaces when the off-surface loss and geometric initialization [Atzmon and Lipman 2020] are used. In this way, we can avoid sampling the ground-truth occupancy field for training. This strategy makes our conversion algorithm friendly to the B-Rep models



Fig. 8. B-Rep solid samples in the ABC dataset [Koch et al. 2019].

with imperfections such as unwanted seams caused by exchanging B-Rep data between different software, and extendable to segmented point clouds where accurate interior and exterior information are not available (see Section 7.4 and Section 8).

Patch grouping. A B-Rep solid may contain many but small surface patches, which results in a large number of Boolean tree leaf nodes and implicit functions. We run a standard graph coloring algorithm on the patch graph of the decomposed patch set to group some disjointed patches and set their corresponding neural functions to be the same. Due to the universal approximation ability of MLP, this strategy is practically useful without compromising conversion quality. To limit the complexity of the group geometry, we set the maximum patch number in a group as 6. This grouping strategy reduces the number of implicit functions to around 50% on average in our experiments and results in a smaller network (fewer neurons in the last layer) and a shorter training time (10% faster).

7 EXPERIMENTS AND ANALYSIS

We conducted a series of experiments and ablation studies to evaluate the efficacy and robustness of our approach and its superiority over other alternative methods.

7.1 Experiment setup

Dataset. We choose the ABC dataset [Koch et al. 2019] as our testbed dataset, in which each B-Rep model is a collection of parameterized curve segments and surface patches, including NURBS patches. We choose the first chunk of the ABC dataset that contains 10,000 B-Rep models as our testbed. Some models are illustrated in Fig. 8. We filter out non-manifold and non-closed models, and very simple models like boxes and cylinders. For models with multiple disconnected components, we treat each component as a model instance for conversion, and the resulting NH-Reps can be easily combined via union. The total number of model instances is 10,935. 10 % models contain NURBS patches. Among them, 24 models require patch decomposition during tree construction. We utilized the discretized mesh provided in the ABC dataset for patch decomposition, where the mesh contains u-v coordinates of the parametric patch. The average and maximal surface patch numbers for a model instance are 24.07 and 199, respectively. They are reduced to 10.69 and 144 by our patch grouping algorithm. Fig. 9 illustrate the Boolean trees of two models computed by our method.



Fig. 9. Boolean trees of two B-Rep solids. B-Rep patches are rendered in different colors. The zero isosurfaces of the Boolean functions at tree nodes are also illustrated (truncated by the bounding box).

Training data preparation. We randomly sample 50,000 points uniformly from each B-Rep model for training, and the corresponding surface normals are assigned to the sampled points. The set of these sample points is denoted as *PS*. On average, we sample [50000/L] points on each patch, where *L* is the number of patches in the decomposed patch set. We also ensure that each patch contains at least 50 sampled points, to avoid extremely insufficient sampling. The sampled point cloud is normalized to fit inside a $[-0.9, 0.9]^3$ box.

Training details. We train the network to convert a single B-Rep solid to NH-Rep, on an Nvidia GTX 1080 Ti GPU with the PyTorch framework [Paszke et al. 2017]. We use Adam Optimizer for 15 000 iterations with an initial learning rate of 0.005 scheduled to drop by a factor of 2 every 2000 iterations. The correction loss term is enabled after 10,000 iterations. We randomly select 16,384 points from *PS* in each iteration to calculate E_p , E_n , E_o , and E_{cons} , where 16384/L points are sampled on each patch. To calculate E_{eik} and E_o , we randomly sample a set of points within $[-1, 1]^3$ in each iteration. These points fall into two categories: local samples and global samples. Local samples are points perturbed from the sampled points with a normal distribution (mean: 0, stdev: σ) along a random direction, where σ is the shortest distance from the given sample point to the densely sampled point cloud. In total, the local samples contain 16,384 points. Global samples contain 2048 points randomly sampled in $[-1, 1]^3$ according to a normal distribution (mean: 0, stdev: 1.8). Eeik uses all local and global sample points, and E_o uses global samples only. The average training time for a single model is about 10 minutes.

Isosurface extraction. For evaluating the approximation quality of NH-Rep to the input B-Rep boundary, we extracted the isosurface of NH-Rep as a polygon mesh with feature preservation by the advanced isosurfacing algorithms [Ju et al. 2002; Kobbelt et al. 2001] as follows. On each edge of the cube, we compute its intersection point with h = 0, and also record the functional gradient as the point normal. All intersection computations are performed in parallel in GPU. The intersection points with normals are fed to the feature-aware marching cube algorithm to extract a mesh surface. The default grid resolution is 256^3 and we increase the resolution to 512^3 automatically if the isosurfacing algorithm generates very long mesh edges, indicating that the model may contain extremely narrow geometry.

For models with very sharp features, we found that the dual contour algorithm [Ju et al. 2002] is more stable and accurate than the algorithm of [Kobbelt et al. 2001]. In our implementation, we used an octree structure to speed up isosurface extraction. Here, we note that recent learning-based isosurfacing works [Chen et al. 2022b; Chen and Zhang 2021] are different from our work. These methods take discrete signals as input without access to surface gradients, such as a distance field grid, and predict mesh vertex location and edge crossing in cube cells for robustly reconstructing mesh facets and sharp edges. They are designed for mesh reconstruction and do not have access to ground-truth isosurfaces during the test phase. In our work, any isosurface of NH-Rep can be accurately evaluated, and thus the isosurface extraction step is just a conversion from implicit surfaces to mesh format.

Evaluation metrics. We measure the quality of implicit conversion with respect to the following metrics.

- Chamfer distance (CD) and two-side Hausdorff Distance (HD) between the extracted zero surface and the corresponding B-Rep surface. 50,000 points are randomly sampled on both surfaces for computation.
- Average normal error (NAE): Chamfer distance with respect to surface normals, using the same sample points as used for CD.
- Feature Chamfer distance (FCD) and feature angle error (FAE). We first detect sharp feature edges on M with the dirhedral angle threshold: $\delta_s = 30^\circ$, then uniformly sample points on the detected feature edges where the point distance interval is 0.004. We also record the dihedral angle for each edge sample point. A similar sampling is also done on ∂S . The Chamfer distance between these sample points defines FCD, and the sum of the absolution error of the dihedral angles between any nearest paired points in the FCD computation defines FAE.
- Occupancy IoU (**IoU**) with respect to the *ground-truth* mesh model provided in the ABC data set, calculated on a set of random sample points in $[-1, 1]^3$.
- The sum of the relative error of the predicted SDF with the real SDF (**DE**), calculated on a set of random sample points in $[-1, 1]^3$. The real SDF is calculated on the *ground-truth* mesh model.

Here, **CD** and **HD** measure the overall quality of the zero surfaces; **FCD** and **FAE** measure the sharp feature quality of the zero surfaces;





Fig. 10. Visual comparison of different approaches for implicit conversion. We render the input B-Rep models as point clouds that are the inputs to the compared methods, different B-Rep patches in random colors. The detected feature edges are illustrated in black. Only our method can reproduce correct sharp features after implicit conversion. The zoom-in views highlight the artifacts of the results from other methods.

input

DE and **IoU** measure the implicit field quality. The exact metric formulas are provided in Appendix C.

7.2 Experiment analysis

Benchmark and comparison. We applied our method to convert 10,935 B-Rep models to implicit representations. For comparison, we choose two classic feature-aware reconstruction methods: Screened Poisson Reconstruction (SPR) [Kazhdan and Hoppe 2013] and Robust Implicit Moving Least Squares (RIMLS) [Oztireli et al. 2009], and three representative learning-based implicit approaches: IGR [Gropp et al. 2020], SIREN [Sitzmann et al. 2020], Neural Spline (NS) [Williams et al. 2021]. The input to these competitive methods is the same sample points from the B-Rep model as we described in Section 7.1. Note that these methods cannot leverage patch adjacency information as ours. The SPR and RIMLS implementations are from MeshLab [Cignoni et al. 2008] with default settings. For SPR, its interpolation weight is set to 50 for better feature preservation. For IGR, SIREN, and NS, we use their default network and the suggested optimal parameter settings. The average training time for a single model is 40 minutes for IGR, 20 minutes for SIREN, and 1 minute for NS. Our network parameter size is 137k on average, which is smaller than SIREN (198k), IGR (1837k) and NS (250k). For IGR and SIREN, we apply our surface extraction algorithm to recover sharp features, as their implicit functions are easy to access. For SPR, RIMLS, and NS, we use the marching cube algorithm supplied by their original implementation. For SIREN, we also remove unwanted small components and pick the largest component for evaluation. The failure cases of RIMLS (126 models) and SPR (3 models) are not counted in the performance report.

The performance of all methods is reported in Table 1. SPR, IGR, and our approach have comparable quality in terms of CD, HD and



two rendered views of h(x) = 0

Fig. 11. NH-Rep conversion of three complicated B-Rep models. The patch numbers (from top to bottom) are: 121, 178, 191. The zero isosurfaces of h(x) are rendered from two different views for better visualization.

NAE, while RIMLS, SIREN, and NS perform worse. Our approach achieves the best performance on the sharp-feature-related metrics (FCD, FAE) and IoU. The DE of ours is worse than IGR, as NH-Rep does not recover the exact signed distance function due to the use of Boolean operations. The visual comparison shown in Fig. 10 further confirms the superiority of our method in preserving sharp features and recovering surface geometry. The robustness of our conversion method to B-Rep inputs with many surface patches is also illustrated in Fig. 10-bottom and Fig. 11 where the inputs have more than 100 patches. In our supplemental material, we provide metric histograms and more conversion results for further analysis.

Table 1. Quantitative evaluation of implicit conversion of different methods on 10,935 B-Rep models. CD, HD and FCD are scaled by 10^3 .

Method	$ $ CD \downarrow	$\mathbf{HD}\downarrow$	$\mathbf{NAE} \downarrow$	$\mathbf{FCD} \downarrow$	$\mathbf{FAE} \downarrow$	$\mathbf{DE}\downarrow$	IoU ↑
SPR	5.07	21.5	4.14°	14.5	43.3°	n/a	0.979
RIMLS	6.17	32.6	4.07°	20.7	31.1°	n/a	0.325
SIREN	14.3	87.3	4.57°	25.6	28.4°	0.657	0.889
IGR	7.10	41.0	2.92°	13.5	9.61°	0.039	0.965
NS	16.7	95.7	6.97°	30.3	48.4°	n/a	0.762
ours	5.31	24.3	2.42°	7.07	3.69°	0.064	0.979

Table 2. Quantitative evaluation of implicit conversion of different methods on Fig. 10-bottom model under equal-time setting.

Method	$\mathbf{C}\mathbf{D}\downarrow$	$\mathbf{HD}\downarrow$	$\mathbf{NAE}\downarrow$	$\mathbf{FCD}\downarrow$	$\mathbf{FAE}\downarrow$	$\mathbf{DE}\downarrow$	IoU ↑
SPR	7.86	28.2	6.00°	2.28	42.9°	n/a	0.997
NS	8.89	154	7.45°	5.50	45.3°	n/a	0.326
ours	7.93	30.6	6.30°	2.07	7.59°	0.0668	0.987



Fig. 12. Visual comparison of different approaches for implicit conversion under equal-time setting. Our method reproduces features more accurately than other methods.

Equal-time comparison with SPR and NS. As both NS and SPR with their default settings have less running time than our approach, we feed more sample points (5M points) and enable longer iterations (SPR: 800 iterations, NS: 200 iterations) and higher resolution for mesh extraction (NS: depth = 9, SPR: depth = 10), to maximize their conversion capacity. We used the complex model shown at the bottom of Fig. 10 as a test example. NS and SPR took 40 and 10 minutes to compute, respectively. The results of the comparison are reported in Table 2 and Fig. 12. NS produces bumpy geometry and spurious zero surface, and SPR achieves comparable accuracy (CD&HD) to our result trained with 50K points, but still has larger feature errors and contains tiny fluctuation in the flat region; furthermore, its implicit field storage is huge, exceeding 1.8 GB memory.

Training with sharp feature samples. Our default point sampling strategy does not include many sharp feature points for training. We design an additional test to check whether IGR and SIREN training with more sharp feature points can outperform our method. We selected 100 B-Rep models randomly from our benchmark dataset

ACM Trans. Graph., Vol. 41, No. 6, Article 276. Publication date: December 2022.

Table 3. Qualitative evaluation of implicit conversion of different methods using additional sharp feature points on 100 B-Rep models. **Fea.** indicates whether these additional points are used.

				•				
Method	Fea.	$\mathbf{C}\mathbf{D}\downarrow$	$\mathbf{HD}\downarrow$	$\mathbf{NAE}\downarrow$	$\mathbf{FCD}\downarrow$	$\mathbf{FAE} \downarrow$	$\mathbf{DE}\downarrow$	$IoU \uparrow$
IGR	\checkmark	7.21	42.6	3.11°	5.16	9.94°	0.0444	0.981
SIREN	\checkmark	7.07	29.5	4.17°	8.49	29.4°	0.4570	0.968
ours	×	5.49	22.7	2.67°	3.80	3.75°	0.0583	0.989
ours	\checkmark	5.42	21.7	2.57°	3.45	3.43°	0.0526	0.992



BSP-Net CSG-Stump CAPRI-Net Ours Ground truth Fig. 13. Comparison with BSP-Net, CSG-Stump and CAPRI-Net on implicit conversion. Inputs are selected from ShapeNet. We illustrate the boundary surfaces extracted from different methods. BSP-Net, CSG-Stump and CAPRI-Net fail to approximate CAD models accurately due to the limited representation power and inaccurate prediction.

and added 10k points sampled at feature edges for each model, with the original sampled points. As seen in the qualitative evaluation (Table 3), IGR and SIREN with these additional inputs cannot yet compete with our method which does not utilize sharp feature points, while these additional inputs slightly improve our method.

Comparison with CSG-based methods. We also compared our method with BSP-Net [Chen et al. 2020], CSG-Stump [Ren et al. 2021], and CAPRI-Net [Yu et al. 2022], which predict the CSG tree and a set of primitives to approximate the input solid. As these methods were trained on ShapeNet [Chang et al. 2015], we selected five typical models from their test sets and used their trained networks and prepared input data for comparison. The inputs are voxel cells (resolution 64³) for BSP-Net, 2048 points for CSG-Stump, and 8192 points with normal vectors for CAPRI-Net. For our method, we detect sharp features of the input mesh models and segment each mesh into a set of patches to construct a B-Rep for implicit conversion. We extracted the zero isosurfaces of these methods for visual comparison. As seen in Fig. 13, BSP-Net, CSG-Stump and CAPRI-Net approximate the inputs roughly with compact and simple elements but cannot accurately represent the input solids; while our method achieves more accurate implicit conversion results.

Implicit Conversion of Manifold B-Rep Solids by Neural Halfspace Representation • 276:11



Table 4. Ablation study of loss terms on 100 models.

Config.	$\mathbf{C}\mathbf{D}\downarrow$	$\mathbf{H}\mathbf{D}\downarrow$	$\mathbf{NAE} \downarrow$	$\mathbf{FCD} \downarrow$	$\mathbf{FAE}\downarrow$	$\mathbf{DE}\downarrow$	IoU ↑
w/o E_n	26.0	207	12.7°	39.7	24.3°	0.3440	0.625
w/o E_o	6.07	41.3	2.75°	4.88	4.10°	0.0612	0.986
w/o Ec&Econs	5.50	24.7	2.69°	3.82	3.85°	0.0868	0.989
default	5.49	22.7	2.67°	3.80	3.75°	0.0583	0.989

7.3 Ablation study

We validate our loss design and network design through a set of ablation studies, performed on 100 B-Rep models randomly selected from the benchmark dataset.

Efficacy of loss terms. We validate the efficacy of each loss term by dropping one of them during training. The performance of these ablations is reported in Table 4. We have the following observations.

- The missing of E_n produces the highest errors in all metrics, since the normal orientation of h on the surface points could be very different from the ground truth. Fig. 14-(a) illustrates typical artifacts such as distorted and non-smooth geometry, even additional parts, and missing sharp features.
- Without *E*₀, the network has less ability to constrain *h* = 0 on the input patches; thus, it may produce additional components as shown in Fig. 14-(b), and lead to a higher HD error.
- Without *E_{cons}* and *E_c*, *f_i* may be inactivated when evaluating *h* on patch s_i, resulting in a higher HD error than our default setting, as shown in Fig. 14-(c).
- Our default setting achieves the lowest error in all metrics, being more faithful to the input.

Impact of network size. We tested our network performance with different numbers of MLP layers and neurons. The network configuration is indicated as $N \times M$, N is the number of neurons in each MLP layer, and M is the number of MLP layers. Table 5 reports the average performance. With a fixed M (M = 3), a larger N helps to improve all metrics; when M = 7, the improvement brought by a larger N is less significant and worse on some metrics such as HD and FAE possibly due to overfitting. Our default setting 256×3 balances network performance and network size. It obtains the best performance on feature preservation and is slightly worse than the configuration of 256×7 in other metrics.

Table 5. Network performance with different network configurations on 100 models. 256×3 is our default setting.

Config.	$\mathbf{C}\mathbf{D}\downarrow$	$\mathbf{H}\mathbf{D}\downarrow$	$\mathbf{NAE}\downarrow$	$\textbf{FCD} \downarrow$	$\mathbf{FAE}\downarrow$	$\mathbf{DE}\downarrow$	IoU ↑
64×3	5.79	34.7	3.06°	4.33	4.28°	0.0839	0.981
128×3	5.66	27.9	2.81°	3.98	3.89°	0.0669	0.981
256×3	5.49	22.7	2.67°	3.80	3.75°	0.0583	0.989
256×7	5.44	21.3	2.63°	3.81	5.02°	0.0569	0.990
512×7	5.46	23.4	2.70°	3.61	6.18°	0.0477	0.988



Fig. 15. Ablation study of shared MLPs and separated MLPs on two models.

Table 6. Qualitative evaluation of implicit conversion of models in Fig. 15.

Model	Config.	CD↓	$HD\downarrow$	NAE \downarrow	FCD ↓	$FAE \downarrow$	$\mathbf{DE}\downarrow$	IoU↑
А	sep.	4.26	17.4	2.98°	1.31	0.522°	0.0742	0.989
А	shared	4.23	16.3	3.06°	1.34	0.625°	0.0712	0.990
В	sep.	7.90	28.3	1.61°	6.97	7.67°	0.0590	0.996
В	shared	7.86	29.6	1.58°	7.22	7.45°	0.0303	0.997

Separated MLPs for individual patches. In our network architecture, the MLP layers are shared to predict individual f_i s. We conducted an ablation study on two models (see Fig. 15) to verify whether the use of separated MLPs for individual patches could improve the accuracy of implicit conversion. We find that the implicit conversion qualities of both versions are comparable (see Table 6). However, the network size of using separated MLPs is much larger, as it is proportional to the number of patches. Our shared MLP strategy provides a great balance between model size and implicit conversion quality.

276:12 • Guo, et al.



Fig. 17. Conversion from segmented point clouds to NH-Reps. Some sharp features (bottom) are missed due to incorrect segmentation by [Sharma et al. 2020]: the green points occupy two different surface regions. Each model has 10000 sample points.

7.4 Robustness test

Data noise. To test the robustness of our algorithm to noise, we add random noise in the range of [-0.018, 0.018] to point positions along normal directions and perturb normal directions within the angle interval $[-3^{\circ}, 3^{\circ}]$. Here, for our method, we assume that the patch information and feature convexity are correct, and we also drop the correction loss in our training because this loss function is sensitive to noise. Fig. 16 shows the reconstruction results generated by our method, as well as SIREN and IGR, on two noisy inputs. The results show that our approach and IGR are less affected by noise, whereas our approach achieves the best result possibly because we have fewer MLP layers than IGR that avoid the overfitting problem, and our CSG operations are good for recovering sharp features.

Segmented point clouds. In practice, CAD models are obtained by scanning, in point cloud format, without B-Rep information. It is possible to segment the point cloud first, then use our method to convert it to NH-Rep. In Fig. 17, we utilize ParseNet [Sharma et al. 2020] to obtain the segmentation and recognize the type of feature curves (convexity or concavity) between two adjacent patches according to the majority of normal variations at the boundary points. The input models are selected from the ParseNet test set. The zero isosurfaces in the figure show that our method recovers the CAD mesh in good quality if the segmentation is reliable. For the imperfect segmentation shown in the bottom row, our approach still approximates the input but fails to model some sharp features due to the wrong segmentation (dark green region), which consists of two disjoint parts that cross the feature region.

Sensitivity to patch decomposition. In our patch decomposition algorithm, random patch selection can introduce different decomposed patch sets, but has a minor impact on the quality of converted NH-Reps. We select a complicated model (Fig. 18-a) that requires



Fig. 18. NH-Rep generated from different patch decompositions. (a): Input patches of the input model. (b): Two different decomposed patches due to random initialization. (c): Zero surfaces of the generated NH-Reps; (d): Ground truth. Patches are rendered in different colors.

Table 7. Qualitative evaluation of implicit conversion of the model in Fig. 18.

Model	$\mathbf{C}\mathbf{D}\downarrow$	$\mathbf{HD}\downarrow$	$\mathbf{NAE}\downarrow$	$\mathbf{FCD}\downarrow$	$\mathbf{FAE}\downarrow$	$\mathbf{DE}\downarrow$	IoU ↑
Тор	5.87	22.7	5.20°	3.11	11.7°	0.0531	0.991
Bottom	5.85	24.9	5.19°	2.88	11.4°	0.0619	0.993



Fig. 19. Patch merging for NH-Rep. From left to right: Patches of the input model before and after merging, the zero surface of NH-Rep after patch merging, and the ground truth.

patch composition to evaluate the impact of different decompositions. Two different decomposed patch sets are generated (see Fig. 18-b top and bottom). Both the zero isosurfaces of the resulting NH-Reps (Fig. 18-c) are close to ground truth, and the quality of two different NH-Reps is comparable (see Table 7).

Scalability. For CAD models with a large number of patches in the ABC dataset, we find that many patches are very small and narrow, and many of the shared feature curves are not sharp. If we naively apply our implicit conversion algorithm, dense sample points on and around small patches are needed to keep the approximation error low. A practical way is to merge small patches with its neighbors to reduce the patch number if their shared edges are *smooth*. We tested this strategy on a model with 868 patches. Fig. 19 shows that the total number of patches can be reduced to 11, and our conversion algorithm can perform well.

8 APPLICATIONS

NH-Rep representation can be incorporated with many applications, such as inside / outside query, surface offset, Boolean operations, feature edge blending, and mesh repair.

Fast inside/outside query. Due to the implicit representation of NH-Rep and parallel computation on GPU, it is easy to perform inside/outside queries efficiently. We selected five models from our benchmark dataset, and query the functional values on 2^{17} points



Fig. 20. Level sets of SIREN (top), IGR (middle), and our method (bottom) with different isovalues: -0.1, 0, 0.1, 0.2, (from left to right).



Fig. 21. Zero surfaces of applying boolean operations on two shapes *A* and *B* represented by NH-Rep.

randomly sampled inside $[-1, 1]^3$. The average query time on a single model is 33 ms, which is faster than IGR (93 ms), and slightly slower than SIREN (30 ms).

Offset surfaces. The implicit nature of NH-Rep enables us to perform sharp offsets via changing the level sets of h(x). Fig. 20 illustrates the offset surfaces at iso values: -0.1, 0, 0.1, 0.2. NH-Rep maintains sharp features in all offset results. In contrast, IGR cannot model sharp offsets and its offset surfaces are also bumpy with larger iso values; SIREN does not produce plausible offset results.

Boolean operations. NH-Rep supports fast and robust Boolean operations, as no explicit surface intersection is needed. The intersection, union, and complement operated on shapes correspond to max, min, – operated on implicit functions. Fig. 21 shows the results of applying Boolean operations on two complicated surfaces represented by NH-Rep.

Feature blending. NH-Rep can be combined with R-functions [Shapiro 2007] to blend feature edges of solids so that two adjacent patches with sharp features can be joined smoothly. We achieve this goal by replacing the operation max, min with ρ -blending function,



Fig. 22. Feature edge blending of NH-Rep using R-functions. ρ is set to 0.05.



Fig. 23. Mesh repairing. The input meshes are not watertight and contains many small seams (mismatched edges), highlighted with different colors. By sampling dense points on the input mesh (middle) and convert them to NH-Rep by our algorithm, a watertight mesh with feature preserving can be extracted from the zero isosurface of NH-Rep. The zoom-in views highlight the gap before repairing and the seamless results after repairing.

which is defined as follows:

$$B(f,g) = f + g + s\sqrt{f^2 + g^2 + \frac{s_\rho(s_\rho - |s_\rho|)}{8\rho^2}},$$
 (13)

where f, g are the input functions for max or min, ρ controls the blend radius and $s_{\rho} = f^2 + g^2 - \rho^2$. s = 1 for operation max and s = -1 for min. Here, since the ρ -blending function is bivariable, we rewrite the multivariate Boolean operation of NH-Rep as bivariable operations first, such as max{ f_1, f_2, f_3 } = max{max{ f_1, f_2, f_3 }, then replace the operator by the ρ -blending function. Fig. 22 shows the two examples of feature blending.

Mesh repairing. Solid models in B-Rep format are often converted to triangle meshes for different purposes such as file exchange between different 3D software. The conversion may discretize B-Rep patches individually, resulting in mismatching patch edges between adjacent patches, as shown in Fig. 23-left. Thus, the converted mesh is not watertight and exhibits many visible seams which are not good for downstream tasks such as 3D printing. These problems can be repaired using NH-Rep. We first segment the mesh based on the existing seams and sharp features and build the adjacent graph on the segmented patches. Here, we regard two disconnected patches with small seams as connected when constructing the Boolean tree. We then sample dense points on the mesh and fit the NH-Rep to them. The formulation of NH-Rep guarantees that a closed mesh with sharp feature preservation can be extracted. Fig. 23 shows the repaired results of two models. Here, we note that our approach is not designed to fix other kinds of mesh imperfections, such as big holes, non-manifold connectivity, and self-intersection.

9 CONCLUSION

We present neural halfspace representations to convert B-Rep solid models to implicit solids. The efficacy and robustness of our approach are validated through extensive experiments on a large CAD dataset. Compared to other implicit conversion/reconstruction algorithms, our approach offers superior approximation quality and preserves sharp features more faithfully. As demonstrated in Section 8, NH-Rep is useful for various applications and it is promising to integrate it with advanced operations applied to function representations [Pasko et al. 1995; Shapiro 2007], to obtain more shape modeling and optimization capabilities.

Some limitations remain in our work and deserve future development. First, unlike the traditional CSG representation and the boundary-sampled halfspace representation [Du et al. 2021] that are easy to edit by manipulating their primitives via moving, scaling, rotating, and other simple geometry operations, it is not intuitive to manipulate the halfspace functions of NH-Rep for shape editing, as its halfspace functions usually do not correspond to simple solids. Second, the time-consuming learning step hinders interactive shape modeling and implicit conversion. The recent exploration of multilevel and adaptive feature volumes for computing neural implicits [Müller et al. 2022; Takikawa et al. 2021] is promising for accelerating our algorithm. Third, conversion from unsegmented 3D scans to NH-Rep requires reliable segmentation, which is challenging when the input contains large noise, outliers, or insufficient sampling. Lastly, NH-Rep does not support non-manifold B-Rep models. Integrating other representations such as multiphase implicit functions [Yuan et al. 2012] into our formulation could be an interesting research direction.

REFERENCES

- George Allen. 2021. nTopology's Implicit Modeling Technology. https://ntopology.com/ resources/whitepaper-implicit-modeling-technology/. Last accessed 2021-12-1. Matan Atzmon and Yaron Lipman. 2020. SAL: Sign agnostic learning of shapes from raw data. In *CVPR*.
- Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Gaël Guennebaud, Joshua A. Levine, Andrei Sharf, and Claudio T. Silva. 2017. A survey of surface reconstruction from point clouds. Comput. Graph. Forum 36, 1 (2017), 301–329.
- Suzanne F Buchele and Richard H Crawford. 2003. Three-dimensional halfspace constructive solid geometry tree construction from implicit boundary representations.
- ACM Trans. Graph., Vol. 41, No. 6, Article 276. Publication date: December 2022.

In Proceedings of the eighth ACM symposium on Solid modeling and applications. 135–144.

- J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. 2001. Reconstruction and representation of 3D objects with radial basis functions. In SIGGRAPH. 67–76.
- Rohan Chabra, Jan Eric Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. 2020. Deep local shapes: Learning local SDF priors for detailed 3D reconstruction. In ECCV.
- Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. ShapeNet: An information-rich 3D model repository. arXiv:1512.03012 [cs.GR].
- Falai Chen, Tor Dokken, and Géraldine Morin. 2022a. Geometric Modeling: Interoperability and New Challenges (Dagstuhl Seminar 21471). Dagstuhl Reports 11, 10 (2022), 111–150.
- Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. 2022b. Neural dual contouring. ACM Trans. Graph. (2022).
- Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. 2020. BSP-Net: Generating compact meshes via binary space partitioning. In CVPR.
- Zhiqin Chen and Hao Zhang. 2019. Learning implicit fields for generative shape modeling. In CVPR.
- Zhiqin Chen and Hao Zhang. 2021. Neural marching cubes. ACM Trans. Graph 40, 6 (2021), 1–15.
- Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. 2020. Implicit functions in feature space for 3D shape reconstruction and completion. In *CVPR*.
- Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. 2008. MeshLab: an Open-Source Mesh Processing Tool. In *Eurographics Italian Chapter Conference*, Vittorio Scarano, Rosario De Chiara, and Ugo Erra (Eds.). The Eurographics Association.
- Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. 2020. CvxNets: Learnable convex decomposition. In CVPR.
- Tao Du, Jeevana Priya Inala, Yewen Pu, Andrew Spielberg, Adriana Schulz, Daniela Rus, Armando Solar-Lezama, and Wojciech Matusik. 2018. InverseCSG: Automatic conversion of 3D models to CSG trees. ACM Trans. Graph (2018).
- Xingyi Du, Qingnan Zhou, Nathan Carr, and Tao Ju. 2021. Boundary-sampled halfspaces: A new representation for constructive solid modeling. ACM Trans. Graph. 40, 4, Article 53 (2021), 15 pages.
- Markus Friedrich, Pierre-Alain Fayolle, Thomas Gabor, and Claudia Linnhoff-Popien. 2019. Optimizing evolutionary CSG tree extraction. In Proceedings of the Genetic and Evolutionary Computation Conference. 1183–1191.
- Sarah F. Frisken, Ronald N. Perry, Alyn P. Rockwood, and Thouis R. Jones. 2000. Adaptively sampled distance fields: A general representation of shape for computer graphics. In SIGGRAPH. 249–254.
- Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. 2020. Implicit geometric regularization for learning shapes. In ICML.
- Jian Huang, Yan Li, Roger Crawfis, Shao-Chiung Lu, and Shuh-Yuan Liou. 2001. A complete distance field representation. In *IEEE VIS*. IEEE, 247–561.
- Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph G Lambourne, Karl DD Willis, Thomas Davies, Hooman Shayani, and Nigel Morris. 2021. UV-Net: Learning from boundary representations. In CVPR. 11703–11712.
- Chiyu Max Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. 2020. Local implicit grid representations for 3D scenes. In CVPR.
- Mark W Jones, J Andreas Baerentzen, and Milos Sramek. 2006. 3D distance fields: A survey of techniques and applications. *IEEE Trans. Vis. Comput. Graphics* 12, 4 (2006), 581–599.
- Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. 2002. Dual contouring of Hermite data. ACM Trans. Graph. 21, 3 (2002), 339–346.
- Kacper Kania, Maciej Zięba, and Tomasz Kajdanowicz. 2020. UCSG-Net Unsupervised discovering of constructive solid geometry tree. In *NeurIPS*.
- Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson surface reconstruction. In Symp. Geom. Proc. 61–70.
- Michael Kazhdan and Hugues Hoppe. 2013. Screened Poisson surface reconstruction. ACM Trans. Graph. 32, 3 (2013), 29:1–29:13.
- Leif P. Kobbelt, Mario Botsch, Ulrich Schwanecke, and Hans-Peter Seidel. 2001. Feature sensitive surface extraction from volume data. In SIGGRAPH. 57–66.
- Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. 2019. ABC: A big CAD model dataset for geometric deep learning. In *CVPR*.
- Dan Koschier, Crispin Deul, and Jan Bender. 2016. Hierarchical hp-adaptive signed distance fields. In Symposium on Computer Animation. 189-198.
- Joseph G. Lambourne, Karl D.D. Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. 2021. BRepNet: A topological message passing system for solid models. In CVPR. 12773–12782.
- Jiabao Lei and Kui Jia. 2020. Analytic marching: An analytic meshing solution from deep implicit surface networks. In *ICML*. PMLR, 5789–5798.

- Shi-Lin Liu, Hao-Xiang Guo, Hao Pan, Peng-Shuai Wang, Xin Tong, and Yang Liu. 2021. Deep implicit moving least-squares functions for 3D reconstruction. In CVPR. 1788–1797.
- William E. Lorensen and Harvey E. Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. In SIGGRAPH. Association for Computing Machinery, 163–169.
- Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. ACM Trans. Graph. 41, 4, Article 102 (2022), 102:1–102:15 pages.
- Pavol Novotný and Milos Srámek. 2005. Representation of objects with sharp details in truncated distance fields. In *Volume Graphics 2005.*
- Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. 2003. Multi-level partition of unity implicits. ACM Trans. Graph. 22, 3 (2003), 463–470.
- Cengiz Oztireli, Gaël Guennebaud, and Markus Gross. 2009. Feature preserving point set surfaces based on non-linear kernel regression. Comput. Graph. Forum 28, 2 (2009), 493–501.
- Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: learning continuous signed distance functions for shape representation. In *CVPR*.
- Alexander Pasko, Valery Adzhiev, Alexei Sourin, and Vladimir Savchenko. 1995. Function representation in geometric modeling: concepts, implementation and applications. The visual computer 11, 8 (1995), 429–446.
- Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. (2017).
- Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. 2020. Convolutional occupancy networks. In ECCV.
- Huamin Qu, Nan Zhang, Ran Shao, Arie Kaufman, and Klaus Mueller. 2004. Feature preserving distance fields. In 2004 IEEE Symposium on Volume Visualization and Graphics. IEEE, 39-46.
- Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, Haiyong Jiang, Zhongang Cai, Junzhe Zhang, Liang Pan, Mingyuan Zhang, Haiyu Zhao, et al. 2021. CSG-Stump: A learning friendly CSG-Like representation for interpretable shape parsing. In *ICCV*. 12478–12487.
- Antonio Ricci. 1973. A constructive geometry for computer graphics. Comput. J. 16, 2 (1973), 157–160.
- Vadim Shapiro. 2002. Solid Modeling. Handbook of computer aided geometric design 20 (2002), 473–518.
- Vadim Shapiro. 2007. Semi-analytic geometry with R-functions. ACTA numerica 16 (2007), 239–303.
- Vadim Shapiro and Donald L Vossler. 1991a. Efficient CSG representations of twodimensional solids. J. Mech. Des. 113 (1991), 292–305. Issue 3.
- Vadim Shapiro and Donald L. Vossler. 1991b. Construction and optimization of CSG representations. Computer-Aided Design 23, 1 (1991), 4–20.
- Vadim Shapiro and Donald L. Vossler. 1993. Separation for boundary to CSG conversion. ACM Trans. Graph. 12, 1 (1993), 35–55.
- Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. 2018. CSGNet: Neural shape parser for constructive solid geometry. In *CVPR*.
- Gopal Sharma, Difan Liu, Subhransu Maji, Evangelos Kalogerakis, Siddhartha Chaudhuri, and Radomír Měch. 2020. ParseNet: A parametric surface fitting network for 3D point clouds. In ECCV. 261–276.
- Vincent Sitzmann, Julien NP Martel, Alexander W Bergman, David B Lindell, and Gordon Wetzstein. 2020. Implicit neural representations with periodic activation functions. In *NeurIPS*.
- Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2021. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. (2021).
- Francis Williams, Matthew Trager, Joan Bruna, and Denis Zorin. 2021. Neural splines: Fitting 3D surfaces with infinitely-wide neural networks. In CVPR. 9949–9958.
- Karl DD Willis, Pradeep Kumar Jayaraman, Hang Chu, Yunsheng Tian, Yifei Li, Daniele Grandi, Aditya Sanghi, Linh Tran, Joseph G Lambourne, Armando Solar-Lezama, et al. 2022. JoinABLe: Learning bottom-up assembly of parametric CAD joints. In CVPR.
- Q. Wu, K. Xu, and J. Wang. 2018. Constructing 3D CSG models from 3D raw point clouds. Comput. Graph. Forum 37, 5 (2018), 221–232.
- Rundi Wu, Chang Xiao, and Changxi Zheng. 2021. DeepCAD: A deep generative network for computer-aided design models. In ICCV. 6772–6782.
- Xianghao Xu, Wenzhe Peng, Chin-Yi Cheng, Karl DD Willis, and Daniel Ritchie. 2021. Inferring CAD modeling sequences using zone graphs. In CVPR. 6062–6070.
- Guandao Yang, Serge Belongie, Bharath Hariharan, and Vladlen Koltun. 2021. Geometry processing with neural fields. *NeurIPS* 34 (2021).
- Fenggen Yu, Zhiqin Chen, Manyi Li, Aditya Sanghi, Hooman Shayani, Ali Mahdavi-Amiri, and Hao Zhang. 2022. CAPRI-Net: Learning compact CAD shapes with adaptive primitive assembly. In CVPR.
- Zhan Yuan, Yizhou Yu, and Wenping Wang. 2012. Object-space multiphase implicit functions. ACM Trans. Graph. 31, 4 (2012), 1–10.



Fig. 24. Left: The input B-Rep. Middle: The reconstruction result by BSH [Du et al. 2021]. Right: The zero isosurface of our NH-Rep.

A A FAILURE CASE OF BOUNDARY-SAMPLED HALFSPACE

The limitation of BSH [Du et al. 2021] in handling patches tangentially contacted occurs often in CAD B-Rep models. Fig. 24 illustrates a simple failure case. The input B-Rep model consists of two cylindrical halfspaces and four planar halfspaces. Each cylindrical patch and its adjacent planar patch are contacted tangentially. The BSH algorithm cannot produce the correct result, while our approach works well.

B EXISTENCE OF IMPLICIT FUNCTIONS

In this section, we prove that for a decomposed patch set $\{s_1, \ldots, s_n\}$ of a B-Rep solid S, there exist a series of implicit functions $\{f_1, \ldots, f_n\}$ such that their composite function h satisfies Eq. (1).

It is known that an implicit function F can be induced from an oriented manifold surface Z (open or closed without self-intersection), with the condition that Z belongs to the zero isosurface of F. To prove the existence of $\{f_i\}_{i=1}^n$, it is equivalent to proving that there exists a set of orientable manifold surfaces: $\{Z_i\}_{i=1}^n$ which can induce $\{f_i\}_{i=1}^n$ to satisfy Eq. (1).

We prove the existence of $\{Z_i\}_{i=1}^n$ in a constructive way by creating Z_i from top to bottom along the Boolean tree, as follows. Without loss of generality, we show how to create relevant Z_i s on an arbitrary Boolean tree node. We assume that a tree node r has child patches Q_1, \ldots, Q_m , child nodes r_1, \ldots, r_d . Here $Q_i, i = 1, \ldots, m$ are a subset of $\{s_k\}_{k=1}^n$. We denote Q_{m+i} as the union of patches contained in r_i and its descendants. In the following construction process, we assume $op(r) = \max$. When $op(r) = \min$, the process is symmetric.

We process $Q_s, s = 1, ..., m + d$ in sequence as follows. If Q_s lies on the outer boundary of the shape (e.g., Q_1 in Fig. 25), we can extend Q_s from the boundary of Q_s to form an open orientable and non-self-intersecting surface Z_{Q_s} . The extension of Z_s should also avoid intersecting with other Q_i 's except at the boundary of Q_s . This can be achieved because Q_s lies on the boundary of a connected region and the boundary feature curves of Q_s are all convex. All constructed zero surfaces have no intersection with the boundary surface of S except on feature curves (or corners in 2D). Next, we need to remove all the other intersections of Z_i 's outside S, otherwise, the composited halfspace will have outlier regions. Unwanted intersections can be removed by exchanging and smoothing the intersected parts, as shown in Fig. 26.

If Q_s lies on the boundary of a cavity region of the solid (*e.g.*, Q_2 in Fig. 27), the zero surface can be constructed in a similar way, but the resulting surface would be a closed one. If the composited halfspace has an interior outlier region within the cavity, the zero surfaces should be expanded to cover the outlier region, as shown in Fig. 27(c).



Fig. 25. Construction of extended curves on a 2D model. The extended curves are trimmed by the bounding box for better illustration.

After constructing zero surfaces for all Q_i 's, the subpatches of Q_{m+i} , i = 1, ..., d should also be replaced with their extended versions. Suppose that \mathcal{R}_j 's are the subpatches of Q_{m+i} , then \mathcal{R}_j should be extended along $\partial \mathcal{R}_j \cap \partial \mathcal{Q}_{m+i}$, so that the union of all extended \mathcal{R}_j 's is $\mathcal{Z}_{\mathcal{Q}_{m+i}}$. In this way, the generated zero surfaces in the child nodes of r follow that of r.

The above procedure is executed from top to bottom; we can obtain a series of $\{Z_i\}_{i=1}^n$ at each leaf node. From $\{Z_i\}_{i=1}^n$, we can induce a set of implicit functions: $\{f_i\}_{i=1}^n$. Due to the construction process and the above properties, it is easy to verify that Eq. (1) can be fulfilled.

In Fig. 25 and Fig. 27, we illustrate the concept of surface extension in two 2D examples. Fig. 25-(a) presents an input shape with six decomposed patches s_1, \ldots, s_6 . Its Boolean tree is illustrated in Fig. 25-(b), f_i corresponds to s_i . Fig. 25-(c) is the grouped patches at the second layer of the Boolean tree, where Q_5 is the union of \mathbf{s}_1 and $s_2.$ The extended curves $\mathcal{Z}_{\mathcal{Q}_1},\ldots,\mathcal{Z}_{\mathcal{Q}_4}$ are illustrated in Fig. 25-(c), denoted by z_1, \ldots, z_4 . Q_5 is processed at the child node (Fig. 25-(f)) and the corresponding extended curves are z'_1 and z'_2 shown in Fig. 25-(g). By applying the union operation (corresponding to min operation), z_5 is constructed as shown in Fig. 25-(e). In Fig. 25-(h), all extended curves are plotted. Fig. 27(a) shows another example whose input is with genus 1. The extended curves of Q_1, Q_2, Q_3 and Q_4 are closed curves as depicted in Fig. 27-(c).

Here, note that a dedicated and complicated algorithm is needed to implement the above surface extension procedure, and we turn to train a neural network to compute the appropriate implicit functions directly.

C EVALUATION METRICS

We denote M_e as the extracted zero isosurface from any method, and M_q as the boundary surface of B-Rep. \mathcal{P}_e and \mathcal{P}_q are randomly sampled points on M_e and M_q , and \mathcal{F}_e and \mathcal{F}_q are randomly sampled points on the feature edges of M_e and M_q , respectively. For a point set \mathcal{H} and a query point $x \in \mathbb{R}^3$, we define the operation





Fig. 26. Removal of unwanted intersection by exchanging and smoothing when constructing 2D zero curves.



Fig. 27. Construction of extended curves on a 2D model with genus 1. (a) Input model with 4 patches. (b): Improper zero surfaces lead to interior outliers. (c): After expanding the zero surfaces, interior outliers can be removed.

 $Q(\mathbf{x}, \mathcal{H}) = \arg\min_{\mathbf{y}\in\mathcal{H}} \|\mathbf{x} - \mathbf{y}\|$ that returns the nearest point to x in \mathcal{H} , the operator n(x) that returns the surface normal at x, and the operator $\theta(x)$ that returns the dihedral angle at a feature point x. The evaluation metrics Section 7.1 are listed below.

$$\begin{split} \mathsf{CD}(M_e, M_g) &= \frac{1}{2|\mathcal{P}_e|} \sum_{\mathbf{p} \in \mathcal{P}_e} \|\mathbf{p} - Q(\mathbf{p}, \mathcal{P}_g)\| + \frac{1}{2|\mathcal{P}_g|} \sum_{\mathbf{p} \in \mathcal{P}_g} \|\mathbf{p} - Q(\mathbf{p}, \mathcal{P}_e)\|; \\ \mathsf{HD}(M_e, M_g) &= \max\{\max_{\mathbf{p} \in \mathcal{P}_e} \|\mathbf{p} - Q(\mathbf{p}, \mathcal{P}_g)\|, \max_{\mathbf{p} \in \mathcal{P}_g} \|\mathbf{p} - Q(\mathbf{p}, \mathcal{P}_e)\|\}; \\ \mathsf{NAE}(M_e, M_g) &= \frac{180^{\circ}}{2|\mathcal{P}_e|\pi} \sum_{\mathbf{p} \in \mathcal{P}_e} \arccos(\mathbf{n}(\mathbf{p}) \cdot \mathbf{n}(Q(\mathbf{p}, \mathcal{P}_g))) \\ &+ \frac{180^{\circ}}{2|\mathcal{P}_g|\pi} \sum_{\mathbf{p} \in \mathcal{P}_g} \arccos(\mathbf{n}(\mathbf{p}) \cdot \mathbf{n}(Q(\mathbf{p}, \mathcal{P}_e))); \\ \mathsf{FCD}(M_e, M_g) &= \frac{1}{2|\mathcal{F}_e|} \sum_{\mathbf{p} \in \mathcal{F}_e} \|\mathbf{p} - Q(\mathbf{p}, \mathcal{F}_g)\| + \frac{1}{2|\mathcal{F}_g|} \sum_{\mathbf{p} \in \mathcal{F}_g} \|\mathbf{p} - Q(\mathbf{p}, \mathcal{F}_e)\|; \end{split}$$

$$\begin{aligned} \mathsf{FAE}(M_e, M_g) = & \frac{1}{2|\mathcal{F}_e|} \sum_{\mathbf{p} \in \mathcal{F}_e} \|\theta(\mathbf{p}) - \theta(\mathcal{Q}(\mathbf{p}, \mathcal{F}_g))\| \\ &+ \frac{1}{2|\mathcal{F}_g|} \sum_{\mathbf{p} \in \mathcal{F}_g} \|\theta(\mathbf{p}) - \theta(\mathcal{Q}(\mathbf{p}, \mathcal{F}_e))\|. \end{aligned}$$

Ν

For a B-Rep solid, we compute its "ground-truth" signed distance field \mathcal{F}_q based on the accompanied triangle mesh in the ABC dataset, we measure the approximation error between the learned function \mathcal{F}_e to \mathcal{F}_q as follows:

$$\mathsf{DE}(\mathcal{F}_e, \mathcal{F}_g) = \frac{1}{|G|} \sum_{\mathbf{p} \in G} \frac{|\mathcal{F}_g(\mathbf{p}) - f(\mathbf{p})|}{|\mathcal{F}_g(\mathbf{p})| + \delta}.$$

Here G is a set of points randomly sampled in $[-1, 1]^3$, $|G| = 2^{17}$ and $\delta > 0$ is a tiny value to avoid zero division and is set to 10^{-9} .

IoU is the volumetric intersection of M_q and M_e divided by their volume union. 2^{17} points are randomly sampled in $[-1, 1]^3$ and their occupancy values are evaluated for computing IoU.