A Unified Differentiable Boolean Operator with Fuzzy Logic

HSUEH-TI DEREK LIU, Roblox, University of British Columbia, Canada
MANEESH AGRAWALA, Stanford University, Roblox, USA
CEM YUKSEL, University of Utah, Roblox, USA
TIM OMERNICK, Roblox, USA
VINITH MISRA, Roblox, USA
STEFANO CORAZZA, Roblox, USA
MORGAN MCGUIRE, Roblox, McGill University, Canada
VICTOR ZORDAN, Roblox, USA

Fig. 1. *We develop a unified boolean operator $\mathcal{B}_c$ that is differentiable with respect to the type of boolean operations. In the context of inverse CSG, starting with randomly initialize primitives and boolean operations (left tree), our method enables* continuous *optimization on both the primitives and the boolean operations in order to fit the target shape (middle tree). Performing inverse CSG fitting to the ground truth shape (grey) with our method leads to significant quality improvement (blue) over the traditional boolean operations with the min and max operators (red).*

This paper presents a unified differentiable boolean operator for implicit solid shape modeling using Constructive Solid Geometry (CSG). Traditional CSG relies on *min*, *max* operators to perform boolean operations on implicit shapes. But because these boolean operators are discontinuous and discrete in the choice of operations, this makes optimization over the CSG representation challenging. Drawing inspiration from *fuzzy logic*, we present a unified boolean operator that outputs a continuous function and is differentiable with respect to operator types. This enables optimization of both the primitives and the boolean operations employed in CSG with continuous optimization techniques, such as gradient descent. We further demonstrate that such a continuous boolean operator allows the modeling of both sharp mechanical objects and smooth organic shapes with the same framework. Our proposed boolean operator opens up new possibilities for future research toward fully continuous CSG optimization.

ACM Reference Format:
Hsueh-Ti Derek Liu, Maneesh Agrawala, Cem Yuksel, Tim Omernick, Vinith Misra, Stefano Corazza, Morgan McGuire, and Victor Zordan. 2024. A Unified Differentiable Boolean Operator with Fuzzy Logic. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27-August 1, 2024, Denver, CO, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3641519.3657484

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGGRAPH Conference Papers '24, July 27-August 1, 2024, Denver, CO, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0525-0/24/07…$15.00
https://doi.org/10.1145/3641519.3657484

1 INTRODUCTION

Boolean operations are a central ingredient in Constructive Solid Geometry (CSG) – a modeling paradigm that represents a complex shape using a collection of primitive shapes which are combined together via boolean operations (INTERSECTION, UNION, and DIFFERENCE). CSG provides a precise, hierarchical representation of solid shapes and is widely used in computer graphics.

The importance of CSG has motivated researchers to investigate the *inverse* problem; constructing a CSG tree for a given 3D model from a collection of parameterized primitive shapes. A common approach is to treat this as an optimization problem that involves choosing the structure of the CSG tree; the type of boolean operation to perform at each internal node in the tree, as well as the parameters and type (e.g., sphere, cube, cylinder) of the leaf node primitive shapes. The optimization is difficult because it contains a mixture of discrete (type of boolean operation, number and type of primitive shapes) and continuous (parameters of primitives e.g., radius, width, etc.) variables. Moreover, the degrees of freedom grow exponentially with the complexity of the CSG tree, making the optimization landscape very challenging to navigate.

Previous attempts either tackle the inverse optimization directly with *evolutionary algorithms* [Friedrich et al. 2019], or relax some of the discrete variables into continuous variables to reduce the discrete search space. For instance, one of the discrete decisions is

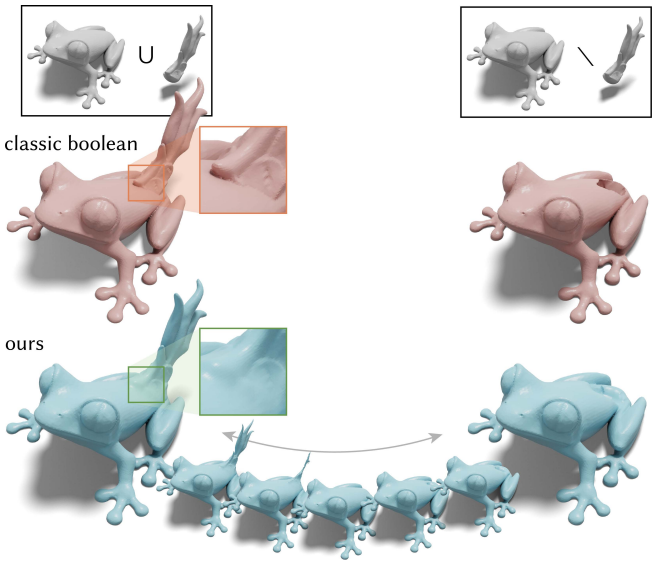


Fig. 2. *We present a differentiable boolean operator with respect to its operands and the operator. Given two implicit shapes, our boolean operator can control the blend region between two shapes and outputs a smoothly differentiable function (see the zoom-in part). One can also continuously switch the operator from one to another, such as from* UNION *(left) to* DIFFERENCE *(right).*

to determine which primitive types (e.g., sphere, cube, cylinder) to use, and a common relaxation is to optimize over a continuously parameterized family of primitives, such as *quadric surfaces* [Yu et al. 2023, 2022]. This approach allows continuous optimization (e.g., gradient descent) over choosing the type of each primitive, but not the entire tree; the choice of boolean operations and the number of primitives remain discrete variables. As a result, these inverse CSG methods pre-determine the structure of the tree including both the boolean operations and the number of primitives and focus on optimizing the primitive parameters.

In this work, we develop a unified differentiable boolean operator and show how this operator can be used to further relax inverse CSG optimization by turning the discrete choice of boolean operation for each internal CSG node into a continuous optimization variable. Drawing inspiration from *Fuzzy Logic* [Zadeh 1965], we first demonstrate how these individual fuzzy logic operations (t-norms, t-conorms) can be applied to boolean operations on solid shapes represented as soft occupancy functions. Fuzzy boolean operators guarantee that the result remains a soft occupancy function, unlike existing boolean operators (with *min*/*max*) that operate on signed distance functions. These fuzzy booleans on the soft occupancy naturally generalize CSG from modeling shapes with sharp edges to modeling smooth organic shapes. We then construct a unified fuzzy boolean operator that uses tetrahedral barycentric interpolation to combine the individual fuzzy boolean operations (see Fig. 2). We show that our unified operator is differentiable, avoids vanishing gradients and is monotonic making it especially well-suited for gradient-based optimization. We apply our unified boolean operator in the context of inverse CSG optimization and find significant improvements in the accuracy of the resulting tree compared to previous methods (see Fig. 1).

2 RELATED WORK

Our contribution uses fuzzy logic to design a unified differentiable boolean operator with applications in gradient-based inverse CSG optimization. While researchers have formulated inverse CSG as a program synthesis [Du et al. 2018; Sharma et al. 2018; Wu et al. 2021], combinatorial optimization [Wu et al. 2018], or a global optimization [Friedrich et al. 2019; Hamza and Saitou 2004] problem, we reformulate it as differentiable gradient descent optimization problem with respect to boolean and primitive parameters. Here, we consider how our work relates to differentiable CSG optimization and to other boolean operators used in geometric modeling.

2.1 Gradient-Based CSG Optimization

Optimizing CSG trees requires determining the structure of the tree, the boolean operations, and the primitive parameters. In order to deploy continuous optimization techniques (such as gradient descent), existing solutions rely on pre-defining all the discrete variables (the tree structure and the boolean operations), and then only optimizing the primitive parameters. The simplest pre-defined "CSG tree" is a union of many parts, including convex shapes [Chen et al. 2020; Deng et al. 2020] and neural implicit functions [Deng et al. 2022]. Some works have also explored more complicated pre-defined tree structures with a mixture of INTERSECTION, UNION, DIFFERENCE operators [Ren et al. 2021; Yu et al. 2023, 2022]. Instead of pre-determining the boolean operators, Kania et al. [2020] proposed a brute force approach to all possible boolean combinations. However this approach suffers from scalability issues as the number of combinations grows exponentially with respect to the depth of the CSG tree.

Our contribution complements these techniques by introducing a unified boolean operator which enables continuous optimization on the choice of boolean operations. This avoids the need for brute force or pre-determining boolean operations, leading to better reconstruction (Fig. 1).

2.2 Boolean Operators in Geometric Modeling

The importance of boolean operations has stimulated research topics in *differentiable* boolean operators in geometric modeling. Despite having the same name, the term "differentiable boolean" can refer to (1) boolean operators that output a differentiable function, and (2) unified boolean operators that can be differentiated with respect to the type of operations.

The most common usage of "differentiable boolean" refers to boolean operations that produce *differentiable* functions, also known as *soft blending*. Traditionally, *min* and *max* operators are used to produce INTERSECTION and UNION between two implicit functions. But the caveat is that their gradients are ill-defined at locations when the input functions have the same value. This motivated Ricci [1973] to introduce a soft blending operator using a variant of *p-norms* $\|x\|_p = (\sum_i |x|^p)^{1/p}$ to produce smooth and differentiable outputs.

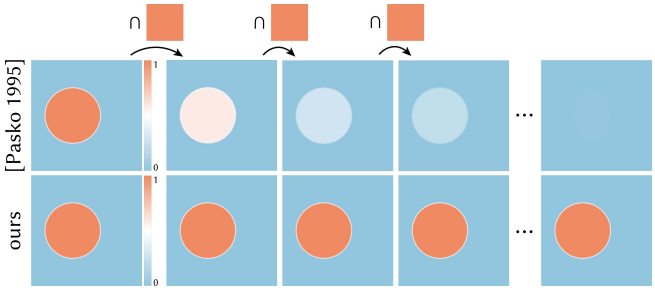


Fig. 3. *We show a comparison of our method against a soft boolean operator, R-function, used in [Pasko et al. 1995]. The R-function does not satisfy the axioms characterizing the behavior of* INTERSECTION*, it produces outputs that do not align with the classic (crisp)* INTERSECTION*. We show that the R-function (top row) produces a nearly empty shape when intersecting with the full shape multiple times. In contrast, our method matches the behavior of a crisp* INTERSECTION*.*

Later on, Pasko et al. [1995] demonstrated the use of the *Rvachev function* (R-function) [Rvachev 1963] to define a soft boolean operation that outputs C^n continuous implicit functions with user controllable n. Several works [Barthe et al. 2004; Gourmel et al. 2013; Wyvill et al. 1999] further extended this soft boolean formulation to provide fine controls on the blended region. Alternative formulations based on polynomial smoothing [Quilez 2013] or fuzzy logic [Li and Tian 2008] are viable choices as well. These approaches enable continuous outputs of individual boolean operations, but, in contrast to ours, they did not focus on differentiating through different boolean operations.

Another usage of "differentiable boolean" refers to unified boolean operators that can differentiably switch from one operation to another. Wyvill et al. [1999] show that a modified R-function is an instance of a unified boolean operator with a smooth transition between UNION and INTERSECTION. However, we demonstrate that the R-function does not satisfy important axioms of boolean operators (see Sec. 3.2), specifically the *boundary condition*. This implies that the R-function is prone to unexpected behavior. For instance, in Fig. 3 we show that if we intersect a shape with the full shape multiple times using the R-function, it ends up producing an almost-empty shape. Our method, instead, possesses the properties of a *valid* boolean operator (see Sec. 3.2) and guarantees to match the expected behavior of classic boolean operations when the inputs are binary.

Our proposed boolean operator is relevant to both usages of "differentiable boolean"; our method outputs continuous functions and can be differentiated through different boolean types. We demonstrate applications in modeling smooth shapes and inverse CSG optimization in Sec. 5.

3 BACKGROUND

The concept of fuzzy logic [Zadeh 1965] has applications in a wide variety of problem domains [Dzitac et al. 2017]. In computer graphics, fuzzy logic has been used in image processing [Chacón M 2006], color compositing [Smith 1995], and spline interpolation [Li and Tian 2008]. Here we summarize the core ideas of fuzzy logic.

3.1 Fuzzy Set

A fuzzy set $X = (P, f_X)$ is a tuple of the universe of elements $P = \{p\}$ and a *membership* function $f_X : P \rightarrow [0, 1]$ such that $f_X(p) = 0$ implies that element p is not a member of X, $f_X(p) = 1$ implies p is a full member of X and $0 < f_X(p) < 1$ implies p is a partial member of X. Fuzzy sets are a generalization of the classic "crisp" set, whose membership function only outputs 0 or 1. For instance, suppose we define a fuzzy set $H = (P, f_H)$ of the temperatures one considers hot. The universe of elements P are all possible temperature values. One might consider some temperature values, such as 40 degrees Celsius, as full members of H so that $f_H(40°C) = 1$. But 25 degrees Celsius, might only be a partial member $f_H(25°H) = 0.3$. Fuzzy sets model this notion of partial membership.

3.2 Fuzzy Logic

Fuzzy logic develops boolean operations on fuzzy sets. Given two fuzzy sets $X = (P, f_X)$ and $Y = (P, f_Y)$, a boolean operation is defined on the membership function. For instance, INTERSECTION $\cap$, UNION $\cup$, and COMPLEMENT $\neg$ between fuzzy sets are defined as

$$X \cap Y = (P, f_{X\cap Y}), \quad X \cup Y = (P, f_{X\cup Y}), \quad \neg X = (P, f_{\neg X}). \tag{1}$$

A core question in fuzzy logic research is how to define these boolean membership functions $f_{X\cap Y}, f_{X\cup Y}, f_{\neg X}$. A very common approach is to define them using the *min*, *max* operators [Gödel 1932] as

$$f_{X\cap Y} = min\big(f_X(p), f_Y(p)\big) = min(x, y), \tag{2}$$
$$f_{X\cup Y} = max\big(f_X(p), f_Y(p)\big) = max(x, y), \tag{3}$$
$$f_{\neg X} = 1 - f_X(p) = 1 - x. \tag{4}$$

To simplify notation, here and for the rest of the paper, we use the lowercase letter x to refer to $f_X(p)$, the membership function of X applied to a generic element $p \in P$. Similarly, y refers to $f_Y(p)$.

Fuzzy Intersection. Fuzzy logic researchers have explored other definitions of $f_{X\cap Y}$ [Klir and Yuan 1995]. Suppose $f_{X\cap Y} = \top(x, y)$. They define $\top$ as a valid intersection function when the following axioms hold:

$$\top(x, 1) = x \qquad \text{(boundary condition)}$$
$$\top(x, y) \leq \top(x, z), \text{if } y \leq z \qquad \text{(monotonicity)}$$
$$\top(x, y) = \top(y, x) \qquad \text{(commutativity)}$$
$$\top(x, \top(y, z)) = \top(\top(x, y), z) \qquad \text{(associativity)}$$

where $x = f_X(p), y = f_Y(p), z = f_Z(p) \in [0, 1]$ are fuzzy membership values for generic element p. Any function that satisfies these axioms is called a *t-norm* $\top$ [Menger 1942]. These axioms ensure that the behavior of the fuzzy intersection operator converges to the classic intersection (AND) operator when membership values are binary. Some popular t-norms include Gödel's [1932] minimum where $\top(x, y) = min(x, y)$, product $\top(x, y) = x \cdot y$, Łukasiewicz $\top(x, y) = max(0, x + y - 1)$, and Yager [1980] $\top(x, y) = max(1 - ((1 - x)^p + (1 - b)^p)^{1/p}, 0)$.

Fuzzy Union. Similarly, suppose $f_{X\cup Y} = \perp(x, y)$. In fuzzy logic, $\perp$ is a valid UNION function if:

$$\begin{aligned}
&\perp(x, 0) = x && \text{(boundary condition)}\\
&\perp(x, y) \le \perp(x, z), \text{if } y \le z && \text{(monotonicity)}\\
&\perp(x, y) = \perp(y, x) && \text{(commutativity)}\\
&\perp(x, \perp(y, z)) = \perp(\perp(x, y), z) && \text{(associativity)}
\end{aligned}$$

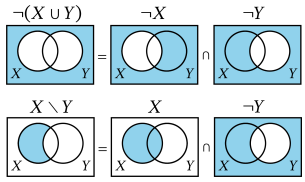
The functions that satisfy these axioms are known as *t-conorms* $\perp$. Common t-conorms include Gödel's [1932] maximum $\perp(x, y) = max(x, y)$, probabilistic sum $\perp(x, y) = x + y - x \cdot y$, bounded sum $\perp(x, y) = min(x + y, 1)$, and Yager [1980] $\perp(x, y) = min((x^p + y^p)^{1/p}, 1)$.

Fuzzy Complement. The set of axioms for defining a valid COMPLEMENT function $f_{\neg X} = C(x)$ are

$$\begin{aligned}
&C(0) = 1, C(1) = 0 && \text{(boundary condition)}\\
&\text{if } x \le y, \text{ then } C(x) < C(y) && \text{(monotonicity)}
\end{aligned}$$

Valid complement functions include cosine $C(x) = {}^{1+\cos(\pi x)}/_{2}$, Sugeno $C(x) = {}^{1-x}/_{1+\lambda x}$ with $\lambda \in (-1, \infty)$, and Yager [1980] $C(x) = (1 - x^\lambda)^{1/\lambda}$. The widely used complement $C(x) = 1 - x$, is the Yager complement with $\lambda = 1$.

Fuzzy Difference. In fuzzy logic, the DIFFERENCE operator $\setminus$ is usually derived from the De Morgan's laws (see inset), which state the relationship between the UNION, INTERSECTION, and COMPLEMENT operators,



$$\neg(X \cup Y) = \neg X \cap \neg Y, \tag{5}$$

and the relationship between DIFFERENCE and the other operators

$$X \setminus Y = X \cap \neg Y. \tag{6}$$

For the De Morgan's law to hold, one has to jointly define the INTERSECTION, UNION, and COMPLEMENT operators so that they satisfy Eq. 5. Then a valid DIFFERENCE operator $\setminus$ can be derived from INTERSECTION and COMPLEMENT using Eq. 6 as

$$f_{X\setminus Y} = \top(x, C(y)). \tag{7}$$

4 A UNIFIED DIFFERENTIABLE BOOLEAN OPERATOR

To apply fuzzy logic to CSG modeling, we interpret a solid shape, represented by a soft occupancy function, as a fuzzy set $X = \{P, f_X\}$. Here, $P = \{p\}$ denotes the universe of points $p \in \mathbb{R}^d$ and the membership function $f_X : P \to [0, 1]$ is the soft occupancy function representing the probability of a point lying inside the shape. Then we can directly apply the fuzzy boolean operations presented in Sec. 3. However, we must choose INTERSECTION, UNION, and COMPLEMENT appropriate to our task. We first present our choice for each of these functions (Sec. 4.1) and then describe how to combine them into a unified boolean operator (Sec. 4.2).

4.1 Product Fuzzy Logic

Motivated by our goal of continuous optimization, we would like each of our individual fuzzy boolean operations INTERSECTION $\top$, UNION $\perp$ and COMPLEMENT C to be differentiable and have non-vanishing (i.e. non-zero) gradients with respect to their inputs. Vanishing gradients can result in plateaus in the energy landscape making gradient-based optimization difficult.

Boolean operators as defined by the product fuzzy logic meet these criteria. Specifically they are defined as

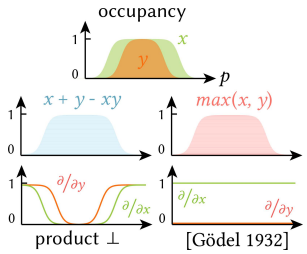
$$f_{X\cap Y} = \top(x, y) = xy \tag{8}$$

$$f_{X\cup Y} = \perp(x, y) = x + y - xy \tag{9}$$

$$f_{\neg X} = C(x) = 1 - x \tag{10}$$

where X and Y are two solid shapes and $x = f_X(p), y = f_Y(p) \in [0, 1]$ are their soft occupancy values at a generic point p. These definitions satisfy the axioms of valid boolean operators (see Sec. 3). They correspond to valid t-norm $\top$, t-conorm $\perp$, and complement C functions, respectively, in fuzzy logic. They also satisfy De Morgan's law Eq. 5, allowing us to compute DIFFERENCE as

$$f_{X\setminus Y} = x - xy, \quad f_{Y\setminus X} = y - xy \tag{11}$$



The product fuzzy logic boolean functions are differentiable with respect to their inputs x and y. Other fuzzy logic functions, such as Gödel's [1932] *min*/*max*, t-norm/t-conorm, are not differentiable at singularities. In addition, the product fuzzy logic functions are also much less prone to *vanishing gradients* compared to many other fuzzy logic function definitions [van Krieken et al. 2022]. More formally vanishing gradients occur when the partial derivatives $\partial/\partial x, \partial/\partial y$ equal zero (or become very small). In the inset, we illustrate a 1D example where occupancy values x are strictly larger than or equal to y. Defining UNION with the Gödel's *max* operator results in a zero gradient for y, as $\partial/\partial y = 0$. In contrast, using the UNION defined in Eq. 8 still possesses non-zero gradients for both x, y. In Fig. 4, we further demonstrate the importance of avoiding vanishing gradient in a simple example of our inverse CSG task with continuous optimization (see Sec. 5.2 for implementation details).

4.2 Unifying Boolean Operations

To create a unified fuzzy boolean operator that is differentiable with respect to the type of boolean operation (INTERSECTION, UNION, DIFFERENCE), our approach is to interpolate their respective membership functions using a set of interpolation control parameters $\mathbf{c}$. Our goal is to design an interpolation scheme that is continuous and monotonic in the parameters $\mathbf{c}$ so that the interpolation function avoids unnecessary local minima.

A naive solution is to use bilinear interpolation between the four boolean operations $f_{X\cap Y}, f_{X\cup Y}, f_{X\setminus Y}, f_{Y\setminus X}$. While such interpolation can look smooth, bilinear interpolation exhibits non-monotonic changes and creates local minima in the interpolated occupancy (see Fig. 5). This is because bilinear interpolation implicitly forces the

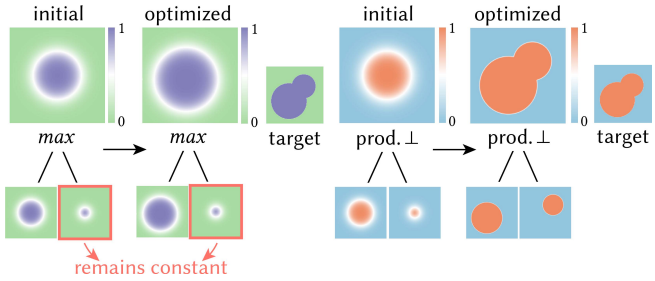


Fig. 4. *We demonstrate the importance of avoiding vanishing gradients in an inverse CSG for fitting the union of two circles. Using the traditional max[Gödel 1932] operator suffers from vanishing gradients, leading to a primitive (left, red) remaining unchanged throughout the optimization and thus failing to reconstruct the target shape. In contrast, our method presented in Eq. 8 avoids vanishing gradient and is able to recover the ground truth (right).*

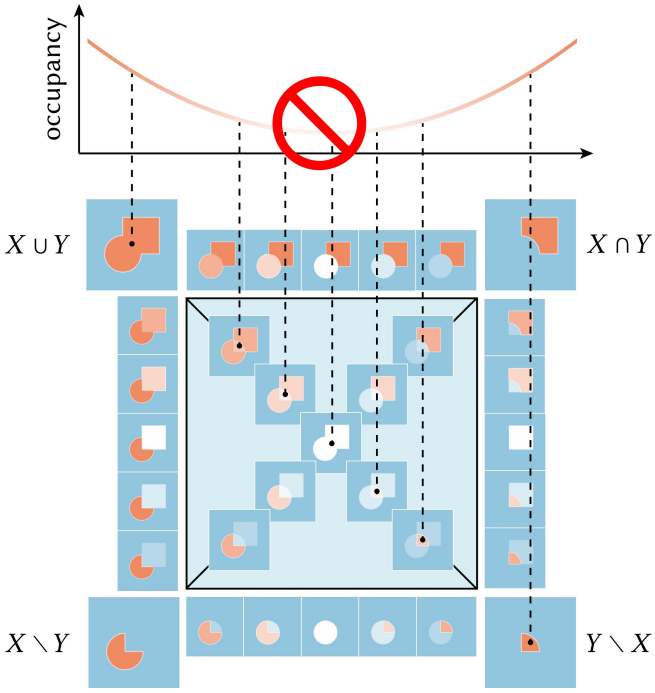


Fig. 5. *Naive bilinear interpolation introduces additional local minima in the occupancy value (top plot) when interpolating between, for instance,* UNION *(top left) and* DIFFERENCE *(bottom right).*

average between $f_{X\cup Y}$, $f_{Y\setminus X}$ and the average between $f_{X\cap Y}$, $f_{X\setminus Y}$ to be equivalent. In many cases, these averages are not equivalent and thus the constraint forces the interpolation to be non-monotonic.

Instead of this, we use tetrahedral barycentric interpolation. More specifically we treat individual boolean operations (UNION, INTERSECTION, and two DIFFERENCES) as vertices of a tetrahedron and

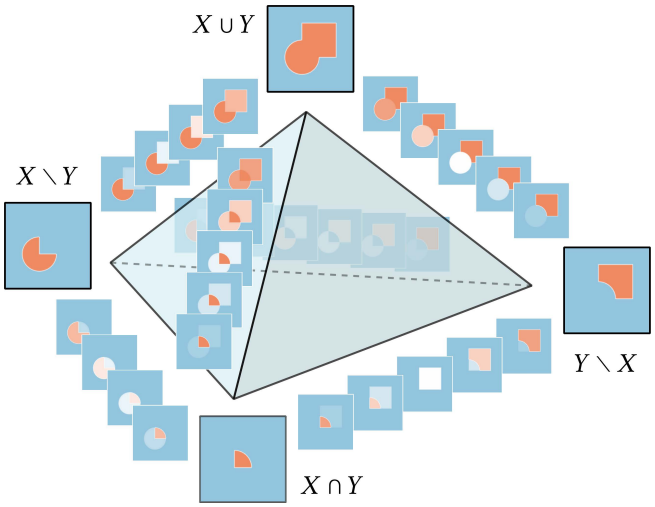


Fig. 6. *We use barycentric interpolation to produce monotonic interpolation between different boolean operators. This avoids undesired local minima in occupancy interpolation.*

define our unified boolean operator function $\mathcal{B}_{\mathbf{c}}$ as barycentric interpolation within it as

$$\boxed{\mathcal{B}_{\mathbf{c}}(x,y) = (c_1+c_2)x + (c_1+c_3)y + (c_0-c_1-c_2-c_3)xy} \tag{12}$$

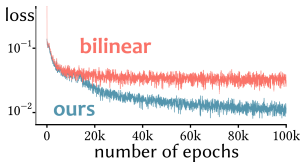
where $\mathbf{c} = \{c_0, c_1, c_2, c_3\}$ are parameters that control the type of boolean operations and they satisfy the properties of barycentric coordinates

$$0 \le c_i \le 1 \qquad \text{and} \qquad c_0 + c_1 + c_2 + c_3 = 1\,. \tag{13}$$

When the parameter $\mathbf{c}$ is a one-hot vector, i.e. the barycentric coordinates for the vertices of a tetrahedron, it exactly reproduces the product logic operators

$$\mathcal{B}_{1,0,0,0}(x,y) = xy = f_{X\cap Y} \tag{14}$$
$$\mathcal{B}_{0,1,0,0}(x,y) = x+y-xy = f_{X\cup Y} \tag{15}$$
$$\mathcal{B}_{0,0,1,0}(x,y) = x-xy = f_{X\setminus Y} \tag{16}$$
$$\mathcal{B}_{0,0,0,1}(x,y) = y-xy = f_{Y\setminus X} \tag{17}$$

From Eq. 12, we can immediately observe that our unified operator is continuously differentiable with respect to both the inputs $\partial\mathcal{B}_{\mathbf{c}}/\partial x$, $\partial\mathcal{B}_{\mathbf{c}}/\partial y$ and the control parameters $\partial\mathcal{B}_{\mathbf{c}}/\partial c_i$ by design. Moreover, our operator $\mathcal{B}_{\mathbf{c}}$ provides monotonic interpolation between the individual boolean operations at the vertices because interpolation along the edge of a tetrahedron is equivalent to a 1D convex combination (Fig. 6). Empirically, using barycentric interpolation leads to a smaller error compared to using bilinear interpolation (see inset).



5 RESULTS

Building on top of fuzzy logic, we first demonstrate our choice of individual operators from Eq. 8 leads to a natural generalization

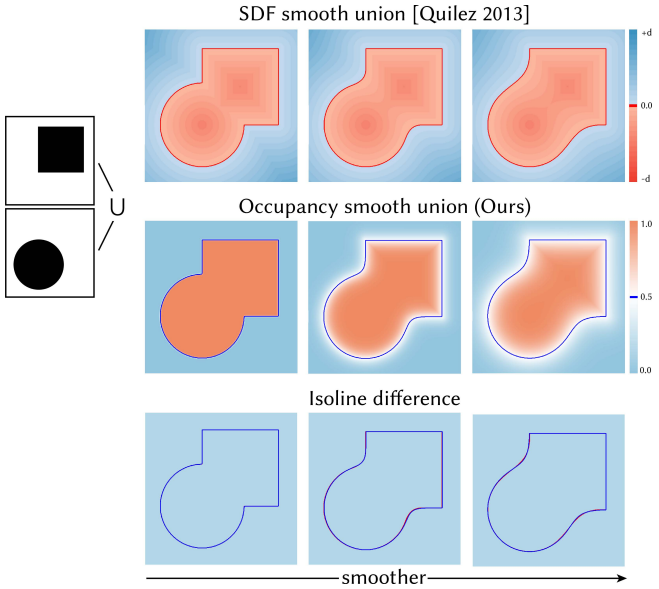


Fig. 7. *The soft boolean operator of Quilez [2013] has been demonstrated to be an effective way to model smooth shapes (top row). Our fuzzy boolean operator can also produce smooth boolean results (middle row) with visually indistinguishable isolines (bottom row).*

from modeling sharp solid objects to smooth organic shapes in Sec. 5.1. When combined with our unified boolean operator (see Eq. 12), they lead to significant improvements in the inverse CSG tasks, including fitting a single shape Sec. 5.2 or a collection of shapes Sec. 5.3.

5.1 Fuzzy CSG System

Using fuzzy boolean operators in CSG gives the ability to model both mechanical objects with crisp edges and smooth organic shapes with the same framework. Specifically, if the underlying implicit shapes are crisp binary occupancy functions, our method produces the same sharp results as the traditional CSG. If the input shapes are soft occupancy functions, our method outputs smooth shapes based on the "softness" present in the input shape.

This capability allows us to obtain visually indistinguishable results compared to the popular smoothed *min*/*max* [Quilez 2013] operations on the signed distance function (Fig. 7). Moreover, our approach is free from artifacts caused by discrepancy between the input and the output (see Fig. 8). This is because our method is *closed*: both the input and the output are guaranteed to be soft occupancy functions. This is different from the previous method by Quilez [2013] such that their outputs are not signed distance functions [Marschner et al. 2023], even though the input is.

As the smoothness is controlled at the primitive level, we can easily have adaptive smoothness across the shape by simply changing the softness of each primitive occupancy (see Fig. 9). Specifically, we consider primitive shapes represented as the signed distance function s, and we convert it to occupancy with the sigmoid function $sigmoid(t \cdot s)$ with different softnesses by adjusting the positive temperature parameter t.

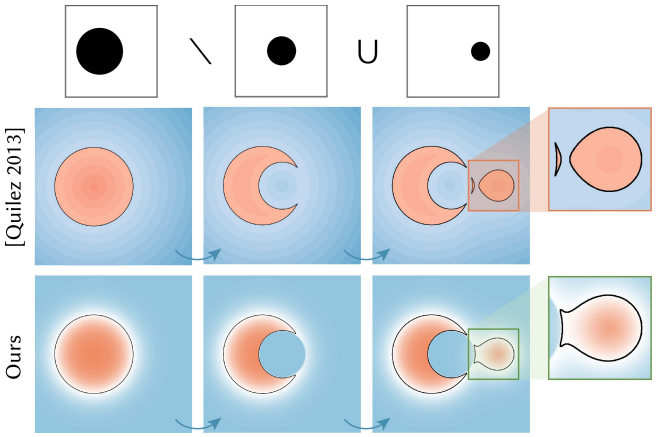


Fig. 8. *Using the soft union presented by Quilez [2013] to compute the boolean expression leads to "floating island" artifacts (top row). This is because boolean operations on the signed distance function do not output a* correct *signed distance function (see [Marschner et al. 2023]). Our boolean operator operates on the occupancy function and remains an occupancy function after boolean operations. This leads to soft blending results that are free from artifacts (bottom row).*

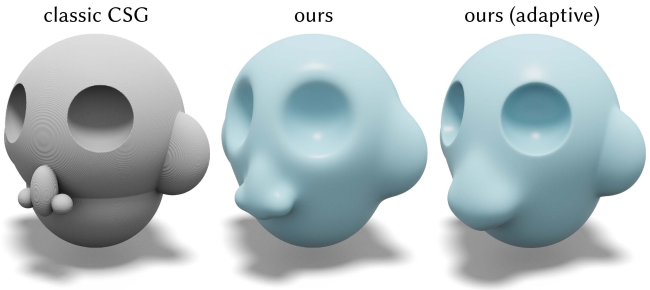
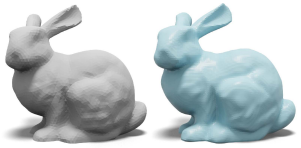


Fig. 9. *Unlike classic CSG which can only model hard booleans (left), our method enables both crisp and smooth outputs (middle) and can control the smoothness adaptively (right) at the primitive level.*

5.2 Single Shape Inverse CSG with Gradient Descent



Our approach enables us to simply use gradient descent to optimize a CSG tree that outputs a given shape (see Fig. 10), even for smooth organic objects (see inset). Our method starts with a *full* binary CSG tree (each boolean node has exactly two children, each primitive node is a leaf node) with randomly initialized (unified) boolean operators $\mathcal{B}_c$ and primitive parameters (see Fig. 1). Given a ground truth occupancy function, we minimize the mean square error between the output occupancy from the CSG tree and the ground truth with the ADAM optimizer [Kingma and Ba 2015]. To enforce our unified boolean operators converge to one of the boolean operations, we use a temperetured softmax function (see App. A for implementation details). At the end of the optimization,

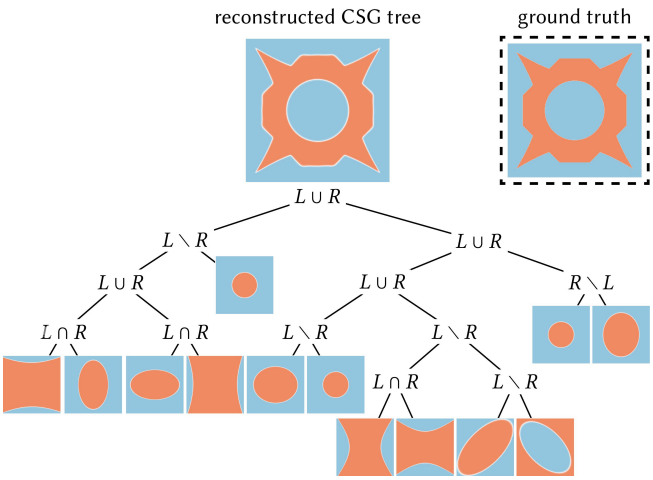


Fig. 10. *After fitting, we run the classic CSG tree pruning [Tilove 1984] to reduce the size of the tree from 128 primitives initially down to 13.*

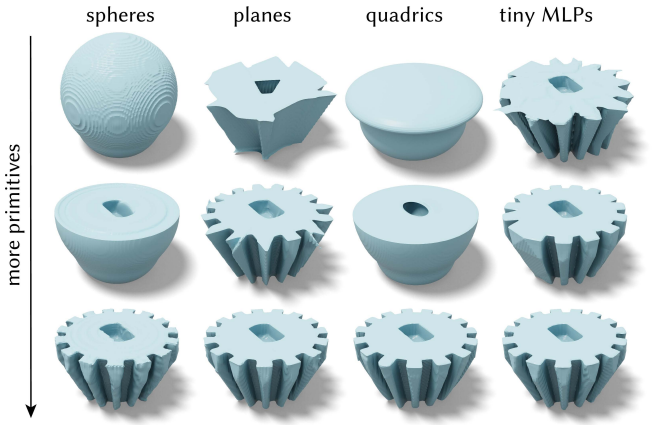


Fig. 11. *Our method is applicable to different types of primitives, including spheres, planes, quadrics, and tiny neural implicit networks. The choice of primitives will change the inductive bias of the optimization, leading to favoring different results when the amount of primitives is insufficient.*

our result is a set of optimized boolean nodes and primitive parameters. We then prune the redundant nodes with the classic CSG pruning [Tilove 1984] to obtain a more compact full binary CSG tree (see Fig. 10 and App. A for implementation details). Our approach is independent of the choice of primitive families. We are able to convert a shape into CSG of spheres, planes, quadrics, or even tiny neural networks (Fig. 11). Compared to fixing boolean nodes and only optimizing the primitive parameters, using our unified operator leads to better reconstruction (see Fig. 12).

5.3 CSG Generative Models

We demonstrate how our method can improve existing methods for fitting a shape dataset and generating CSG trees. Specifically, we

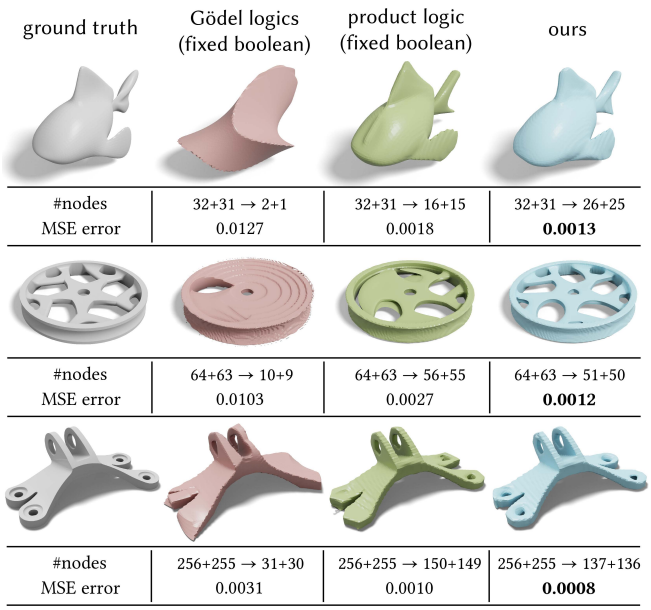


Fig. 12. *Our method (blue) allows optimization of the type of boolean operations. This leads to a better fitting result compared to the product logic (green) and the Gödel logic (red) with (randomly initialized) fixed boolean operations. We present the total number of nodes (primitive + boolean) before and after the optimization with pruning, showcasing improvements over different initial tree complexities. We also show the mean squared error on the occupancy evaluated on 2 million points sampled uniformly.*

Table 1. *By swapping the decoder in [Ren et al. 2021] with a decoder based on our unified boolean operator, we achieve improvements in quantitative metrics, including the mean squared error (MSE), classification accuracy, and F-score, on the ShapeNet dataset [Chang et al. 2015]. We provide the maximum number of nodes in the decoder (#primitives + #boolean nodes) and show that, despite being more compact, our approach still leads to better reconstruction.*

	MSE	*Accu.*	*F-score*	*total #nodes*
Ren et al. 2021	0.049	0.912	0.938	256+65792
Ours	**0.018**	**0.982**	**0.978**	**512+511**

focus on improving a hypernetwork approach proposed by Ren et al. [2021]. In short, given a point cloud, they propose to train a hypernetwork conditioned on the point cloud to output the parameters of their proposed CSG tree structure with pre-determined boolean operations. To demonstrate improvements over their method, we conduct the experiment on the same dataset, loss function, and hypernetwork, but we replace their CSG tree structure with our fuzzy boolean CSG tree. With such change, we demonstrate noticeable improvement over both qualitative Fig. 13 and quantitative Tab. 1 evaluations. The baseline [Ren et al. 2021] is based on their implementation and their pre-trained model weights.

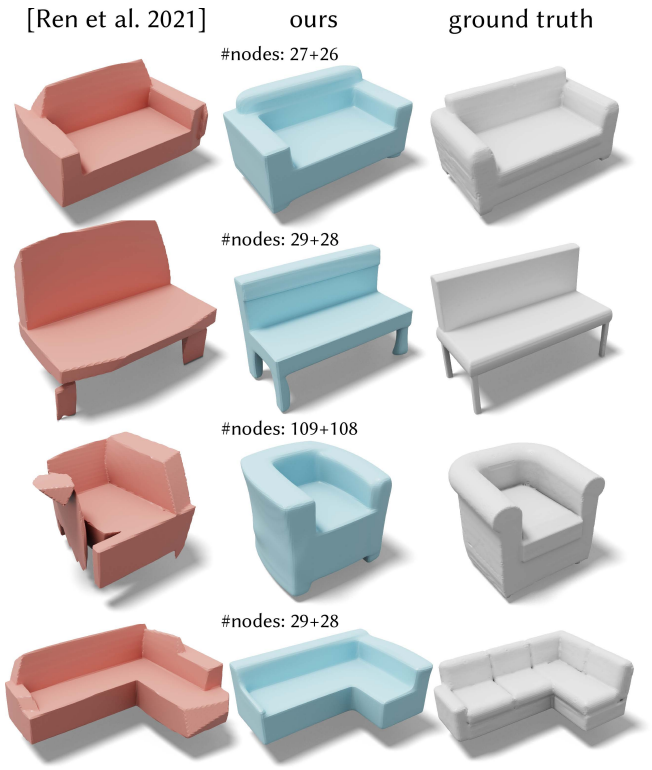


Fig. 13. *By simply replacing the CSG tree structure in [Ren et al. 2021] with our method, our approach can obtain qualitative improvement in fitting the ground truth shape over their approach. We also report the number of nodes (#primitives + #boolean) after pruning for our method. Note that the raw output of [Ren et al. 2021] requires around 15K boolean operations with many redundant ones. Thus, we do not report their number because a pruning method for them would be required for a fair comparison on the compactness of the tree.*

6 LIMITATIONS & FUTURE WORK

We introduce a unified differentiable boolean operator for solid shapes with soft occupancy. Our approach enables optimization of both the primitives and the boolean operations with continuous optimization techniques, such as gradient descent. As a preliminary investigation, the efforts described here open a cast of new directions for future work as well as room for improvement. We describe next a set of the limitations and opportunities for additional next steps and research directions.

Optimizing Tree Structures. Although we have enabled differentiation through boolean and primitive nodes, currently, the structure of our tree is held fixed during optimization. Despite fitting the shape well, our approach often leads to complicated CSG trees as a result, even after pruning. We believe future research in optimizing among tree structures and identifying when to grow/prune/rotate the tree nodes would be beneficial to reduce tree complexity.

Tree Properties. The ability to optimize the tree structure could unlock optimizing the tree to have certain properties, such as compactness or editability. A well-known challenge of inverse CSG is that a shape can be constructed by an infinite number of different CSG trees. We suffer from the same issue that our approach only finds one of the trees, but there is no guarantee that the tree we obtain is, for instance, the most compact option.

Extended Fuzzy CSG System. In our work, we explore how fuzzy logic may be applied to CSG modeling. We evaluate the fuzzy counterparts of existing CSG operations (UNION, INTERSECTION, DIFFERENCE), but there are more fuzzy logic operators that do not exist in CSG traditionally. For instance, the *fuzzy aggregation* operator [Klir and Yuan 1995] can be perceived as a generalization of UNION or INTERSECTION on a collection of primitive shapes, instead of two. Adding such operators could enable new possibilities in tree structure optimization by, for instance, selecting which primitives to use when performing boolean operations.

Hardware Acceleration. Our current fuzzy CSG system is based on an un-optimized implementation of fuzzy logic operators. However, as shown in several other fields, fuzzy logic operators can be greatly accelerated with parallel hardware implementations (e.g., [Ontiveros-Robles et al. 2016]). A hardware-accelerated version of our CSG system based on fuzzy logic could accelerate our method to run in real time.

CSG Generative Models. Making CSG systems differentiable could be beneficial for future exploration on (black boxed) neural symbolic generative models [Ritchie et al. 2023] that output (white boxed) CSG tree parameters. As this is orthogonal to our contributions, we simply evaluate our method based on the off-the-shelf architecture in Sec. 5.3. Future work on better neural network architectures for tree generation would be beneficial to empower CSG generation.

CSG modeling with optimization offers alternatives to mesh-based geometry representations that can be compact and less resolution dependent. Our exploration of the fuzzy boolean operator brings the automatic production of CSG models one step closer and opens avenues for further advances including speed-ups and relaxed constraints on CSG hierarchies. Inspired by the connections between fuzzy logic and other graphics applications (e.g., image/volumetric compositing), exploring applications of fuzzy logic beyond CSG could be an interesting future direction.

REFERENCES

Loïc Barthe, Brian Wyvill, and Erwin de Groot. 2004. Controllable Binary Csg Operators for "soft Objects". *Int. J. Shape Model.* 10, 2 (2004), 135–154.

Mario I Chacón M. 2006. Fuzzy logic for image processing: definition and applications of a fuzzy image processing scheme. In *Advanced Fuzzy Logic Technologies in Industrial Applications*. Springer, 101–113.

Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing 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. *CoRR* abs/1512.03012 (2015). arXiv:1512.03012 http://arxiv.org/abs/1512.03012

Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. 2020. BSP-Net: Generating Compact Meshes via Binary Space Partitioning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 42–51.

Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. 2020. Cvxnet: Learnable convex decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 31–44.

Boyang Deng, Sumith Kulal, Zhengyang Dong, Congyue Deng, Yonglong Tian, and Jiajun Wu. 2022. Unsupervised Learning of Shape Programs with Repeatable Implicit Parts. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 37837–37850.

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 Transactions on Graphics (TOG)* 37, 6 (2018), 1–16.

Ioan Dzitac, Florin Gheorghe Filip, and Misu-Jan Manolescu. 2017. Fuzzy logic is not fuzzy: World-renowned computer scientist Lotfi A. Zadeh. *International Journal of Computers Communications & Control* 12, 6 (2017), 748–789.

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, GECCO 2019, Prague, Czech Republic, July 13-17, 2019*, Anne Auger and Thomas Stützle (Eds.). ACM, 1183–1191.

K. Gödel. 1932. Zum Intuitionistischen Aussagenkalkül. *Anzeiger der Akademie der Wissenschaften in Wien* 69 (1932), 65–66.

Olivier Gourmel, Loic Barthe, Marie-Paule Cani, Brian Wyvill, Adrien Bernhardt, Mathias Paulin, and Herbert Grasberger. 2013. A gradient-based implicit blend. *ACM Transactions on Graphics (TOG)* 32, 2 (2013), 1–12.

Herbert Grasberger, Jean-Luc Duprat, Brian Wyvill, Paul Lalonde, and Jarek Rossignac. 2016. Efficient data-parallel tree-traversal for BlobTrees. *Comput. Aided Des.* 70 (2016), 171–181. https://doi.org/10.1016/j.cad.2015.06.013

Karim Hamza and Kazuhiro Saitou. 2004. Optimization of Constructive Solid Geometry Via a Tree-Based Multi-objective Genetic Algorithm. In *Genetic and Evolutionary Computation - GECCO 2004, Genetic and Evolutionary Computation Conference, Seattle, WA, USA, June 26-30, 2004, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 3103)*, Kalyanmoy Deb, Riccardo Poli, Wolfgang Banzhaf, Hans-Georg Beyer, Edmund K. Burke, Paul J. Darwen, Dipankar Dasgupta, Dario Floreano, James A. Foster, Mark Harman, Owen Holland, Pier Luca Lanzi, Lee Spector, Andrea Tettamanzi, Dirk Thierens, and Andrew M. Tyrrell (Eds.). Springer, 981–992. https://doi.org/10.1007/978-3-540-24855-2_110

Kacper Kania, Maciej Zieba, and Tomasz Kajdanowicz. 2020. UCSG-NET-unsupervised discovering of constructive solid geometry tree. *Advances in Neural Information Processing Systems* 33 (2020), 8776–8786.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

George J. Klir and Bo Yuan. 1995. *Fuzzy sets and fuzzy logic - theory and applications*. Prentice Hall.

Qingde Li and Jie Tian. 2008. Blending Implicit Shapes Using Fuzzy Set Operations. *WSEAS Trans. Info. Sci. and App.* 5, 7 (jul 2008), 1230–1240.

Zoë Marschner, Silvia Sellán, Hsueh-Ti Derek Liu, and Alec Jacobson. 2023. Constructive Solid Geometry on Neural Signed Distance Fields. In *SIGGRAPH Asia 2023 Conference Papers, SA 2023, Sydney, NSW, Australia, December 12-15, 2023*, June Kim, Ming C. Lin, and Bernd Bickel (Eds.). ACM, 121:1–121:12.

Karl Menger. 1942. Statistical Metrics. *Proceedings of the National Academy of Sciences of the United States of America* 28, 12 (1942), 535–537. http://www.jstor.org/stable/87805

Emanuel Ontiveros-Robles, JL Gonzalez-Vazquez, Juan R Castro, and Oscar Castillo. 2016. A hardware architecture for real-time edge detection based on interval type-2 fuzzy logic. In *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 804–810.

Alexander A. Pasko, Valery Adzhiev, Alexei Sourin, and Vladimir V. Savchenko. 1995. Function representation in geometric modeling: concepts, implementation and applications. *Vis. Comput.* 11, 8 (1995), 429–446.

Inigo Quilez. 2013. Smooth Minimum. (2013). https://iquilezles.org/articles/smin/

Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, Haiyong Jiang, Zhongang Cai, Junzhe Zhang, Liang Pan, Mingyuan Zhang, Haiyu Zhao, and Shuai Yi. 2021. CSG-Stump: A Learning Friendly CSG-Like Representation for Interpretable Shape Parsing. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 12458–12467.

A. Ricci. 1973. A Constructive Geometry for Computer Graphics. *Comput. J.* 16, 2 (1973), 157–160. https://doi.org/10.1093/comjnl/16.2.157

Daniel Ritchie, Paul Guerrero, R. Kenny Jones, Niloy J. Mitra, Adriana Schulz, Karl D. D. Willis, and Jiajun Wu. 2023. Neurosymbolic Models for Computer Graphics. *Comput. Graph. Forum* 42, 2 (2023), 545–568.

V. L. Rvachev. 1963. On the analytical description of some geometric objects. *Reports of Ukrainian Academy of Sciences* 153, 4 (1963), 765–767.

NM Samuel, Aristides Requicha, and SA Elkind. 1976. Methodology and results of an industrial part survey. (1976).

Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. 2018. CSGNet: Neural Shape Parser for Constructive Solid Geometry. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, 5515–5523.

Alvy Ray Smith. 1995. Image compositing fundamentals. *Microsoft Corporation* 5 (1995).

Robert B. Tilove. 1984. A Null-Object Detection Algorithm for Constructive Solid Geometry. *Commun. ACM* 27, 7 (jul 1984), 684–694. https://doi.org/10.1145/358105.358195

Emile van Krieken, Erman Acar, and Frank van Harmelen. 2022. Analyzing differentiable fuzzy logic operators. *Artificial Intelligence* 302 (2022), 103602.

Q. Wu, K. Xu, and Jun Wang. 2018. Constructing 3D CSG Models from 3D Raw Point Clouds. *Comput. Graph. Forum* 37, 5 (2018), 221–232. https://doi.org/10.1111/cgf.13504

Rundi Wu, Chang Xiao, and Changxi Zheng. 2021. DeepCAD: A Deep Generative Network for Computer-Aided Design Models. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 6752–6762.

Brian Wyvill, Andrew Guy, and Eric Galin. 1999. Extending the CSG Tree - Warping, Blending and Boolean Operations in an Implicit Surface Modeling System. *Comput. Graph. Forum* 18, 2 (1999), 149–158.

Ronald R Yager. 1980. On a general class of fuzzy connectives. *Fuzzy sets and Systems* 4, 3 (1980), 235–242.

Fenggen Yu, Qimin Chen, Maham Tanveer, Ali Mahdavi Amiri, and Hao Zhang. 2023. DualCSG: Learning Dual CSG Trees for General and Compact CAD Modeling. https://arxiv.org/abs/2301.11497

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11768–11778.

Lotfi A Zadeh. 1965. Fuzzy sets. *Information and control* 8, 3 (1965), 338–353.

A IMPLEMENTATION DETAILS FOR INVERSE CSG

Initialization. Our method starts with a randomly initialized full binary CSG tree that consists of our fuzzy boolean nodes Eq. 12 and primitive shape represented as soft occupancy functions. We initialize the parameters of the boolean and primitive nodes with a uniform distribution between -0.5 and 0.5. As the required tree complexity is unknown, we initialize a "big" CSG tree (e.g., 1024 primitive shapes) to reduce the chance of having an insufficient number of primitives.

Primitive Choices. In terms of the choice of primitives, except the one in Fig. 11, we use quadric surfaces q

$$q(x, y, z) = q_0x^2 + q_1y^2 + q_2z^2 \tag{18}$$

$$+ q_3xy + q_4yz + q_5zx + q6x + q_7y + q_8z + q_9 \tag{19}$$

in all our experiments partly due to its popularity in industry [Samuel et al. 1976]. More crucially, we believe using a less expressive primitive (compared to MLPs) give us a clearer signal on the performance of our proposed boolean operator. This is because an expressive primitive family, such as a big neural network, is able to fit a shape even without using any boolean operations. Then we convert the quadric function into a soft occupancy function with the *sigmoid* function

$$o(x, y, z) = sigmoid(s \times q(x, y, z)) \tag{20}$$

where s is a trainable "sharpness" parameter to uniformly scale the quadric function to make it sharper or smoother. This allows the model to change the sharpness of quadric surface without changing the shape. Empirically, we notice a better convergence rate with a trainable sharpness.

Boolean Parameterization. The side effect of having a unified boolean operator in is the possibility of not converging the one of the boolean operations. We alleviate this issue by parameterizing $\mathbf{c}$ with $\tilde{\mathbf{c}} \in \mathbb{R}^4$ as

$$\mathbf{c} = softmax(\sin(\omega\tilde{\mathbf{c}}) \cdot t) \tag{21}$$

where $t \in \mathbb{R}$ is the temperature. We leverage the *softmax* function to ensure the resulting $\mathbf{c}$ is always a valid barycentric coordinate. We set the temperature t to a high value (e.g., $t = 10^3$) to encourage $\mathbf{c}$ to be numerically close to a one-hot vector for most parameter choices of $\tilde{\mathbf{c}}$. The $\sin(\omega\cdot)$ (with $\omega = 10$) function is to ensure boolean operator type can still be changed easily in the later stage of the optimization. Without it, changing $\mathbf{c}$ will require many iterations when $\tilde{\mathbf{c}}$ has a large magnitude because each gradient update only updates $\tilde{\mathbf{c}}$ a little. We observe that this parameterization of $\mathbf{c}$ converges to a one-hot vector in all our experiments, even though we only softly encourage most parameter choices of $\tilde{\mathbf{c}}$ to be one-hot vectors. We suspect this is because any in-between operations will have occupancy values away from 0 or 1, whereas the target shape has binary occupancy values, converging to in-between operations can still occur when imperfect fitting happens.

Optimization. We define the loss function as the mean square error between the output occupancy from the CSG tree and the ground truth occupancy, evaluated on some sampled 3D points. We sample the points with approximately 40% on the surface, 40% near the surface, and 20% randomly in the volume. We regenerate these sampled point every couple iterations (e.g., 10) to make sure we sample most areas in the volume. We use the ADAM optimizer [Kingma and Ba 2015] with learning rate 1e-3 to train our model.

Pruning. After training, we prune redundant primitive/boolean nodes with post-processing. To determine redundant nodes, we follow the definition proposed by Tilove [1984] to characterize each boolean or primitive node as either *W-redundant* or *∅-redundant*. Intuitively, given a boolean node and its two child subtrees, if a subtree can be replaced with a full (soft occupancy with all 1s) or an empty (soft occupancy with all 0s) function without changing the output after the boolean operation, then this node is redundant and can be removed. We generalize such a redundancy definition to fuzzy boolean operations by setting a small threshold (e.g., mean squared soft occupancy error 10^{-3}) to determine whether the difference after replacing a subtree with full/empty function is small enough. With the notion of redundancy, we visit each node in the CSG tree in *post-order* and delete the node (including its children) if it is classified as a redundant node. We demonstrate the effectiveness of such a simple pruning strategy to greatly reduce the complexity of the optimized CSG tree in Fig. 10.

Linear Time CSG Forward Pass. During training, the full binary tree structure can be implemented in parallel for each layer by leveraging the fact that the number of node is pre-determined. However, after pruning, the tree structure becomes irregular. Running the forward pass after pruning requires graph traversal from the leaf primitive nodes to the boolean nodes and all the way to the root boolean node. To facilitate efficient inference, we employ the linear time traversal algorithm proposed by Grasberger et al. [2016] to speed up the forward pass. Their key idea is to traverse the CSG tree in post-order and push/pop intermediate results from a *stack*. This traversal has a continuous memory storage of all the nodes and only requires reading each node once.