Boundary-Sampled Halfspaces: A New Representation for Constructive Solid Modeling

XINGYI DU, Washington University in St. Louis, USA
QINGNAN ZHOU and NATHAN CARR, Adobe Research, USA
TAO JU, Washington University in St. Louis, USA

We present a novel representation of solid models for shape design. Like Constructive Solid Geometry (CSG), the solid shape is constructed from a set of halfspaces without the need for an explicit boundary structure. Instead of using Boolean expressions as in CSG, the shape is defined by sparsely placed samples on the boundary of each halfspace. This representation, called Boundary-Sampled Halfspaces (BSH), affords greater agility and expressiveness than CSG while simplifying the reverse engineering process. We discuss theoretical properties of the representation and present practical algorithms for boundary extraction and conversion from other representations. Our algorithms are demonstrated on both 2D and 3D examples.

CCS Concepts: • **Computing methodologies** → **Volumetric models**; *Shape analysis.*

Additional Key Words and Phrases: CSG, shape design, reverse engineering, arrangements, graph cut

ACM Reference Format:
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 (August 2021), 15 pages. https://doi.org/10.1145/3450626.3459870

1 INTRODUCTION

Solid modeling has wide-ranging applications in design, entertainment, engineering, and manufacturing. Computer representations of solid shapes have been extensively studied over the past few decades [Hoffmann 1989]. One of the most successful representations is Constructive Solid Geometry (CSG) [Requicha and Voelcker 1977]. In CSG, a complex shape is built from a set of simpler shapes (called *halfspaces*) using a sequence of Boolean operations on sets, including union (∪), intersection (∩), and difference (\). A key benefit of CSG, compared to other representations such as boundary representations (B-reps), is that it guarantees the solidity (i.e., watertightness) of the shape without maintaining an explicit boundary structure. This makes CSG particularly simple to use and one of the essential tools for shape design.

However, CSG has several limitations that are rooted in Boolean operations. First, composing a shape made up of multiple halfspaces often requires a complex Boolean expression (often known as the CSG tree). Maintaining and updating this expression, in addition to the halfspaces, may add extra burden to the user during shape editing. Second, it is well-known that Boolean expressions cannot describe all possible shapes bounded by a set of halfspaces, and additional *separating halfspaces* might be needed even though they do not contribute to the shape's geometry [Shapiro and Vossler 1991]. For example, given the two oval-shaped halfspaces in Figure 2 (a), Boolean operations can only create the first four shapes in (b), whereas the last two shapes in (b) each requires an extra linear halfspace (green) to be describable by CSG. The need for these "hidden" halfspaces not only makes both shape design and reverse engineering more challenging, but also limits CSG to mostly simple primitives (e.g., planes, spheres, cylinders, etc.) where the separating halfspaces are relatively simple to construct. Third, given a shape and a set of halfspaces, there may exist a large number of different Boolean expressions that can express the same shape. Finding a Boolean expression that is intuitive for downstream editing remains a major challenge in reverse engineering CSG shapes from other representations [Buchele and Crawford 2003; Du et al. 2018; Fayolle and Pasko 2016; Fayolle et al. 2008; Hamza and Saitou 2004; Silva et al. 2005; Wu et al. 2018].

In this paper, we propose a new halfspace-based representation of solid shapes that does not rely on Boolean operations. The composition of the halfspaces is expressed by sparse *samples* on the parts of the halfspace boundaries that bound the shape. Given the halfspaces and samples, the shape boundary is defined as the subset

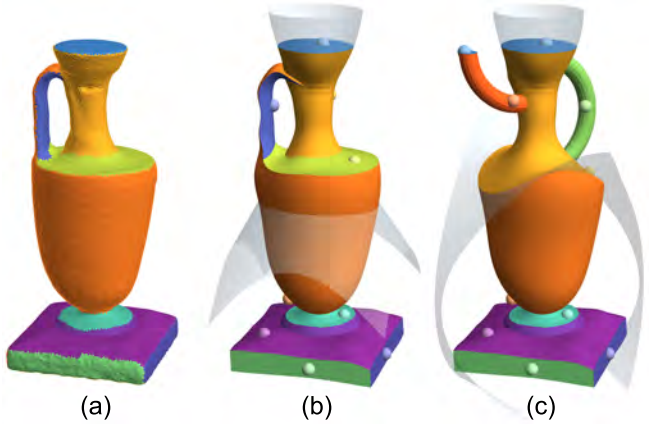


Fig. 1. A segmented shape (a) is converted into our representation (b), which consists of halfspaces associated with sparse samples (colored spheres). Each halfspace is either a simple primitive (e.g., plane, sphere, etc.) or a free-form implicit surface (one is shown in transparency). The representation can be easily edited by modifying the halfspaces and/or their samples (c).

Authors' addresses: Xingyi Du, Washington University in St. Louis, USA, du.xingyi@wustl.edu; Qingnan Zhou; Nathan Carr, Adobe Research, USA; Tao Ju, Washington University in St. Louis, USA.

0730-0301/2021/8-ART53
https://doi.org/10.1145/3450626.3459870

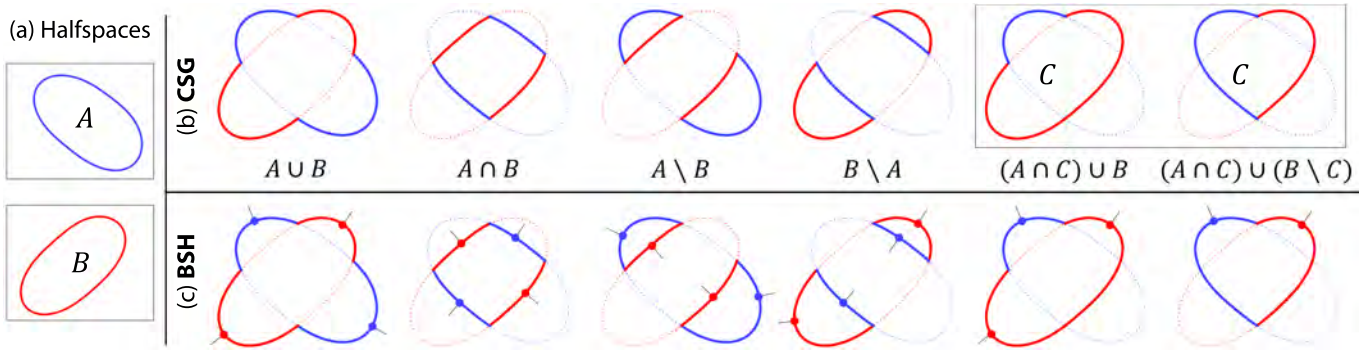


Fig. 2. Comparing CSG (b) and BSH (c) in representing six different shapes that are bounded by two halfspaces A, B (a). To represent the last two shapes, CSG requires an additional *separating halfspace* C (halfplane on the left of the green dashed line), whereas BSH does not. The line segment at each sample point in BSH indicates the outward orientation of the respective halfspace.

of the union of halfspace boundaries that contains the most samples, has the minimal area, and satisfies certain constraints. These constraints enforce the solidity of the shape, preserve the orientation of the halfspaces, and avoid artifacts caused by area minimization (e.g., short-cutting). We call our representation *Boundary-Sampled Halfspaces* (or BSH). As examples, Figure 2 (c) shows the BSH representations of the CSG shapes in (b).

Replacing Boolean expressions by boundary samples allows BSH to overcome the limitations of CSG mentioned earlier. Since the samples lie on the shape boundary, they can be conveniently visualized and manipulated. With enough samples, BSH can express *all* possible shapes bounded by a given set of halfspaces without needing additional "hidden" halfspaces as in CSG. In practice, we observe that only a small number of samples is sufficient (e.g., Figure 2 (c)). Lastly, without the need for recovering a Boolean expression, reverse engineering BSH is much more straight-forward than CSG. Both the halfspaces and samples can be readily obtained from a segmented shape. With these properties, BSH is well-suited to represent piecewise smooth shapes, such as the one shown in Figure 1, in an easily editable form.

We make several technical contributions in this paper:

- We propose the BSH presentation (Section 3). The representation is inspired by previous work on piecewise planar shape reconstruction [Bauchet and Lafarge 2020; Boulch et al. 2014; Chauve et al. 2010; Oesau et al. 2014; Verdie et al. 2015], and we introduce a novel constraint to avoid artifacts caused by area minimization when the samples are sparse.
- We demonstrate several theoretical properties of BSH (Section 4). Most notably, we show that BSH can describe any solid shape using only the halfspaces bounding the shape and a bounded number of samples (Proposition 4.1).
- We present a theoretically complete algorithm, and a practically efficient variant, for extracting the shape boundary defined by BSH (Section 5). These algorithms combine state-space search with graph-cuts over the arrangement of the halfspaces.
- For reverse-engineering, we present a simple but effective heuristic algorithm for generating a minimal set of samples that reproduces a target shape (Section 6).

Our algorithms are demonstrated on both 2D and 3D examples in Section 7. A key limitation of our algorithms is their scalability, because boundary extraction requires the computation of the 3D arrangement of all halfspaces. We discuss ideas to improve scalability and other future directions in Section 8.

2 RELATED WORK

2.1 Representations of solid models

Various researchers have studied the classifications of solid model representations [Requicha 1980; Rossignac and Requicha 1999; Shapiro 2002]. For example, Rossignac and Requicha [1999] grouped representations into three categories. Boundary representations (B-reps) represents the shape's boundary as a set of non-overlapping parametric or trimmed patches that form a closed manifold. Constructive representations, which include CSG but also others like twisting, bending, and Minkowski sums, capture the process in which a shape is constructed by a sequence of operations. Finally, spatial decomposition schemes, such as voxel grids, octrees, and binary spatial partitioning (BSP), represent the solid interior as the union of spatial *cells*. We refer readers to textbooks such as [Hoffmann 1989] for a thorough discussion on these representations and associated techniques.

BSH draws elements from all three categories in Rossignac's classification. Like CSG, BSH composes a complex shape from a set of simpler shapes (halfspaces). Like B-reps, BSH defines a solid by its boundary, which is selected from the union of all halfspace boundaries. Computationally, BSH is represented as a union of spatial cells in the arrangement of the halfspaces.

2.2 CSG

Pioneered by Requicha and Voelcker in the 70's [1977] and defined by *regularized* Boolean set operations on halfspaces, CSG has become one of the most popular approaches for solid modeling, due to its guarantee of solidity and the ease of control. The most commonly used halfspaces are parameterized primitives, such as boxes, spheres,

cones, cylinders, and tori, but other forms of halfspaces have been extensively studied in the research community, such as implicit functions [Nielson 2004; Pasko et al. 1995], parametric surfaces [Casale and Bobrow 1989; Keyser et al. 2004], and meshes [Bernstein and Fussell 2009; Douze et al. 2017; Feito et al. 2013; Pavić et al. 2010; Smith and Dodgson 2007; Wang 2010; Xu and Keyser 2013]. CSG has been extended to support other types of constructive modeling operations such as blending [Rossignac and Requicha 1984], offsetting [Rossignac and Requicha 1986], and warping [Wyvill et al. 1999]. Variants of CSG, such as the union of convex shapes [Chen et al. 2020], have also been proposed for shape representation.

Shapiro and Vossler [1991] were the first to study the expressiveness of CSG. Their *describability theorem* states that a shape can be represented by a given set of halfspaces using Boolean operations if and only if all connected components in each *canonical intersection term* of the halfspaces have the same inside/outside classification with respect to the shape. However, even simple shapes may fail the describability criteria (e.g., last two shapes in Figure 2 (b)), and additional *separating halfspaces* are needed for such shapes to be describable by CSG. Shapiro and Vossler [1993] showed that linear separating halfspaces are sufficient for shapes bounded by piecewise quadrics (such as the primitives mentioned above, except tori). Several heuristics, such as plane enumeration [Du et al. 2018] and oriented bounding planes [Fayolle and Pasko 2016; Wu et al. 2018], have been proposed for constructing linear separating halfspaces. However, shapes bounded by higher-order surfaces (e.g., the Vase in Figure 1) would require separating halfspaces of higher order [Shapiro and Vossler 1993], and their construction remains an untreated problem.

A key challenge in reverse engineering CSG shapes is to recover a simple and intuitive Boolean expression for downstream editing. Even when the set of halfspaces is fixed, the same shape can be defined using different Boolean expressions with drastically different complexity. The problem of simplifying a given Boolean expression of halfspaces (e.g., to minimize the number of literals) is known to be NP-hard [Shapiro and Vossler 1991]. This problem is still under active research, and example strategies include greedy heuristics [Buchele and Crawford 2003], genetic algorithms [Fayolle and Pasko 2016; Fayolle et al. 2008; Hamza and Saitou 2004; Silva et al. 2005], binary optimization [Wu et al. 2018] and program synthesis [Du et al. 2018]. As we show in Section 7, even the state-of-the-art methods may produce complex Boolean expressions that are difficult for humans to understand or manipulate.

2.3 Piecewise planar reconstruction

Our work is inspired by methods for reconstructing piecewise planar shapes, such as indoor environments [Boulch et al. 2014; Oesau et al. 2014], architectural structures [Chauve et al. 2010; Verdie et al. 2015], and general man-made shapes [Bauchet and Lafarge 2020]. A common theme in these works is formulating reconstruction as a labelling problem on the arrangement of planar proxies detected from the input scan. The labelling energy typically consists of a data term that leverages the availability of a dense set of input points and scanner information (e.g., lines of sight), and a simple regularization term that minimizes the total surface area. As we will show in Section 3, when the data points are sparse, minimizing the area alone leads to undesirable "short-cutting" artifacts. Boulch and coworkers [Boulch et al. 2014; Langlois et al. 2019] introduced additional regularization terms that minimize the length of edges and number of corners on the reconstructed surface. While these terms reduce short-cuts, they cannot eliminate them (see Figure 5). Furthermore, these terms introduce additional parameters to the labeling energy that need to be tuned. Our BSH formulation adopts a novel connectivity constraint that eliminates short-cutting while keeping the formulation parameter-free.

2.4 Arrangements

To extract the boundary of a BSH shape, our algorithm first needs to compute the arrangement of the halfspaces. While algorithms for computing the arrangement of lines and planes are well-established [Agarwal and Sharir 2000], computing geometrically and topologically accurate arrangements of curved (parametric or algebraic) geometry is significantly more challenging, and exact algorithms are only known for 2D curves [Alberti et al. 2008; Berberich et al. 2012; Lien et al. 2014] and 3D quadrics [Dupont et al. 2007; Mourrain et al. 2005; Schömer and Wolpert 2006]. Since we are computing a polygonal approximation of the shape's boundary, exact arrangements are not necessary, and we instead compute the arrangement after first polygonizing the boundary surface of each halfspace. This allows us to leverage recent development of robust and efficient algorithms for computing arrangements of closed meshes [Cherchi et al. 2020; Zhou et al. 2016].

3 SHAPE REPRESENTATION

The input to our representation is set of *halfspaces*, whose boundaries are denoted as $\{b_1, \ldots, b_n\}$, and a set of *samples* s_i located on each boundary b_i. A halfspace defines a solid region, or more precisely, a regular semi-analytic subset of the space $\mathbb{R}^d$ [Shapiro and Vossler 1991]. A halfspace can assume any representation, such as an implicit function, a B-rep, a CSG, or a BSH as defined here. We assume that each boundary b_i is a $(d-1)$-dimensional manifold, and the intersections between boundaries are generic and have a finite combinatorial structure. We also assume that most (if not all) of the samples also lie on the shape boundary.

The shape is defined as another halfspace whose boundary, denoted as b, is taken as a subset of the union of all input boundaries $\cup_i b_i$. Intuitively, the shape is *composed* of some part from each input shape. However, merely bounding a halfspace and being a subset of the input boundaries is far from sufficient to define the shape. Our definition of the shape boundary b is based on several additional criteria, which we discuss next. While similar criteria have been considered in context of shape reconstruction [Bauchet and Lafarge 2020; Boulch et al. 2014; Chauve et al. 2010; Oesau et al. 2014; Verdie et al. 2015], we make some new observations and propose a novel criteria (*sample-connectedness*) to better handle sparse samples that are desirable for shape design.

3.1 Shape criteria

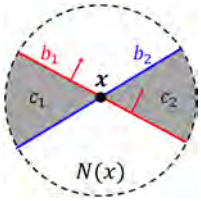
Sample coverage As the samples are expected to lie on the shape, we ask the shape boundary b to contain as many samples as possible.

Note that we do not *require* b to contain all samples. This allows our representation to be more tolerant to inaccuracy in the input, for example when the samples are provided by casual interactions or read in from a noisy point cloud.

Orientation preservation It is natural to ask that each input boundary b_i to preserve its inside/outside orientation on b. More precisely, let Ωb_i be the interior of the halfspace bounded by b_i, and $N(x)$ the local neighborhood of a point x, we define:

Definition 3.1. A halfspace boundary $b \subseteq \cup_i b_i$ is *orientation-preserving* if for any $i = 1, \ldots, n$ and any point $x \in b \cap b_i$ such that $x \notin b_j$ for all $j \neq i$ (that is, b_i is the only boundary that contains x), then $N(x) \cap \Omega b_i = N(x) \cap \Omega b$ (that is, both b_i and b give the same inside/outside classification to the local neighborhood of x).

Although this criterion has been previously considered [Chauve et al. 2010], we make a new observation that *any orientation-preserving and halfspace-bounding subset $b \subseteq \cup_i b_i$ is a $(d-1)$-manifold.* For example, all shapes in Figure 3 are orientation-preserving except for the non-manifold shape in (a). We give an argument for the manifoldness in $d = 2, 3$ dimensions. Suppose otherwise that b is non-manifold at some point x. Since each b_i is a manifold, x must lie on the intersection between multiple b_i. Since b is non-manifold at x, the local neighborhood of x interior to b, $N(x) \cap \Omega b$, has multiple connected components. Consider any two of these components, noted by c_1, c_2 (see insert for an illustration in 2 dimensions). Due to generic intersections of input boundaries, it can be verified by case enumeration in 2 and 3 dimensions that there exists some b_i containing x (e.g., b_1 in the insert) such that it bounds both c_1 and c_2 but from opposite sides (e.g., c_1 is inside b_i but c_2 is outside), which violates the orientation preservation criterion.



Compactness Preserving boundary orientations and maximally covering samples are still not sufficient to define the shape. For example, all four shapes in Figure 3 (b-e) meet these two criteria. A regularization term commonly considered in surface reconstruction is to ask the boundary to have a minimal measure (e.g., curve length, surface area, etc.). This criterion encourages compact shapes (e.g., Figure 3 (e)), which often better capture design intentions.

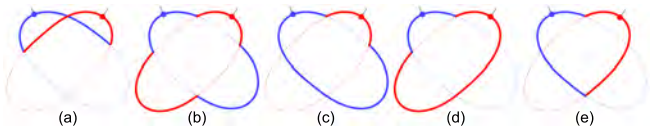


Fig. 3. Possible shapes defined by the same halfspaces and samples: (a) not orientation-preserving; (b,c,d) orientation-preserving but not compact; (e) both orientation-preserving and compact.

Sample-connectedness The side effect of encouraging compactness, particularly when there are only few samples, is that parts of the shape can be "short-cut" to reduce the boundary size. This is illustrated by the example in Figure 4. The three halfspaces and three

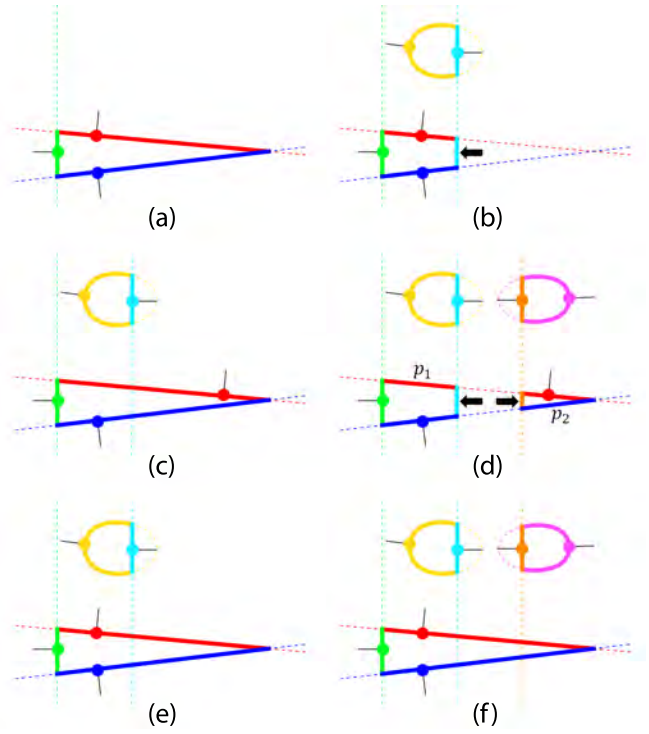


Fig. 4. Starting from an initial triangle (a), adding a semi-circle on the top (b) results in short-cutting of the triangle if only boundary length is minimized. Moving a sample recovers the missing portion of the triangle (c), but more short-cutting appears after adding another semi-circle on the top (d). Enforcing sample-connectedness avoids short-cutting without moving the samples (e,f). Arrows indicate short-cutting pieces.

samples in (a) define a triangular shape. After adding a new component at the top in (b), however, minimizing boundary length results in the triangular shape being truncated by one of the halfspaces that define the top component (indicated by the arrow).

One approach to avoid short-cuts is moving or adding new samples. For example, the missing part of the triangle in Figure 4 (b) can be recovered by relocating one of the samples (red), as shown in (c). However, new short-cuts may appear after more edits to the shape, as shown in (d), where the middle section of the triangle is now truncated by two short-cutting pieces (marked by arrows). The short-cutting can be fixed again by relocating existing samples or adding new samples. This approach, however, leads to increased burden on the modeler and possibly a large number of samples.

Short-cutting is not a major problem for shape reconstruction from scanned inputs, where there are usually sufficient number of input points that cover the shape. Aiming at a slightly different goal, Boulch and coworkers [Boulch et al. 2014; Langlois et al. 2019] proposed to additionally minimize the combinatorial complexity of the boundary surface, measured in terms of the number of "corners" (where three or more halfspaces meet) and length of "edges" (where two halfspaces meet). Although short-cuts often increase the combinatorial complexity (e.g., compare Figure 4 (b) and (c)), this is not always the case. For example, the shape in Figure 5 (a) is defined by

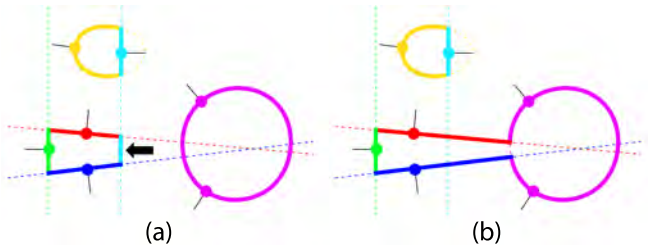


Fig. 5. The boundary defined without (a) and with (b) sample-connectedness. Short-cutting occurs on the first shape (see arrow) but not on the second shape. Note that both shapes have the same number of *corners* (where two halfspaces meet) on their boundary.

an input similar to Figure 4 (b) with the addition of a circle, and a similar short-cut takes place (indicated by the arrow) that stops the bottom triangle from meeting the circle. Note that this boundary has the same number of corners (6) as the one in Figure 5 (b), where there is no short-cutting.

To eliminate short-cuts, our key observation is that a short-cutting piece that belongs to some input boundary b_i is always *disconnected* from those parts of b_i that are meant to be on the shape boundary b. For example, the short-cutting piece in Figure 4 (b), which belongs to the cyan halfspace, is disconnected from the cyan segment that bounds the semi-circle at the top of the shape. A simple criteria to prevent short-cutting is to ask b to be *sample-connected* as defined below:

Definition 3.2. A set of patches b is called *sample-connected* if for any input boundary b_i, each connected component of $b_i \cap b$ contains at least one sample of s_i.

For example, imposing sample-connectedness would lead to short-cut-free shapes in Figure 4 (d,e) and Figure 5 (b), without the need to relocate or add samples. Note that applying this criteria requires the input to contain at least one sample for each connected component of patches on the shape that belong to the same halfspace. In our experiments, we find this to be a reasonable requirement for a large class of shapes.

3.2 Shape definition

Considering the criteria above, we define the BSH shape as follows:

Definition 3.3 (Boundary-Sampled Halfspaces). Given halfspace boundaries b_i and their samples s_i, the BSH shape is a halfspace bounded by some $b \subseteq \cup_i b_i$ that is orientation-preserving (Definition 3.1), sample-connected (Definition 3.2), and minimizing the following energy:

$$E(b) = \lambda |S \setminus b| + |b| \tag{1}$$

where $|\cdot|$ gives the cardinality of a finite set or geometric measure of a continuous set, $S = \cup_i s_i$, and λ is any constant that is greater than $|\cup_i b_i|$.

The two terms of $E(b)$ measure the number of missing samples on b and the size of b, respectively. The constant λ plays the role of prioritizing sample coverage over compactness, and its lower-bound is set such that minimizing $E(b)$ is equivalent to minimizing the tuple $\{|S \setminus b|, |b|\}$ in lexicographical ordering. As a result, the actual value of λ has no impact on the optimal boundary b, and the definition is parameter-free.

4 PROPERTIES

We will show that the shape definition given above enjoys several desirable properties. First, the shape as defined always exists and is unique (assuming geometric genericity). More importantly, any solid shape can be defined as BSH using a set of halfspaces bounding the solid and a sufficient number of samples.

4.1 Existence

We can show that a boundary b meeting the constraints in Definition 3.3 always exists. Consider the set H of all halfspace boundaries $b \subseteq \cup_i b_i$ that are orientation-preserving and sample-connected. It suffices to show that H is finite and non-empty, which would imply that a member $b \in H$ minimizing the energy $E(b)$ would always exist. H is a non-empty set, because it has at least one member, the empty boundary $b = \emptyset$. To see that H is a finite set, consider the partitioning of each input boundary b_i by the remaining input boundaries, and we call a region of b_i in this partitioning a *patch*. Since each member $b \in H$ bounds a halfspace, and b is subset of $\cup_i b_i$, it follows that b must consist of a collection of patches. On the other hand, since we assume that the intersections of boundaries have a finite structure, there are only finite number of patches on all b_i. Therefore H, whose elements are subsets of a finite number of patches, must be finite.

4.2 Uniqueness

Using the argument above, the boundary b defined in Definition 3.3 is the member of a finite set H that has the least energy $E(b)$. Since $E(b)$ considers the geometric measure of b, any tie between multiple (but finite) members of B with the same minimal energy can be broken by an infinitesimal perturbation of the geometry of the input boundaries. Hence, generically speaking, the BSH shape is unique.

We give a non-generic example in Figure 6. In this case, the shaded region is a perfect square, which results in two valid BSH shapes (one with two separate components and the other with a single component) with equal boundary length. The tie would be broken as soon as one of the halfspaces that bound the square region is perturbed.

4.3 Describability

Unlike CSG, BSH can fully describe any solid shape bounded by the given halfspaces, if sufficient and well-chosen samples are given:

Proposition 4.1 (Describability). *Consider any halfspace boundary $b \subseteq \cup_i b_i$ that is orientation-preserving. Then there exists a (possibly empty) sample set S such that b is the unique boundary defined by Definition 3.3.*

Proof. Following the earlier arguments on existence, b is made up of a subset of patches on the input boundaries. Consider the set $S = \{s_1, \ldots, s_n\}$ such that s_i consists of one sample for each patch of b_i that lies on b (the geometric location of the sample

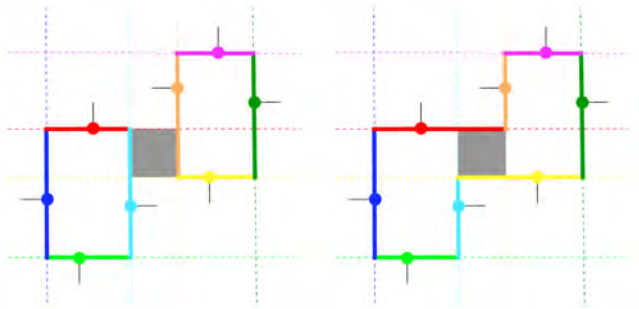


Fig. 6. A non-generic input (the shaded region being a perfect square) that results in two valid BSH shapes.

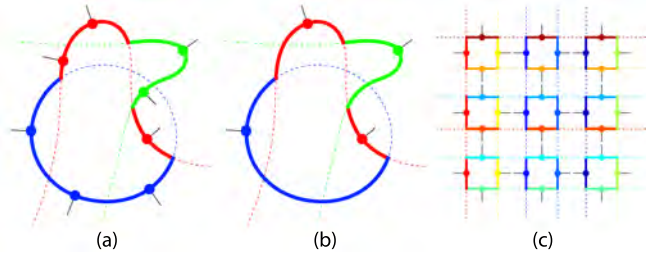


Fig. 7. (a,b): The same boundary can be defined with different sets of samples. (c): An example of a shape where the number of samples needed is quadratic to the number of halfspaces.

on the patch does not matter), as illustrated in Figure 7 (a). We will show that the boundaries $\{b_1, \ldots, b_n\}$ and samples S uniquely define b via Definition 3.3. First, b is sample-connected, since each connected component of $b_i \cap b$ contains at least one patch, and each such patch includes a sample in s_i. Next, we show that there exists no other boundary $b' \subseteq \cup_i b_i$, which is also halfspace-bounding, orientation-preserving, and sample-connected, such that $E(b') \leq E(b)$. Assuming otherwise, then by the formulation of E we know that $|S \setminus b'| \leq |S \setminus b|$ and $|b'| \leq |b|$. Since b contains all samples S, $|S \setminus b'| = |S \setminus b| = 0$, and hence b' must contain all samples as well. Since each patch of b contains a sample, $b \subseteq b'$. On the other hand, since $|b'| \leq |b|$, we conclude that $b = b'$. □

The proof also shows that the maximum number of samples needed to describe the shape is upper-bounded by the total number of patches on the given boundaries. This bound is always finite, but it can be quite large. Even in the case that all b_i are planar, there are as many as $O(n^d)$ patches in $\mathbb{R}^d$. In practice, however, much fewer samples are needed to represent the shape, thanks to the compactness and sample-connectedness criteria. Figure 7 (a,b) show the same boundary defined by two sets of samples, a larger set (a) constructed as in the proof above, where there is one sample on each patch of the shape boundary, and a smaller (and minimal) set (b), where there is one sample for each connected component of patches of the same halfspace. However, for shapes like that shown in Figure 7 (c) (a set of disjoint squares on a 2D lattice), the number of connected components of patches of the same halfspace that cover the shape is proportional to the total number of patches on all halfspace boundaries. Hence as many as $O(n^d)$ samples would be needed to define such a shape.

5 BOUNDARY EXTRACTION

To extract the boundary defined by BSH, we formulate it as an inside/outside labelling problem over the solid cells of the *arrangement* of the input boundaries b_i. The shape boundary b is then constructed as the interface between cells of different labels. The advantage of this *dual* formulation is that, by construction, the output is a subset of the input boundaries and always bounds a halfspace.

If the sample-connectedness constraint is ignored, the remaining constraint (orientation-preservation) and the energy E (Equation 1) can be formulated as localized labelling energy involving a single cell or two adjacent cells. Such energy can be optimally and efficiently solved using graph-cuts, as similarly done for reconstructing piecewise planar shapes [Bauchet and Lafarge 2020; Boulch et al. 2014; Chauve et al. 2010; Oesau et al. 2014; Verdie et al. 2015]. However, enforcing sample-connectedness makes the labelling problem much more challenging, as the connectivity of boundary components cannot be measured locally.

In the following, we first describe the labelling formulation and graph-cut algorithm for extracting the shape boundary without the sample-connectedness constraint. We will then present a heuristic search algorithm for solving the sample-connected labelling problem that uses the graph-cut algorithm as a building block.

5.1 Without sample-connectedness

Consider the d-dimensional cells C in the arrangement of the boundaries $\{b_1, \ldots, b_n\}$. We say two cells are *adjacent* if they share a common patch (recall that a patch is a $(d-2)$-dimensional region in the partitioning of one input boundary by the remaining boundaries). We seek a binary labelling function $L : C \rightarrow \{1, 0\}$ (1 meaning inside and 0 meaning outside) over the arrangement cells that minimize the following energy:

$$h(L) = \sum_{c \in C} h_c(L(c)) + \sum_{\text{adjacent } \{c_1, c_2\} \subseteq C} h_{c_1,c_2}(L(c_1), L(c_2)) \quad (2)$$

where h_c, h_{c_1,c_2} are unary and binary terms to be explained below. These terms are defined such that, given an energy-minimizing labelling L, the union b of all patches shared by cells with different labels is the BSH shape without the sample-connectedness constraint (i.e., b is orientation-preserving and minimizes $E(b)$ in Equation 1).

The unary term captures the first term of the energy E (number of missing samples). To motivate our definition, let us first consider a patch p shared by two cells c_1, c_2, and let i_p be the index of the input boundary that contains p. Note that c_1, c_2 are on different sides of the boundary b_{i_p}. The shape boundary b will not contain p if both cells c_1 and c_2 have the same label, which means that exactly one of them has a label that differs from its inside/outside classification by b_{i_p}. We therefore "charge" the cost of the missing samples on p to that cell. The unary term for a cell c and a given label l, $h_c(l)$, is therefore defined as the total number of samples on those patches

bounding c that classify c differently from l,

$$h_c(l) = \lambda * \sum_{p \in \partial c, b_{i_p}(c) \neq l} |s_p|. \tag{3}$$

Here, ∂c is the set of patches bounding c, s_p is the set of samples on p, and $b_i(c)$ indicates the classification of c by boundary b_i such that $b_i(c) = 1$ or 0 if c is inside or outside b_i.

The binary term captures both the orientation-preservation constraint and the second term of E (boundary size). Consider two adjacent cells c_1, c_2, and let i_{c_1,c_2} be the index of the input boundary that separates them. The binary term is non-zero if the two cells have different labels. Specifically, if their labels agree with their classification by the input boundary $b_{i_{c_1,c_2}}$, then the cost is the geometric size of the common patches that they share. Otherwise, to enforce orientation-preservation, the cost is set to infinity. More precisely,

$$h_{c_1,c_2}(l_1, l_2) = \begin{cases} \sum_{p \in \partial c_1 \cap \partial c_2} |p|, & \text{if } l_1 \neq l_2 \text{ and } l_1 = b_{i_{c_1,c_2}}(c_1) \\ \infty, & \text{if } l_1 \neq l_2 \text{ and } l_1 \neq b_{i_{c_1,c_2}}(c_1) \\ 0, & \text{if } l_1 = l_2 \end{cases} \tag{4}$$

The labelling energy in Equation 2, in turn, can be formulated as the cost of an $s-t$ cut on a directed graph $G = \{V, E\}$. Note that a directed graph is necessary because the binary term (Equation 4) is asymmetric (i.e., $h_{c_1,c_2}(1,0) \neq h_{c_1,c_2}(0,1)$). The vertices V represent the cells of C with two additional *terminal* vertices, s and t. The edges E and their weights are defined as follows (see illustration in Figure 8):

- There is an edge $v_1 \rightarrow v_2$ for every (ordered) pair of non-terminal vertices $\{v_1, v_2\}$, whose weight is $h_{c_1,c_2}(1,0)$ where c_1, c_2 are the cells represented by v_1, v_2.
- There is an edge $s \rightarrow v$ and an edge $v \rightarrow t$ for every non-terminal vertex v, whose weights are respectively $h_c(0)$ and $h_c(1)$ where c is the cell represented by v.

An $s-t$ cut on G divides the vertices V into two subsets S, T such that $s \in S, t \in T$. The cost of the cut is the sum of weights associated with the *cut set*, which are edges connecting from a node in S to a node in T. A labelling on the arrangement cells C can be obtained from an $s-t$ cut on G by labelling each cell as 1 (resp. 0) if the corresponding vertex in G lies in subset S (resp. T). Similarly, an $s-t$ cut on G can be obtained from any labelling of C. It can be verified that, using the edge weights defined above, the cost of an $s-t$ cut on G is the same as the energy $h(L)$ of its corresponding labelling L on C. As a result, the energy-minimizing labelling can be obtained by computing the minimal-cost $s-t$ cut on G, which can be done efficiently using the Edmonds-Karp algorithm [Edmonds and Karp 1972]. As an example, the $s-t$ cut of the graph in Figure 8 (b) corresponds to the labelling in the arrangement in (a).

5.2 With sample-connectedness

Recall that the sample-connectedness constraint asks that, for any input boundary b_i, each connected component of $b \cap b_i$ must contain a sample. Checking for this constraint requires computing the connected components on the shape boundary b, which is not a local operation and hence cannot be formulated as a local energy like h in Equation 2. To enforce the constraint, we propose a best-first search algorithm that starts from a non-sample-connected solution (e.g., result of graph-cut) and iteratively explores solutions that get closer to being sample-connected. We will show that the search is *complete*, in the sense that it always returns the BSH shape in finite iterations. Although the computational complexity can be prohibitive, we found that a simple modification of the search leads to an approximate algorithm that is both efficient and effective in practice.

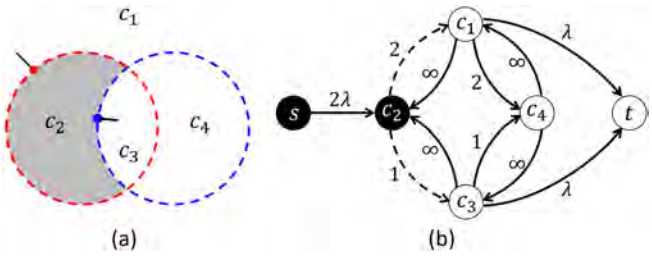


Fig. 8. (a): An arrangement of input boundaries with 4 cells (c_1, c_2, c_3, c_4) and 4 patches (each having a length of 1 or 2), and the energy-minimizing labelling (only c_2 is inside). (b): The corresponding directed graph with edge weights and the min $s-t$ cut (S vertices are black, T vertices are white, and cut set edges are dashed). Zero-weighted edges are not shown.

The basic idea of our search is iteratively running the graph-cut algorithm with more constraints of what patches can (or cannot) be used. The algorithm maintains a collection of *states* $\{r, b\}$, where $r \subseteq \cup_i b_i$ is a set of patches marked as "removed" and b is the minimal-energy boundary using the unmarked patches. The latter is computed by the graph-cut algorithm described above after setting $|p| = \infty$ in the binary term (Equation 4) for all marked patches $p \in r$. The search starts with a single state with no patches marked for removal. At each iteration, the state $\{r, b\}$ with the least energy $E(b)$ is deleted from the queue. If boundary b is not sample-connected, we identify a collection of patch sets on b, called *removable sets*, which are likely to not belong to the desired BSH boundary. For each removable set, a new state is created with patches $r \cup s$ marked for removal and a new boundary is computed by graph-cut. The algorithm terminates when the boundary b is sample-connected. The pseudo-code of the algorithm is provided in Figure 9.

A key piece of the algorithm is identifying the removable sets for a given boundary b. For the search to succeed, at least one removable set should not appear on the desired boundary. Since our goal is to achieve sample-connectedness, a natural candidate of a removable set is a sample-free connected component of $b \cap b_i$, for some b_i. We call such component an *island*. However, while some islands are short-cutting pieces that indeed do not lie on the intended shape (e.g., those marked by arrows in Figure 4 and 5), an island can also be part of the intended shape but separated from other components by short-cuts (e.g., p_1, p_2 in Figure 4 (d)). Nevertheless, we observe that, for any given sample-connected boundary b' (such as the desired BSH boundary), either an island of b or one of its adjacent patches on b do not appear on b':

Proposition 5.1. *Let $b, b' \subseteq \cup_i b_i$ be two halfspace bounding, orientation-preserving boundaries such that b is not sample-connected but b' is. For any island $I \subseteq b$, defined as a connected component of*

```
EXTRACT-BOUNDARY (B,S)
    b← graph-cut(B,S,∅)
    Q← ∅
    Q.push({∅, b})
    Repeat:
        {r, b}← Q.pop()
        If b is sample-connected:
            Return b
        Repeat for each removable set s of b:
            r′ ← r ∪ s
            b′ ← graph-cut(B,S,r′)
            Q.push({r′, b′})
```

Fig. 9. Pseudo-code of the heuristic search for extracting the BSH boundary given halfspace boundaries $B = \{b_1, \ldots, b_n\}$ and samples $S = \{s_1, \ldots, s_n\}$. Graph-cut(B,S,r) computes the BSH boundary for $\{B, S\}$ without the sample-connectedness constraint and without using patches r, and $Q.pop()$ returns the element $\{r, b\}$ with the least energy $E(b)$.

$b \cap b_i$ for some b_i that does not contain a sample, either $I \cap b' = \emptyset$ or there is some patch $p \in b$ adjacent to I such that $p \notin b'$.

Proof. Suppose the statement is false, that is, all patches in b adjacent to I are in b', and a subset $I' \subseteq I$ is in b' as well. Since b' is orientation-preserving, it is also a manifold (see Section 3.1). Let b_i be the input boundary that contains I. It follows from the manifoldness of b', and that all patches in b that surround I (which do not lie on b_i) are also on b', that I' is disconnected from other patches of b_i that lie on b'. Since I is an island of b, it does not contain any sample, and hence I' is also sample-free. So any connected component of I' is a connected component of $b' \cap b_i$ without a sample, which contradicts to the assumption that b' is sample-connected. □

The observation motivates us to define a removable set of b as *either an island of b or one of the patches of b adjacent to an island.* With this definition, we can show that the search algorithm is complete (see proof in Appendix A):

Proposition 5.2. *Given halfspace boundaries $B = \{b_1, \ldots, b_n\}$ and samples $S = \{s_1, \ldots, s_n\}$, the EXTRACT-BOUNDARY algorithm in Figure 9 terminates in finite steps and returns the BSH shape boundary defined by $\{B, S\}$.*

Despite its completeness, the best-first search has a prohibitive time and space complexity, since both the degree and depth of the search tree are upper bounded by the total number of patches. To make the algorithm practical, we adopt a *beam search* by keeping maximally K states in the queue at any time. Specifically, when a state is deleted from the queue, we compute all its children states, add them to the queue, and then prune the queue to keep only the K states with the least energy. We found that the most greedy (and efficient) setting using $K = 1$ can already produce optimal results in many of our test cases. Even in the case that the result is sub-optimal, it takes much less effort to fix (by moving or adding samples) than the result of the graph-cut algorithm without imposing sample-connectedness. For consistency, we adopt $K = 1$ in all our experiments.

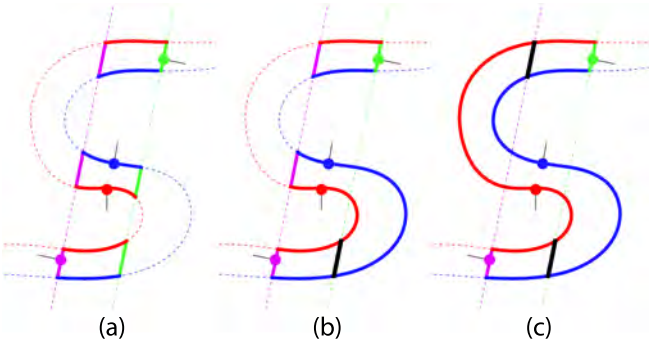


Fig. 10. Steps in the beam search ($K = 1$) for extracting the boundary of the "S" shape. Black segments in (b,c) are patches marked for removal at each state.

The beam search (with $K = 1$) is demonstrated in 2D in Figure 10. Without enforcing sample-connectedness, the initial graph-cut results in multiple short-cuts that minimize the boundary length. Each step of the search recovers more of the missing parts of the intended shape (letter "S") by running graph-cut on an increasingly restrictive search space with more patches marked as removed (colored black).

6 CONVERSION FROM OTHER REPRESENTATIONS

As a constructive representation, a BSH shape can be created from scratch by incrementally adding halfspaces and their samples (see examples in Section 7). A BSH shape can also be converted from other representations, such as a B-rep or a CSG. By the Describability Theorem 4.1, the conversion only needs to find a set of halfspaces that fully bound the shape and sufficient samples on those halfspaces. This is conceptually simpler than converting into CSG, which has to additionally infer the separating halfspaces and optimize for the Boolean operations on the halfspaces [Shapiro and Vossler 1991]. As we shall see below, the halfspaces and samples needed for representing a given shape as BSH can be obtained by either existing geometry processing methods or simple heuristics.

6.1 Creating halfspaces

If the input shape is represented as a B-rep (e.g., a polygonal mesh), the halfspaces can be created by first segmenting the shape boundary and fitting each segment by a suitable representation of a halfspace. Numerous methods are available for these tasks, including surface segmentation [Shamir 2008], surface fitting using solid primitives [Kaiser et al. 2019] and general implicit functions [Berger et al. 2017]. If the input shape is a CSG, we can simply use the same set of halfspaces that defines the shape (the separating halfspaces will be ignored after sample generation; see below).

6.2 Generating samples

It is straightforward to generate a sufficient (but possibly overly redundant) set of samples to represent an input shape. For a B-rep shape, dense samples can be obtained from the segments on the shape's boundary and associated with the corresponding fitted halfspaces. For a CSG shape, we can first compute the patches in the arrangement of the halfspaces that bound the shape by labelling

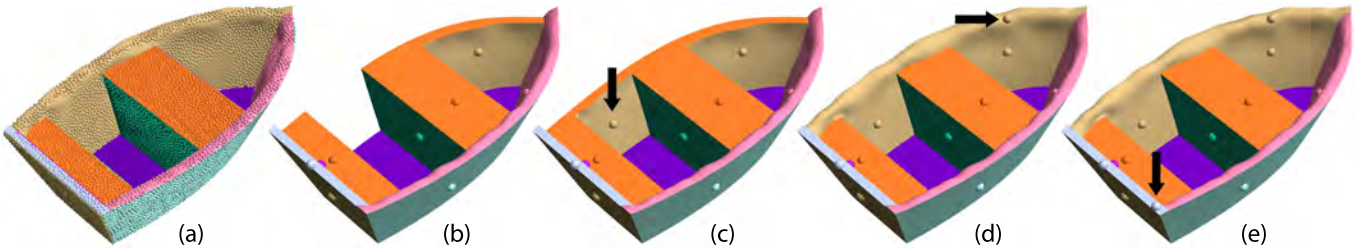


Fig. 11. Given the halfspaces and an initial (dense) set of samples converted from a mesh (a), our algorithm starts with a necessary set of 15 samples (b), one for each connected component of patches from the same halfspace, and incrementally adds one sample per iteration (c,d,e; added samples indicated by arrows) until they are sufficient to reproduce the same shape boundary as the dense samples.

the arrangement cells using the CSG boolean expressions. Based on the proof of the Describability Theorem 4.1, placing one sample on each such patch is sufficient to reproduce the shape as BSH (note that the separating halfspaces, which do not bound the shape, will not be associated with any sample).

The dense samples created above could be overwhelming for a user in downstream design tasks (e.g., Figure 11 (a)). To simplify user control, and to leverage the sparsity afforded by the BSH definition, we consider the problem of finding a minimal set of samples to represent a given shape. The problem can be formulated as follows: given halfspace boundaries $B = \{b_1, \ldots, b_n\}$ and some *target* boundary $b \subseteq \cup_i b_i$ (e.g., the BSH shape defined by the dense samples), find sample sets $S = \{s_1, \ldots, s_n\}$ such that b is the BSH shape defined by $\{B, S\}$ and that $|S|$ is minimal. While the hardness of this problem is unknown (we suspect it is NP-hard), we present a simple heuristic that has a theoretical guarantee of termination and significantly reduces the number of samples in our experiments.

The idea of this heuristic is to incrementally grow the sample set until the target boundary is reproduced. Since a BSH is sample-connected, we need at least one sample on each connected component of $b \cap b_i$ for each input boundary b_i. For visual intuitiveness, we place the sample at the geodesic center of the respective connected component. These samples form the initial sample set S. At each iteration of the algorithm, we compute the BSH shape boundary defined by the current sample set, which we denote by b'. If b' differs with the target b, we consider those patches on b that are missing on b' and are not associated with a sample yet (such patches always exist, as we shall prove below), and we add a sample to the largest of these patches. The pseudo-code of the algorithm is shown in Figure 12. The algorithm is guarantee to return a target-reproducing sample set (see proof in Appendix B):

Proposition 6.1. *Given halfspace boundaries $B = \{b_1, \ldots, b_n\}$ and a halfspace-bounding, orientation-preserving and sample-connected boundary $b \subseteq \cup_i b_i$, the algorithm GENERATE-SAMPLE in Figure 12 terminates in finite steps and returns a sample set S such that b is the BSH shape boundary defined by $\{B, S\}$.*

The termination guarantee above holds when the BSH boundary is extracted using the complete search algorithm in Figure 9, which has prohibitive complexity. Fortunately, we can show that the guarantee remains when the more efficient beam search is used instead

GENERATE-SAMPLES (B,b)
  //Initialize samples
  Repeat for each $b_i \in B$:
    $s_i \leftarrow \emptyset$
    Repeat for each connected component c of $b_i \cap b$:
      $x \leftarrow$ geodesic center of c
      $s_i \leftarrow s_i \cup \{x\}$
  //Incrementally add new samples
  Repeat:
    $b' \leftarrow$ EXTRACT-BOUNDARY($B, \{s_1, \ldots, s_n\}$)
    If $b = b'$:
      Return $\{s_1, \ldots, s_n\}$
    $p \leftarrow$ largest sample-free patch in $b \setminus b'$
    $i \leftarrow$ index of b_i containing p
    $x \leftarrow$ geodesic center of p
    $s_i \leftarrow s_i \cup \{x\}$

Fig. 12. Pseudo-code of the algorithm for generating samples on halfspace boundaries $B = \{b_1, \ldots, b_n\}$ that reproduce a target boundary $b \subseteq \cup_i b_i$ as BSH. It calls the EXTRACT-BOUNDARY algorithm in Figure 9.

and with the specific beam width of $K = 1$ (which is used in our experiments). This is stated and proved in Appendix B.

The algorithm is demonstrated in Figure 11 with the Boat example. With the initial samples placed on connected components of patches of the same halfspace boundary, the extracted boundary, shown in (b), misses a number of parts of the boat, such as sections on the side and along the top rim. The algorithm recovers those parts by adding three samples in three iterations (shown in (c,d,e)). The total number of samples (18) is significantly fewer than the initial dense samples (a) as well as the number of patches on the shape boundary (87).

7 RESULTS

We will demonstrate the use of BSH representation in shape design and reverse engineering. Our current implementation considers halfspaces represented either as basic primitives (e.g., planes, spheres, cylinders, cones, tori) or an VIPSS implicit function [Huang et al. 2019]. Our choice of VIPSS is motivated by its ability to interpolate sparse spatial locations (e.g., control points), which makes it ideal for shape control. However, our formulation and algorithms apply

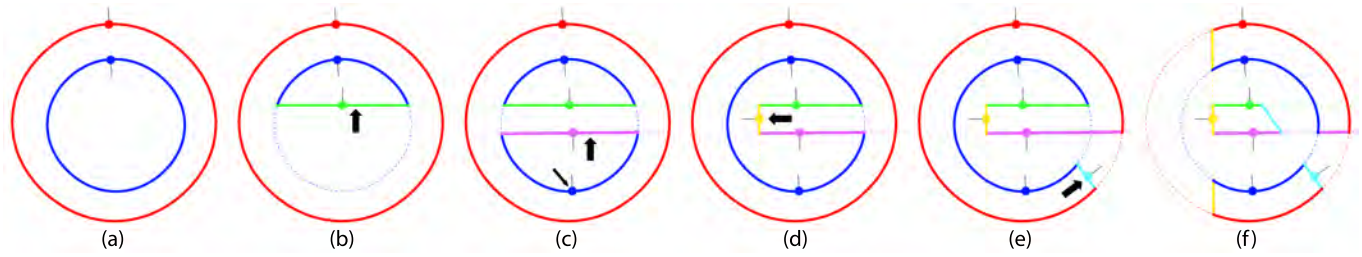


Fig. 13. (a-e): Edit sequence that modifies the letter "o" into the letter "e". Each step introduces a new (linear) halfspace with a sample (marked by thick arrows) and occasionally modifies samples on existing halfspaces (marked by thin arrows). (f): Result of the basic graph-cut algorithm (Section 5.1) without imposing sample-connectedness.

to other halfspace representations as well, such as closed meshes, parametric surfaces, and other types of implicit surfaces. To compute the arrangement of the halfspaces, we first tessellate the primitives and polygonize the implicit surfaces using Marching Cubes (we use a uniform grid of size 64^3), and we compute the mesh arrangement using the recent method of [Cherchi et al. 2020].

7.1 Shape design

The flexibility and expressiveness of BSH make it well suited for editing tasks that could have been difficult with either CSG or boundary representations. Consider the 2D example in Figure 13, which shows a sequence of edits (each adding one halfspace) from the letter "o" to the letter "e". Like CSG, BSH can naturally handle changes in the combinatorial structure of the boundary. Such changes would have to be carefully managed by the user if a boundary representation is used. On the other hand, the boundary samples in BSH offer more direct control than Boolean expressions in CSG. For example, the two linear halfspaces added in (d,e) only affect the parts of the shape where the samples are located. Although the same effects can be achieved using CSG, the Boolean expressions will become more complex and additional halfspaces might be needed. The controllability of BSH owes in large part to the *sample-connectedness* constraint. As seen in (f), the shape can look quite different without the constraint.

We show a few more 2D designs using BSH in Figure 14. Modeling these shapes with CSG would require complex Boolean expressions (even for the simple Star) and possibly additional halfspaces (e.g., where the shaft of the Musical Note joins the ellipse). These shapes can be represented easily and intuitively in BSH using only one sample on each segment of the boundary.

We demonstrate the flexibility of BSH in designing 3D shapes in Figure 15. First, the union of a torus and four spheres (a) is created by placing one sample on each sphere and four samples on the four segments of the torus. After deleting two samples from the torus, the corresponding segments of the torus are removed while the other segments remain connected to the spheres (b). This shape cannot be created using CSG without adding separating halfspaces that prevent the torus segments from going beyond the spheres. Using CSG, these additional halfspaces will need to be modified when the spheres "slide" along the torus (c), and more separating halfspaces are needed when new spheres are added to the torus (d).

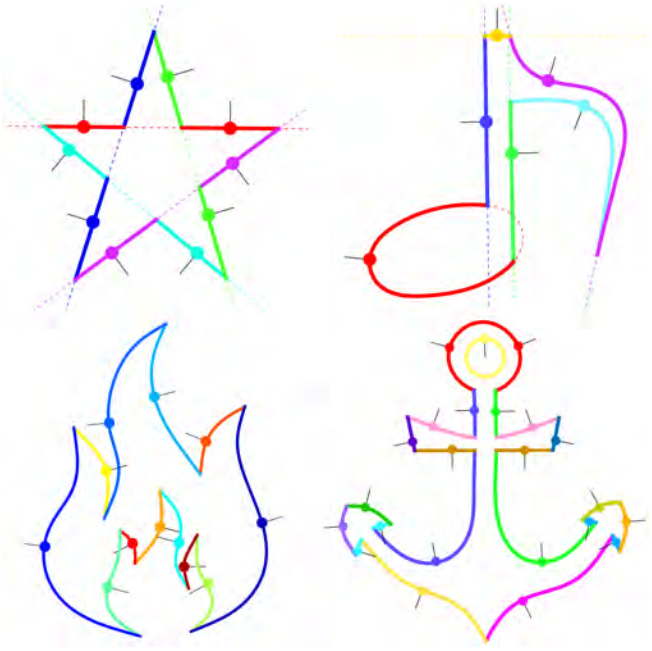


Fig. 14. Several 2D shapes modeled by BSH. Halfspaces for the last two shapes are not shown due to their complexity.

These edits can be achieved more easily with BSH by manipulating the boundary samples.

Several 3D shapes designed with BSH are shown in Figure 16. Although some of these shapes are quite simple (e.g., the first two shapes are made up of 3 to 4 primitive halfspaces), none of them can be described by CSG without non-trivial Boolean expressions or introducing additional halfspaces.

7.2 Reverse engineering

As discussed in Section 6.1, many methods exist to construct the halfspaces from a given input shape. We adopt the following protocol in all our experiments to compute halfspaces (except for Figure 19), although our sample-generation algorithm (Section 6.2) works with any set of input halfspaces. Given a mesh, we use the feature-aligned segmentation algorithm of [Zhuang et al. 2017] to partition the

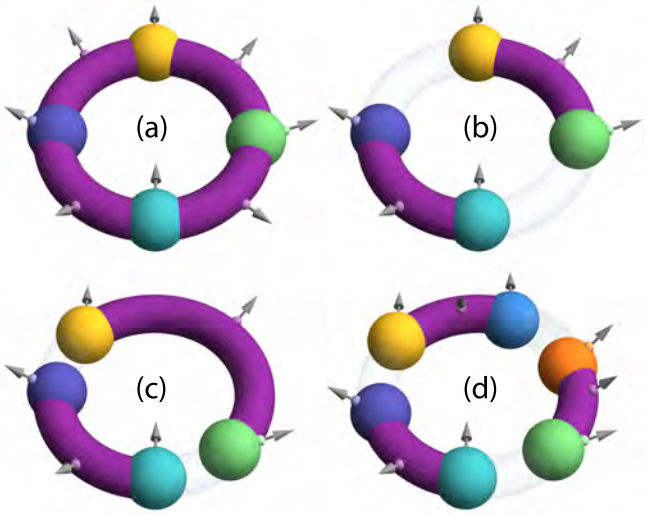


Fig. 15. Various BSH shapes created from one torus and several spheres. By choosing which segment of the torus has a sample, different segments can be kept or deleted while the shape remains a solid.

surface into smooth regions that meet along ridges and valleys. We fit one halfspace to each segment as follows. We first fit each of the five primitives using RANSAC [Schnabel et al. 2007] followed by least-square refinement [Lukács et al. 1998]. If the fitting error of one or multiple primitives is lower than a given tolerance ϵ, we select the least complex primitive according to this order of ascending complexity: plane, cylinder, cone, sphere, and torus. If the fitting errors of all primitives are greater than ϵ, we fit a VIPSS implicit surface [Huang et al. 2019] in a coarse-to-fine fashion as in [Carr et al. 2001]. Starting with three well-spaced points on the segment (i.e., the control points), we iteratively add the point with the greatest distance to the VIPSS surface interpolating the current set of control points as a new control point, until such distance is lower than ϵ.

The protocol above is controlled by the number of segments in the surface segmentation and the fitting tolerance ϵ. We demonstrate how these parameters affect the balance between fitting accuracy and simplicity of the representation in Figure 17. Observe that fewer segments (left column) lead to fewer halfspaces, but each halfspace tends to require a VIPSS with a large number of control points, due to the complex geometry of each segment. On the other hand, more segments (middle column) means that each segment has a simpler geometry, which can be fit with simpler primitives or VIPSS with fewer control points, at the cost of increased number of halfspaces. Finally, increasing the tolerance (right column) leads to even simpler primitives and VIPSS with even fewer control points, but the halfspaces deviate more from the input geometry. In our experiments, we set ϵ as $1/128$ of the diagonal length of the bounding box and choose the number of segments that is qualitatively comparable to the last two columns.

More examples of free-form shapes converted to BSH are given in Figures 1, 11 and 18. These shapes are fitted with a mixture of primitives and VIPSS implicits. The BSH representation allows each

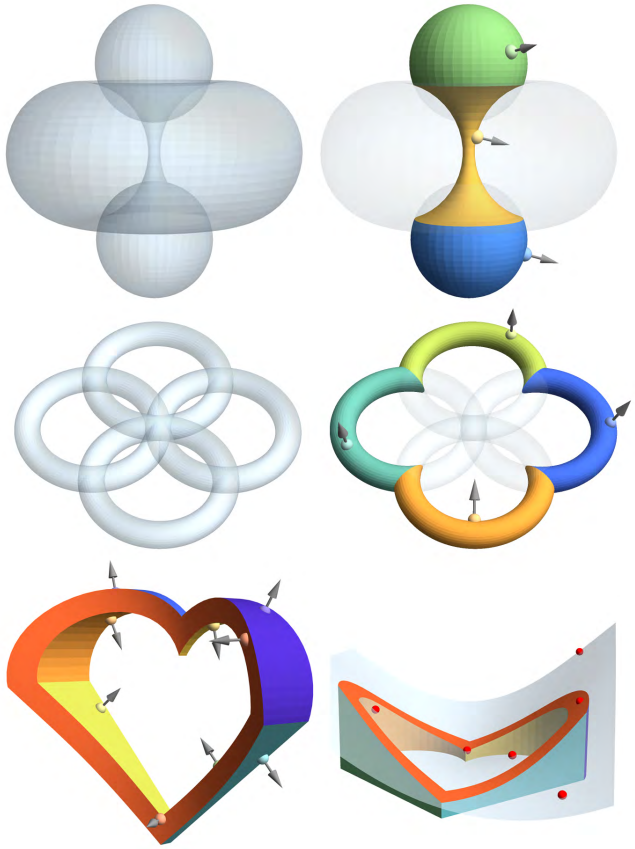


Fig. 16. Shapes modeled by BSH that cannot be represented by CSG without additional halfspaces. For the first two shapes, input halfspaces are on the left and the final shapes are on the right. The last shape ("Heart") is shown in two views, and the second view shows a halfspace represented as a VIPSS implicit surface interpolating a sparse set of control points (red spheres).

surface partition, and their composition, to be easily modified while maintaining the solidity of the result. We showcase a few interesting edits, such as replacing the handles of the Vase, turning the Elk into a bull, raising the pedals of the Flower, adjusting the seats on the Boat, and adding arms to the Chair. Please refer to the accompanying video for the editing process of some of these shapes.

Finally, we compare with reverse engineering of CSG shapes. Existing reverse CSG methods are limited to halfspaces represented as primitives, in part due to the difficulty in constructing separating halfspaces for higher-order surfaces [Shapiro and Vossler 1993]. Even in this limited setting, a key challenge for CSG reverse engineering is recovering Boolean expressions that are intuitive for downstream editing. As an example, the CAD model in Figure 19 (a) can be constructed using a fairly straight-forward sequence of Boolean operations: first intersect a cylinder with a rectangular prism, then subtract three rectangular prisms (to create the "notches"), and finally union with six tori. Taking the mesh as the input, the state-of-the-art InverseCSG method [Du et al. 2018] produces a Boolean expression tree as shown in Figure 20 (top). The partial shapes defined at three nodes of the tree (highlighted in the

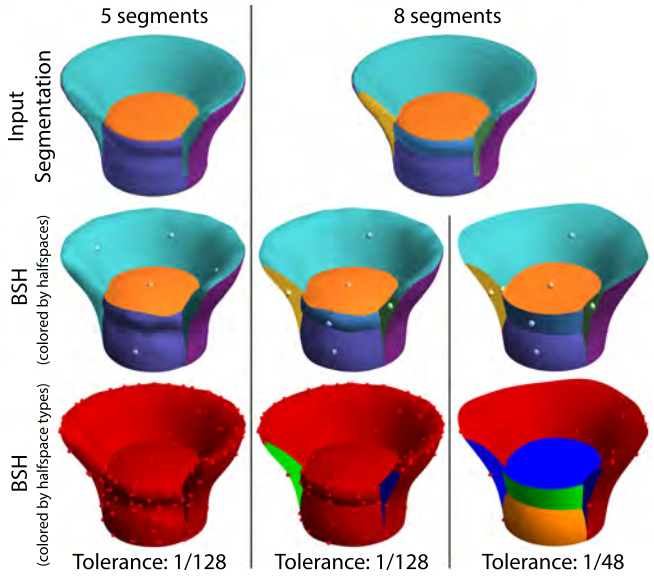


Fig. 17. Halfspaces (middle and bottom rows) constructed from an input shape (top row) with different number of segments and fitting tolerances (as fraction of the diagonal of the bounding box). Halfspaces in the bottom row are the same as those in the middle row but colored by their types (planes, cylinders, spheres, and VIPSS) with VIPSS control points shown as red spheres.

tree) are shown at the bottom. Observe that these Boolean operations are not only difficulty to understand but also unnecessarily complex. In contrast, starting from the same set of halfspaces as used by [Du et al. 2018], which consists of 17 primitives including planes, cylinders, and tori, our algorithm produces a BSH with 23 samples as shown in Figure 19 (b). The BSH shape can be conveniently edited in a semantic-aware manner, such as removing a "shelf" and a few tori (c) and changing the dimension of the primitives (d). These edits would be difficult to achieve with the CSG representation in Figure 20.

To evaluate the effectiveness of the sample generation algorithm (Section 6.2), we report in Table 1 the number of samples produced by our algorithm for each example. The table also shows the number of connected components of the same halfspace on the shape boundary as well as the total number of patches on the boundary, which are respectively the lower and upper bound of the number of samples needed to represent the shape (due to sample-connectedness and Proposition 4.1). Our greedy algorithm produces far fewer samples than the upper bound and achieves the lower bound in all by two examples.

8 LIMITATIONS AND FUTURE WORK

A key limitation of our work is that boundary extraction currently requires the complete arrangement of all input halfspaces. Even if the halfspaces are planes, the time complexity of arrangement computation is cubic to the number of planes [Agarwal and Sharir 2000]. In our experiments, computing mesh arrangement using the latest mesh arrangement algorithm [Cherchi et al. 2020] took over 5

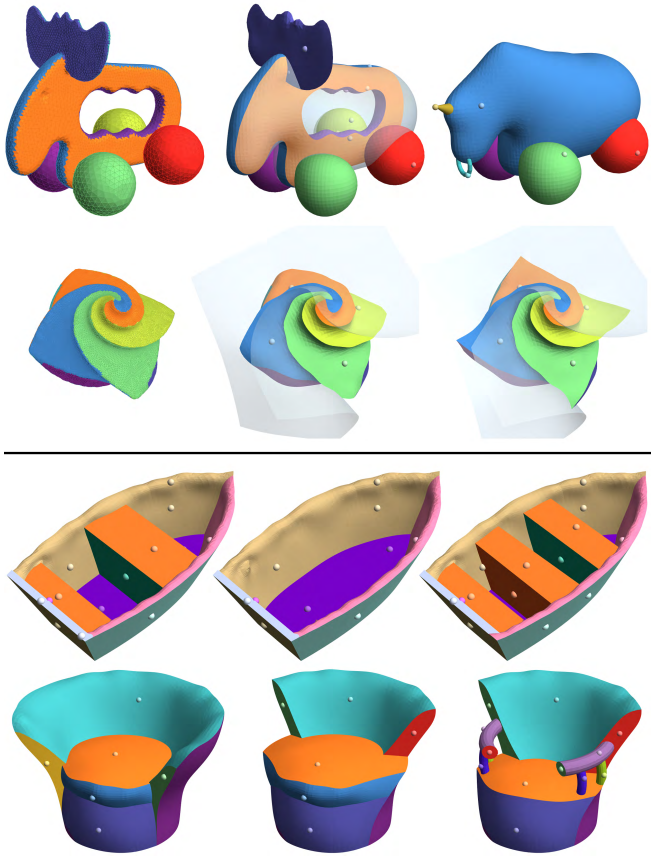


Fig. 18. Free-form BSH shapes (Elk, Flower, Boat, and Chair) converted from meshes and undergone editing of the halfspaces and/or their samples. Top: each row shows the input segmented mesh, the converted BSH, and result after editing. Selected halfspaces before and after editing are shown with transparency. Bottom: each row shows the converted BSH from Figure 11 or 17 and two editing results.

Table 1. Comparing the number of samples generated by our algorithm for each 3D reverse engineering example with the number of connected components of the same halfspace (a lower bound) and the total number of patches on the shape boundary (an upper bound). The Chair model is from the middle column of Figure 17, and CAD is from Figure 19.

	Vase	Boat	Chair	Elk	Flower	CAD
#samples	13	18	8	16	8	23
#components	13	15	8	10	8	23
#patches	45	89	60	46	10	116

seconds for 17 halfspaces (Figure 19), which is already too slow for interactive shape design. In contrast, boundary extraction on the arrangement takes almost negligible time. Ideally, we would like to be able to extract the boundary without computing the full arrangement. There are several promising ideas in this direction. We could compute the arrangement of only a portion of the input halfspace boundaries that are likely to contain the shape. This portion could be defined within some spatial range from the samples, as done for

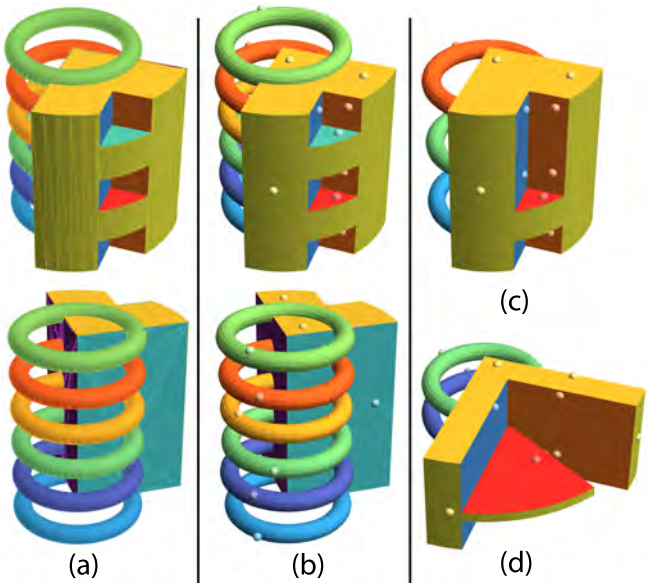


Fig. 19. A CAD mesh segmented and fitted by primitives (a; showing two views), the converted BSH shape (b; showing two views), and two edited shapes with altered structure (e.g., fewer rings and a missing shelf) (c) and modified primitive geometry (d).

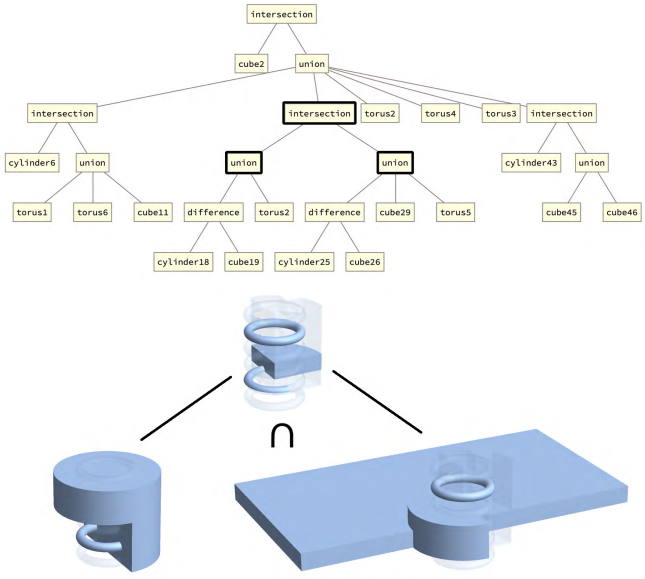


Fig. 20. Top: The CSG tree converted from the mesh in Figure 19(a) using InverseCSG [Du et al. 2018]. Bottom: Partial shapes evaluated at three highlighted nodes of the tree (the entire shape is shown as a transparent background).

linear halfspaces [Boulch et al. 2014; Chauve et al. 2010; Oesau et al. 2014]), or using a kinetic structure similar to [Bauchet and Lafarge 2020]. An alternative, divide-and-conquer idea is to group halfspaces that define nearby parts of the shape and compute the BSH shape for each group (which requires a much smaller arrangement computation) before computing the entire BSH shape, treating each group-wise BSH shape as a single halfspace. Lastly, user interaction at the design phase can also help with defining the relevant portion on, or the grouping of, the halfspaces.

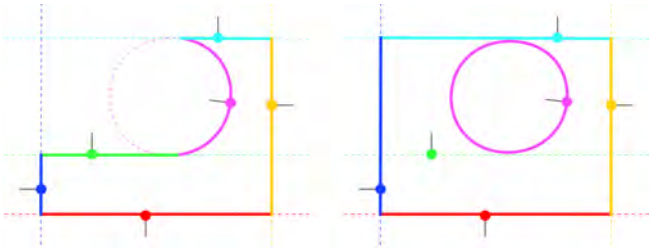


Fig. 21. Sensitivity of BSH to halfspaces that meet almost tangentially (e.g., the purple circle meeting the green and cyan lines).

While BSH is well suited for representing smooth surfaces meeting at non-smooth junctions, it may have challenges in representing surfaces that meet almost *tangentially* (i.e., they share similar tangent planes). This is because the intersection of their underlying halfspaces is likely missing on the arrangement, either due to numerical inaccuracy or discretization. In the 2D example on the left of Figure 21, the purple circular arc meets almost tangentially with the green and cyan line segments. Slightly shrinking the purple circle, however, causes it to miss the intersection with both lines and results in an undesirable shape shown on the right. Robust means for handling tangential connections would make BSH more appealing for CAD objects, which often contain such connections (e.g., fillets).

As a new shape representation, BSH has the potential to improve existing geometric processing pipelines. For example, if BSH can be directly created from an input cloud without first reconstructing a surface, it could lead to savings in both time and storage. In particular, existing adaptive surface reconstruction schemes tend to produce refined meshes near sharp features, which can be avoided by BSH, because the sharp features are implicitly defined by intersecting smooth halfspaces. It would also be interesting to explore BSH as a representation for geometric learning tasks. BSH possesses several desirable properties for learning. First, it is much more compact than the raw point cloud or mesh. Second, it encodes semantics of the shape, since each halfspace often captures a semantic component. Third, without a hierarchical structure or boundary connectivity information, BSH is more agile than both CSG and B-reps (e.g., NURBS) as a part-based representation.

ACKNOWLEDGMENTS

This work is supported by NSF grant RI-1618685 and a gift from Adobe Systems. We would like to thank anonymous reviewers for their valuable suggestions.

REFERENCES

Pankaj K Agarwal and Micha Sharir. 2000. Arrangements and their applications. In *Handbook of computational geometry*. Elsevier, 49–119.

Lionel Alberti, Bernard Mourrain, and Julien Wintz. 2008. Topology and arrangement computation of semi-algebraic planar curves. *Computer Aided Geometric Design* 25,

8 (2008), 631–651.
Jean-Philippe Bauchet and Florent Lafarge. 2020. Kinetic shape reconstruction. *ACM Transactions on Graphics (TOG)* 39, 5 (2020), 1–14.
Eric Berberich, Pavel Emeliyanenko, Alexander Kobel, and Michael Sagraloff. 2012. Arrangement computation for planar algebraic curves. In *Proceedings of the 2011 International Workshop on Symbolic-Numeric Computation*. 88–98.
Matthew Berger, Andrea Tagliasacchi, Lee M Seversky, Pierre Alliez, Gael Guennebaud, Joshua A Levine, Andrei Sharf, and Claudio T Silva. 2017. A survey of surface reconstruction from point clouds. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 301–329.
Gilbert Bernstein and Don Fussell. 2009. Fast, exact, linear booleans. In *Computer Graphics Forum*, Vol. 28. Wiley Online Library, 1269–1278.
Alexandre Boulch, Martin de La Gorce, and Renaud Marlet. 2014. Piecewise-planar 3D reconstruction with edge and corner regularization. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 55–64.
Suzanne F Buchele and Richard H Crawford. 2003. Three-dimensional halfspace constructive solid geometry tree construction from implicit boundary representations. 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 *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2001)*. 67–76.
Malcolm S Casale and James E Bobrow. 1989. A set operation algorithm for sculptured solids modeled with trimmed patches. *Computer Aided Geometric Design* 6, 3 (1989), 235–247.
Anne-Laure Chauve, Patrick Labatut, and Jean-Philippe Pons. 2010. Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 1261–1268.
Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. 2020. Bsp-net: Generating compact meshes via binary space partitioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 45–54.
Gianmarco Cherchi, Marco Livesu, Riccardo Scateni, and Marco Attene. 2020. Fast and robust mesh arrangements using floating-point arithmetic. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–16.
Matthijs Douze, Jean-Sébastien Franco, and Bruno Raffin. 2017. QuickCSG: Fast Arbitrary Boolean Combinations of N Solids. *arXiv preprint arXiv:1706.01558* (2017).
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.
Laurent Dupont, Michael Hemmer, Sylvain Petitjean, and Elmar Sch"omer. 2007. Complete, exact and efficient implementation for computing the adjacency graph of an arrangement of quadrics. In *European Symposium on Algorithms*. Springer, 633–644.
Jack Edmonds and Richard M Karp. 1972. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)* 19, 2 (1972), 248–264.
Pierre-Alain Fayolle and Alexander Pasko. 2016. An evolutionary approach to the extraction of object construction trees from 3D point clouds. *Computer-Aided Design* 74 (2016), 1–17.
Pierre-Alain Fayolle, Alexander Pasko, Elena Kartasheva, Christophe Rosenberger, and Christian Toinard. 2008. Automation of the volumetric models construction. In *Heterogeneous objects modelling and applications*. Springer, 214–238.
Francisco R Feito, Carlos J Ogáyar, Rafael Jesús Segura, and ML Rivero. 2013. Fast and accurate evaluation of regularized Boolean operations on triangulated solids. *Computer-Aided Design* 45, 3 (2013), 705–716.
Karim Hamza and Kazuhiro Saitou. 2004. Optimization of constructive solid geometry via a tree-based multi-objective genetic algorithm. In *Genetic and Evolutionary Computation Conference*. Springer, 981–992.
Christoph Martin Hoffmann. 1989. Geometric and solid modeling. (1989).
Zhiyang Huang, Nathan Carr, and Tao Ju. 2019. Variational implicit point set surfaces. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–13.
Adrien Kaiser, Jose Alonso Ybanez Zepeda, and Tamy Boubekeur. 2019. A survey of simple geometric primitives detection methods for captured 3D data. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 167–196.
John Keyser, Tim Culver, Mark Foskey, Shankar Krishnan, and Dinesh Manocha. 2004. ESOLID - a system for exact boundary evaluation. *Computer-Aided Design* 36, 2 (2004), 175–193.
Pierre-Alain Langlois, Alexandre Boulch, and Renaud Marlet. 2019. Surface reconstruction from 3d line segments. In *2019 International Conference on 3D Vision (3DV)*. IEEE, 553–563.
Jyh-Ming Lien, Vikram Sharma, Gert Vegter, and Chee Yap. 2014. Isotopic arrangement of simple curves: An exact numerical approach based on subdivision. In *International Congress on Mathematical Software*. Springer, 277–282.
Gabor Lukács, Ralph Martin, and Dave Marshall. 1998. Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation. In *European conference on computer vision*. Springer, 671–686.
Bernard Mourrain, Jean-Pierre Técourt, and Monique Teillaud. 2005. On the computation of an arrangement of quadrics in 3d. *Computational Geometry* 30, 2 (2005), 145–164.
Gregory M Nielson. 2004. Radial hermite operators for scattered point cloud data with normal vectors and applications to implicitizing polygon mesh surfaces for generalized CSG operations and smoothing. In *IEEE Visualization 2004*. IEEE, 203–210.
Sven Oesau, Florent Lafarge, and Pierre Alliez. 2014. Indoor scene reconstruction using feature sensitive primitive extraction and graph-cut. *ISPRS journal of photogrammetry and remote sensing* 90 (2014), 68–82.
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.
Darko Pavić, Marcel Campen, and Leif Kobbelt. 2010. Hybrid booleans. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 75–87.
Aristides AG Requicha and Herbert B Voelcker. 1977. Constructive solid geometry. (1977).
Aristides G Requicha. 1980. Representations for rigid solids: Theory, methods, and systems. *ACM Computing Surveys (CSUR)* 12, 4 (1980), 437–464.
Jaroslaw Rossignac and Aristides Requicha. 1984. Constant-radius blending in solid modelling. (1984).
Jaroslaw R Rossignac and Aristides AG Requicha. 1986. Offsetting operations in solid modelling. *Computer Aided Geometric Design* 3, 2 (1986), 129–148.
Jaroslaw R Rossignac and Aristides AG Requicha. 1999. *Solid modeling*. Technical Report. Georgia Institute of Technology.
Ruwen Schnabel, Roland Wahl, and Reinhard Klein. 2007. Efficient RANSAC for point-cloud shape detection. In *Computer graphics forum*, Vol. 26. Wiley Online Library, 214–226.
Elmar Schömer and Nicola Wolpert. 2006. An exact and efficient approach for computing a cell in an arrangement of quadrics. *Computational Geometry* 33, 1-2 (2006), 65–97.
Ariel Shamir. 2008. A survey on mesh segmentation techniques. In *Computer graphics forum*, Vol. 27. Wiley Online Library, 1539–1556.
Vadim Shapiro. 2002. Solid Modeling. *Handbook of computer aided geometric design* 20 (2002), 473–518.
Vadim Shapiro and Donald L Vossler. 1991. Construction and optimization of CSG representations. *Computer-Aided Design* 23, 11 (1991), 4–20.
Vadim Shapiro and Donald L Vossler. 1993. Separation for boundary to CSG conversion. *ACM Transactions on Graphics (TOG)* 12, 1 (1993), 35–55.
Sara Silva, Pierre-Alain Fayolle, Johann Vincent, Guillaume Pauron, Christophe Rosenberger, and Christian Toinard. 2005. Evolutionary computation approaches for shape modelling and fitting. In *Portuguese Conference on Artificial Intelligence*. Springer, 144–155.
JM Smith and Neil A Dodgson. 2007. A topologically robust algorithm for Boolean operations on polyhedral shapes using approximate arithmetic. *Computer-Aided Design* 39, 2 (2007), 149–163.
Yannick Verdie, Florent Lafarge, and Pierre Alliez. 2015. LOD generation for urban scenes. *ACM Transactions on Graphics* 34, ARTICLE (2015), 30.
Charlie CL Wang. 2010. Approximate boolean operations on large polyhedral solids with partial mesh reconstruction. *IEEE transactions on visualization and computer graphics* 17, 6 (2010), 836–849.
Qiaoyun Wu, Kai Xu, and Jun Wang. 2018. Constructing 3D CSG models from 3D raw point clouds. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 221–232.
Brian Wyvill, Andrew Guy, and Eric Galin. 1999. Extending the csg tree. warping, blending and boolean operations in an implicit surface modeling system. In *Computer Graphics Forum*, Vol. 18. Wiley Online Library, 149–158.
Songgang Xu and John Keyser. 2013. Fast and robust Booleans on polyhedra. *Computer-Aided Design* 45, 2 (2013), 529–534.
Qingnan Zhou, Eitan Grinspun, Denis Zorin, and Alec Jacobson. 2016. Mesh arrangements for solid geometry. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–15.
Yixin Zhuang, Hang Dou, Nathan Carr, and Tao Ju. 2017. Feature-aligned segmentation using correlation clustering. In *Computational Visual Media*, Vol. 2. 147–160.

A COMPLETENESS OF THE BOUNDARY EXTRACTION ALGORITHM

We give the proof of Proposition 5.2:

Proof. We can organize all states explored by the algorithm in a tree structure (the search tree), in which a child state is created by adding a removable set to its parent state. The degree of the tree is bounded by the number of removable sets on any given boundary,

and the depth of the tree is bounded by the total number of patches in $\cup_i b_i$. Hence the tree has a finite size, and the algorithm terminates in finite steps.

Since the algorithm always produces an orientation-preserving and sample-connected solution, we only need to show that the solution minimizes the energy E. In the search tree, the energy at any child state is no smaller than that at its parent state, because the graph-cut algorithm at the child state is applied to a graph with increased edge weights from the graph at the parent state (more edges have infinite weight). Since the algorithm performs a best-first traversal of the search tree, the optimality of the algorithm is guaranteed if the goal state (the BSH boundary, denoted by b^*) exists in the search tree. To show the latter, we construct a sequence of states $\{\{r^0, b^0\}, \{r^1, b^1\}, \ldots\}$ such that $r^0 = \emptyset$ and r^{i+1} expands r^i by a removable set on b^i that does not lie on b^*. By the definition of removable sets and Proposition 5.1, the construction can always proceed until reaching some state $\{r^k, b^k\}$ where b^k is sample-connected. Since b^k has the minimal energy among all halfspace-bounding, orientation-preserving and sample-connected boundaries that do not contain r^k, and since b^* is one such boundary, either $b^k = b^*$ or b^k has the same energy as b^* (which only happens for non-generic geometry; see Section 4.2). Since this sequence of states is part of the search tree, the goal state (or a state with the same energy) exists in the tree, which concludes the proof. □

B TERMINATION OF THE SAMPLE GENERATION ALGORITHM

We give the proof of Proposition 6.1:

Proof. We only need to show that, if the BSH shape boundary b' computed from the current sample set S at each iteration differs from b, than there exists some $p \in b \setminus b'$ that does not contain a sample. If this is true, and since each iteration of the algorithm adds one sample to a such a patch, the number of iterations is bounded by the total number of patches on b. Upon termination, the sample set reproduces the BSH shape boundary by construction. To show the statement above, note that b' minimizes the energy E among all halfspace-bounding, orientation-preserving, and sample-connected boundaries that are subsets of $\cup_i b_i$, and one of such boundaries is the target b. Hence $E(b') \leq E(b)$. Since b contains all the current samples S, b' has to contains all samples in S as well, otherwise it will have a higher energy than b. If $b \neq b'$, and since b cannot be a strict subset of b' (otherwise b' will have a higher energy), there must be patches in b that are missing in b', and these patches have to be free of samples (which are already included in b'). □

We next show that the algorithm still terminates when the complete search for extracting the BSH boundary is replaced by a beam search with width $K = 1$:

Proposition B.1. *The algorithm GENERATE-SAMPLE in Figure 12 terminates in finite steps if the EXTRACT-BOUNDARY algorithm is replaced with a beam search of width $K = 1$.*

Proof. Similar to the proof of Proposition 6.1, it suffices to show that there exists some $p \in b \setminus b'$ that does not contain a sample if $b \neq b'$. Consider the beam search algorithm for extracting the boundary b' given the current set of samples S. Since the queue has size 1, the search iteratively retrieves a state from the queue, explores all its child states, and places the child state with the least boundary energy back to the queue. Let the sequence of states that are placed into the queue during the search be $\{\{r^1, b^1\}, \ldots, \{r^k, b^k\}\}$, where r^i is the set of patches marked for removal and b^i is the resulting boundary computed by graph-cut.

We will show that the following is true of r^i for all $i = 1, \ldots, k$: either $r^i \cap b = \emptyset$ (called Type I), or there is some patch $p \in r^i \cap b$ such that p contains no sample in S (called Type II). We prove by induction. For $i = 1$, since r^1 is empty, it is trivially Type I. Now suppose the statement holds up to some $i < k$. The next state, $\{r^{i+1}, b^{i+1}\}$ is constructed by adding some removable set on b^i to r^i, such that the resulting boundary computed by graph-cut without the removed patches has the least energy among all removable sets of b^i. If r^i is Type I, and since b is sample-connected, Proposition 5.1 ensures that there exists some removable set t of b^i such that $t \cap b = \emptyset$. Since $r^i \cap b = \emptyset$, $(t \cup r^i) \cap b = \emptyset$, and hence performing graph-cut after removing patches $t \cup r^i$ would result in a boundary whose energy is no greater than $E(b)$. Hence we conclude that $E(b^{i+1}) \leq E(b)$, and therefore b^{i+1} must contain all samples in S. This implies that r^{i+1} must not contain any sample, and hence it will fall into either Type I or Type II. On the other hand, if r^i is Type II, and since $r^i \subset r^{i+1}$, r^{i+1} is Type II as well.

Finally, note that b^k is the boundary b' returned by the beam search. If r^k is Type I, and since b (like b^k) does not contain any patch in r^k, we know $E(b^k) \leq E(b)$, and the argument in the proof of Proposition 6.1 shows that there is a sample-free patch $p \in b \setminus b^k$ if $b \neq b^k$. If r^k is Type II, the patch $p \in r^k \cap b$ that does not contain any sample is the one that we seek, since it lies on b but not b^k. □