Geometric Modeling of Multi-Material Printed Objects

Tyson Brochu[1] and Ryan Schmidt[2]

[1]Autodesk Research
[2]gradientspace, inc.

Abstract

We introduce a set of tools for interactive modeling of multi-material objects. We use non-manifold surface meshes to define complex objects, which can have multiple connected solid regions of different materials. Our suite of tools can create and edit non-manifold surfaces, while maintaining a consistent labeling of distinct regions. We also introduce a technique for generating approximate material gradients, using a set of thin layers with varying material properties. We demonstrate our approaches by printing physical objects with a multi-material printer.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Modeling—Modeling Interfaces

1. Introduction

Multi-material 3D printing promises to be a significant leap forward in rapid prototyping, manufacturing, and customized design optimization. Being able to combine multiple functional materials into a single print has the potential to greatly increase the utility of printed objects. However, current 3D print pipelines require each discrete material region to be a separate 3D object, leading to complex, inflexible assemblies that are difficult to manipulate.

We present new techniques which make the modeling and fabrication of multi-material objects easier and more intuitive. Leveraging recent work on multi-material surface tracking, we are able to use surface meshes to define different material regions, avoiding the use of volumetric grids or meshes. Our system can extrapolate material regions from marked-up solid surfaces, use mesh intersections to define new regions, and generate approximate gradients using thin regions of discrete material properties.

We introduce the following contributions:

- A user interface for creating and manipulating non-manifold surfaces;
- A fast and robust non-regularized mesh Boolean operation;
- A method for generating discrete layers in a solid object to approximate a gradient;

2. Related Work

We are heavily inspired by the work of Da et al. [DBG14]. They describe a multi-material surface tracking method targeting the simulation of immiscible fluids for animation. We adopt their approach for defining and tracking regions of different materials as the mesh undergoes refinement and other editing operations.

Figure 1: *Left: exploded view of a non-manifold 3D model created with our system. Right: physical object fabricated with a multi-material 3D printer. White regions are hard solid plastic, and black regions are a flexible, rubber-like material.*

Schmidt and Brochu recently introduced an approach for computing Boolean operations on surface mesh objects [SB16]. We extend some of their core ideas to perform non-regularized Boolean operations on surface meshes. This allows an additional, convenient way to create non-manifold surface meshes suitable for multi-material objects.

3. Surface Complexes

Key to our implementation is the use of a *complex* surface type. Sometimes called *cellular topology*, this type of object is comprised of two or more solid regions bounded by a single non-manifold surface. Adjacent solid regions are separated by a single patch of triangles (in this work we will refer to these surface

DOI: 10.2312/egsh.20171011

patches as *interior surfaces*.) The advantage to treating discrete regions as one object is that modeling operations do not need to ensure that separate surface patches are precisely aligned at all times. If separate solid models were used, it would be possible to introduce overlaps or air gaps between surfaces as they deform or are remeshed.

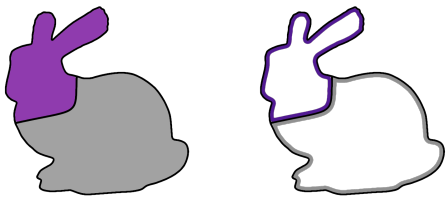

Figure 2: *Left: bunny-shaped volume partitioned into two regions. Right: achieving the same partitioning by tagging surfaces with front and back region identifiers.*

Like Da et al. [DBG14], we use a non-manifold triangle mesh data structure with additional data associated with each triangle. Each triangle is given two integer identifiers, corresponding to the front and back of the triangle (as defined by the triangle winding order.) Triangles on the exterior surface of the object have a reserved integer value representing the space entirely outside of the object. See Figure 2 for a 2D analog. For more details, see the paper by Da et al. [DBG14].

Figure 3: *Two methods for creating non-manifold surface meshes from sets of selected faces. Center: offset the selected faces into the object. Right: fill the loop of boundary edges between two different triangle groups.*

Our interactive interface allows the user to create non-manifold surface meshes in a number of different ways. To aid in this, we allow the user to select individual triangles and tag them with *group IDs*. The system can then construct complex objects from group IDs in two ways: by offsetting a group of triangles into the object to a user-defined depth, or by finding the boundary loop around a group of triangles, and filling this loop with a new mesh (see Figure 3). We have used these two simple techniques to model and print several objects on a Stratasys Objet Connex 260 printer – one example is shown in figure 1.

We have implemented these techniques inside an existing, freely-available mesh editing package. This package also allows for interactive remeshing, sculpting, and other surface editing techniques. We have extended many of these techniques to handle non-manifold meshes, so that the user can continue to edit objects after they have been converted from single-solid to complex objects.

In many cases, enabling remeshing of non-manifold surfaces simply involved changing the underlying assumption that only one or two triangles could be incident on a given edge, and then modifying the mesh-traversal operations. In some cases (for example, the Extrude tool), we re-wrote the tools so that they operated only on *exterior* triangles. In effect, we detach the mesh into an exterior surface and one or more interior surfaces. We then perform the operation as it was originally written on the exterior surface, then reattach the surfaces using a mesh zippering algorithm. To make the mesh zippering feasible, we track changes made to the previously non-manifold edges and repeat them on the interior surfaces.

For example, suppose edge (A,B) is non-manifold. We detach the exterior surface and duplicate this edge, assigning original edge (A,B) to the exterior (now manifold) surface, and creating a new edge (A',B') on the interior surface. Then if we split edge (A,B) on the exterior surface during the mesh operation, for example, we would subsequently find and split edge (A',B') on the interior surface. This ensures a one-to-one matching of vertices when we reattach the interior surface.

4. Non-Regularized Boolean

The methods discussed up to this point allow users to quickly mark up existing solid objects in order to create multi-material objects. Additionally, we also support a more general method for creating non-manifold surfaces: non-regularized Booleans.

Many readers will be familiar with Boolean operations on volumes, e.g. union, intersection, or difference. In order to compute these operations using surface representations many existing libraries will first create "non-regularized" Booleans, which we define as the union of the input surfaces. The idea is to first find the contours of intersection between two surfaces, then join the two surfaces at these intersection contours. (When done using polyhedral meshes, this operation necessarily creates non-manifold meshes.) After this joining is performed, we determine a set of triangles to discard, depending on which Boolean operation we are performing. For example, to compute a Boolean union operation, we discard faces that lie inside the final model (see figure 4).

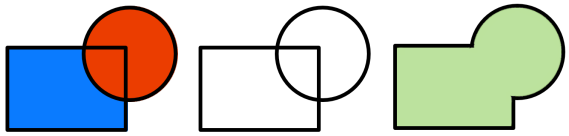

Figure 4: *A standard mesh Boolean operation. Left: two volumes bounded by surfaces are given as input. Center: a non-regularized Boolean (the union of input surfaces) is computed, resulting in a non-manifold surface. Right: interior surface patches are discarded, resulting in a volume that is the union of the two original volumes.*

In our system, we stop the process before discarding triangles to complete the Boolean operation. In other words, we find the intersection contours, and perform the necessary mesh surgery to join

the two input surface at these contours. This leaves us with a non-manifold surface that bounds multiple discrete regions, the same class of surface we are interested in for multi-material modeling (see figure 5).

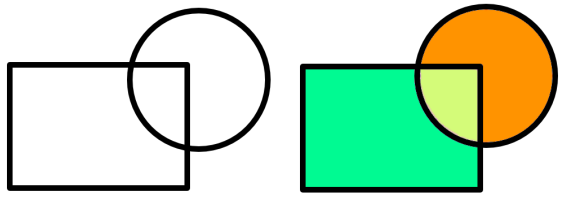

Figure 5: *A non-regularized Boolean is computed as before. If we support non-manifold meshes, we can now use this surface as a partitioning of the original two volumes into three distinct regions.*

To compute the non-regularized Boolean, we follow and extend the ideas of Schmidt and Brochu [SB16]. These authors have developed an *adaptive mesh* approach to computing mesh Booleans. Rather than relying on exact geometric predicates for intersection testing and mesh surgery, they instead rely on conservative intersection testing, remeshing, and dynamic mesh evolution to robustly compute Boolean operations. Their process for performing a full Boolean operation is as follows:

1. Detect intersection contours of two surfaces
2. Remove intersecting triangles from each surface
3. Discard triangles depending on which Boolean operation we are performing. We are left with a maximum of two boundaries corresponding to each intersection contour.
4. Perform an advancing-front style mesh evolution to move each pair of boundaries towards each other and towards the intersection contour. Simultaneously, perform on-the-fly remeshing to attempt to match the resolution of each boundary loop.
5. When the distances between boundaries are small enough, attempt to zipper vertices on corresponding boundaries (the authors use a voting scheme to determine which pairs of vertices should be snapped together.)

In our case, since we are not discarding mesh patches we must join more than two boundaries at once. Our modified version of the algorithm proceeds as follows:

1. Detect intersection contours of two surfaces
2. Remove intersecting triangles from each surface. We are left with a set of boundaries corresponding to each intersection contour (possibly more than two boundaries per contour).
3. Perform an advancing-front style mesh evolution to move each set of boundaries towards the other boundaries as well as the intersection contour.
4. When the distances between boundaries are small enough, we run a gap-closing algorithm [BNK02] which will refine boundary edges to ensure we have a one-to-one vertex correspondence. This is run pair-wise on all boundaries until we achieve a consistent set of vertex matches. Zippering the boundaries together is then trivial.

The main challenge of our approach is in evolving multiple boundaries towards each other simultaneously, and in matching the final boundaries so that they can be easily stitched together by merging vertices.

For the final step, we have found that pair-wise iterations of gap-closing followed by refining arbitrary edges until the number of vertices in each boundary is equal to be very effective. Modifying the advancing-front mesh evolution of Schmidt and Brochu [SB16] from handling a pair of boundaries to an arbitrary set of boundaries was fairly simple. The resulting algorithm is much easier to implement than algorithms that depend on robust intersection testing and construction [CGA15, Ber15, BGF15]. Our approach does not require geometrically exact predicates (conservative bounds are sufficient), or complicated combinatorial operations.

If both input models are closed surfaces, we can generate consistent front and back group IDs automatically, by determining which triangles from model A are inside model B, and vice-versa. This is useful for subsequently selecting and deleting surface patches, as well as for splitting the model up in order to send it to the 3D printer.

Figures 6 and 7 show results of our non-regularized Boolean. To generate the model in figure 6, we ran the non-regularized Boolean algorithm on three knot models, resulting in a complicated collection of distinct volume regions. In figure 7 we modeled a soft grip on the handle of a plastic comb using open sheets. We first thickened the sheets to create volumes, then ran the non-regularized Boolean operation on these volumes. Deleting the appropriate exterior mesh components results in a complex model which could be printed with a multi-material printer.

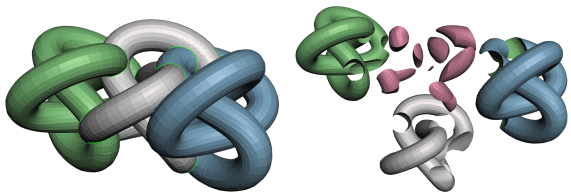

Figure 6: *Left: a complex object created by performing the non-regularized Boolean of three knot models. Right: exploded view showing four sub-regions.*

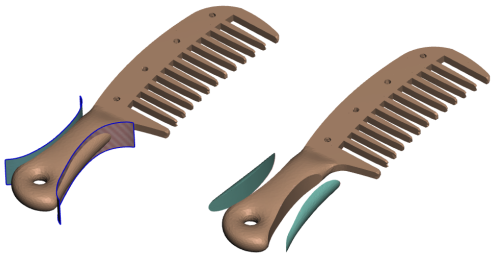

Figure 7: *Creating a complex by cutting a model with sheets. Left: input comb and open surfaces. Right: result of thickening the open surfaces into volumes, performing a non-regularized Boolean operation, and discarding the exterior of the thickened surface (exploded view).*

5. Discrete Gradients

Part of the promise of multi-material 3D printing is that it will enable users to specify arbitrary materials at any point in space. This in turn would allow the creation of new objects which would otherwise be difficult to create. For example, we should be able to achieve a gradient of materials for an object: starting with one colour or set of physical properties and slowly interpolating to another. Unfortunately, existing multi-material 3D printers either do not possess or do not expose this functionality. We present a solution that makes use of our non-manifold modeling paradigm.

To achieve this, we need to approximate the gradient with a set of discrete regions with different material properties. Our approach is to iteratively mesh selected isocontours of a pseudo-distance field going from one surface region to another. We perform a non-regularized Boolean operation on the original model and each meshed isocontour, discard triangles that are outside of the original model, and assign material properties to the regions between isocontours.

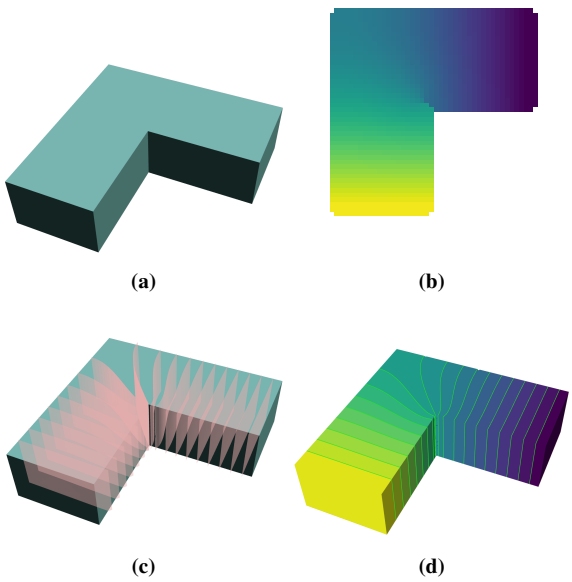

Figure 8: *(a) Input object. (b) One slice of the numerically-solved harmonic interpolation function. (c) Isocontours extracted from the harmonic function. (d) A complex created by merging isocontour meshes with the original object. Varying materials are assigned to each solid region.*

In our interface, we ask the user to mark two contiguous groups of triangles on the input model, then use a regular grid to solve for a harmonic function with boundary conditions specified at these triangle groups. For example, if the user creates triangle groups j and k, we solve for a function ϕ such that $\nabla^2\phi = 0$ on the interior of the model, $\phi = 1$ at triangles in group j, $\phi = 0$ at triangles in group k, and the normal derivative, $\nabla\phi \cdot \vec{n}$, is zero elsewhere. This produces a pseudo-distance function from group j to k (i.e. ϕ is smooth and non-negative on the interior), whose isocontours we can mesh using dual contouring [JLSW02], then attach to the original model as described above. Figure 8 shows an example. In our implementation, we solved for ϕ using finite differences on a regular grid, however other approaches such as the Boundary Element Method could also be employed.

Note that we could use any method to generate the interior mesh slices. For example, we could insert copies of a planar mesh patch at regular intervals along a particular axis in order to achieve a directional gradient, or we could mesh an implicit sphere of varying radius to achieve a radial gradient.

6. Conclusions and Future Work

We have presented a new approach to geometric modeling using triangle surface meshes, specifically targeting multi-material fabrication with state-of-the art 3D printers. Our set of tools allows users to easily create and manipulate surface complexes in order to produce multi-material print jobs.

We see several avenues for future improvement, including:

- Extend our non-regularized Boolean operation into a full-fledged mesh Boolean tool by discarding sets of triangles after computing. The advantage of this over the method of Schmidt and Brochu [SB16] is that there is less ambiguity in which set of triangles to discard *after* the non-regularized Boolean is performed.
- Eliminate the requirement for closed surfaces as inputs to the non-regularized Boolean operation. (This requirement is only essential for performing inside/outside testing, not for actually merging the open boundaries together.) We should then be able to model the object in figure 7 without first thickening the surfaces used for cutting.

Acknowledgements

We thank the anonymous reviewers for their feedback, Robert Bridson for providing the harmonic solver implementation, and Nobuyuki Umetani for suggestions on the written submission.

References

[Ber15] BERNSTEIN G.: Cork Boolean library. https://github.com/gilbo/cork, 2008–2015. 3

[BGF15] BARKI H., GUENNEBAUD G., FOUFOU S.: Exact, robust, and efficient regularized Booleans on general 3D meshes. *Computers & Mathematics with Aplpications 70* (2015), 1235–1254. 3

[BNK02] BORODIN P., NOVOTNI M., KLEIN R.: Progressive gap closing for meshrepairing. In *Advances in Modelling, Animation and Rendering*. Springer, 2002, pp. 201–213. 3

[CGA15] CGAL: *CGAL User and Reference Manual*, 4.6.3 ed. CGAL Editorial Board, 2015. URL: http://doc.cgal.org/4.6.3/Manual/packages.html. 3

[DBG14] DA F., BATTY C., GRINSPUN E.: Multimaterial mesh-based surface tracking. *ACM Trans. Graph. 33*, 4 (July 2014), 112:1–112:11. 1, 2

[JLSW02] JU T., LOSASSO F., SCHAEFER S., WARREN J.: Dual contouring of hermite data. *ACM Trans. Graph. 21*, 3 (July 2002), 339–346. 4

[SB16] SCHMIDT R., BROCHU T.: Adaptive mesh booleans. http://arxiv.org/abs/1605.01760, 2016. 1, 3, 4