

A Topology-Preserving Voxelization Shrinking Algorithm

Daniel Whalen
 Dept. of Physics • Stanford University
 382 Via Pueblo Mall • Stanford, CA 94305 • USA
 dwhalen@stanford.edu

Abstract

Recognition of 3-dimensional objects with complex internal geometries relies more heavily on topology than external shape. We discuss a novel voxelization rescaling algorithm intended to preserve topological structure even at low resolutions, appropriate for artistic reproductions of complex geometries.

Introduction: Voxels and Motivations

Voxels are the 3-dimensional analog of pixels; a filled voxel represents a cube in space. Techniques for acquiring voxel representations of 3-dimensional objects have been well studied [1] for both high-resolution and real-time voxelizations. Low resolution applications, however, have thus far been largely ignored. Current algorithms do not preserve internal topology: thin bridges are broken, small loops are filled in, and disconnected regions merge together. Such topological features are essential components of many mathematical sculptures, so novel voxelization algorithms are needed.

Jia et al. [3] recently developed a 2-dimensional scaling algorithm that reduces images by a factor of two in size while maintaining internal topology. I generalize this method from pixelization to voxelization to provide a similar topology-preserving shrinking. I discuss the effectiveness of the algorithm and give examples of the resulting voxelizations rendered in Lego bricks.

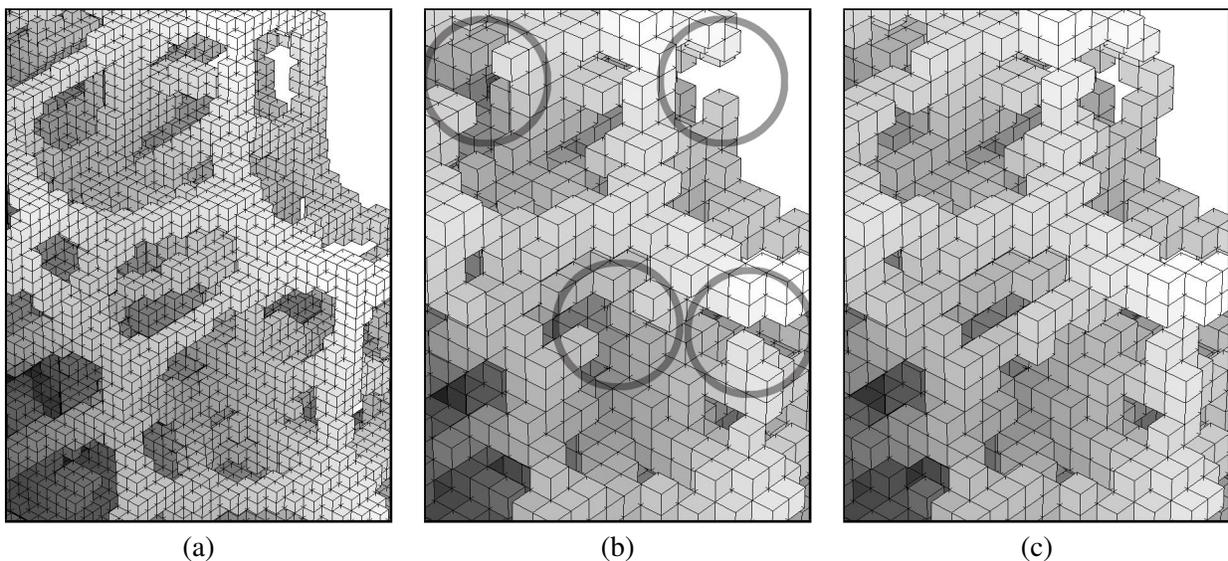


Figure 1: Voxelizations of the Möbius Net. (a) a high resolution voxelization, (b) a reduced version using a thresholding algorithm with $t = 3$, and (c) a reduced version using the proposed algorithm, 6-connectivity, and $t = 3$. The broken bridges are circled in the figure.

The Voxelization Algorithm

Summary. The proposed algorithm iterates over each $2 \times 2 \times 2$ block of subvoxels voxels, replacing the block with an empty or filled voxel as necessary to preserve connections between opposite faces of the surrounding $8 \times 8 \times 8$ block of subvoxels and prevent creating new connections. When in conflict, the algorithm prioritized preserving connections over not creating new connections. When ambiguous, the algorithm reverts to a threshold on the subvoxels.

Effectiveness. The algorithm is compared to the previous standard: a threshold algorithm where a voxel is filled if at least t of its 8 higher-resolution subvoxels are filled. In Figure 1, I use Bathsheba Grossman's Möbius Net [2] to illustrate the differences between these algorithms when applied to models with complex internal geometry. Observe that the proposed algorithm accurately reproduces the thin external bridges in the structure, unlike the thresholding algorithm. A $t=1$ threshold would preserve those bridges at the cost of filling in holes in the model that the proposed algorithm preserves.

Applications. The algorithm proposed herein is widely applicable to low-resolution voxelization, especially to mathematical art. My work uses primarily Legos to implement physical voxels. For the construction of the Möbius Net, after voxelization, layouts for pieces were generated automatically through a genetic algorithm [4], although more modern approaches are also available [5]. The results are illustrated in Figure 2.

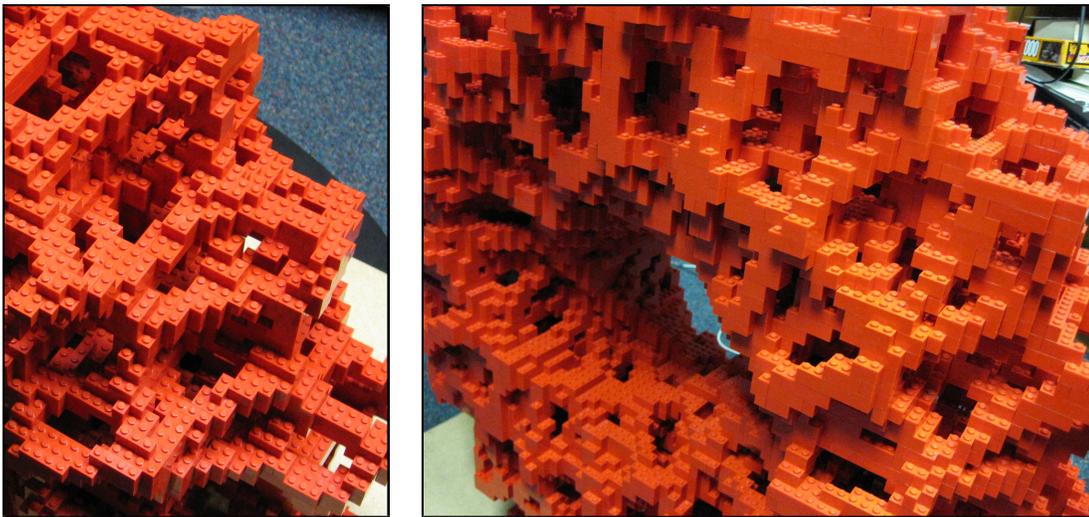


Figure 2 : Two views of the Möbius Net rendered in Lego.

References

- [1] Daniel Cohen-Or and Arie Kaufman. Fundamentals of surface voxelization. *Graphical Models and Image Processing*, 57(6):453 – 461, 1995.
- [2] Bathsheba Grossman. Möbius net, 2012. <http://www.bathsheba.com> (as of Mar 13, 2012).
- [3] Xingxing Jia, Daoshun Wang, Yu-jiang Wu, and Xiangyang Luo. A shrinking algorithm for binary images to preserve topology. In *2010 3rd International Congress on Images and Signal Processing*, 2010.
- [4] Pavel Petrovic. Solving LEGO brick layout problem using evolutionary algorithms. 2001.
- [5] Lynette van Zijl and Eugene Smal. Cellular automata with cell clustering. In *Automata*, pages 425–441, 2008.