Title:

Real-Time Visualization of Bead Based Additive Manufacturing Toolpaths Using Implicit Boundary Representations

Authors:
Shane Peelar, peelar@uwindsor.ca, University of Windsor
Jill Urbanic, jurbanic@uwindsor.ca, University of Windsor
Robert Hedrick, bob.hedrick@camufacturing.com, CAMufacturing Solutions Inc.
Luis Rueda, lrueda@uwindsor.ca, University of Windsor

Keywords:
Visualization, Additive Manufacturing, Part Model Generation, Simulation, Constructive Solid Geometry, Boundary Representations, Implicit Surfaces

DOI: 10.14733/cadconfP.2018.322-326

Introduction:

Additive Manufacturing poses unique challenges in part model generation and visualization compared to subtractive methods. Building a part using AM involves first "slicing" it into layers, and then rasterizing each layer using a fill technique. Here, the toolpath is the path that is used in depositing material to form the part. The part model essentially provides a boundary representation for the part being manufactured using the toolpath, process parameters and bead geometry. It is important to generate an accurate part model for visualization purposes and further processing.

Explicit boundary representations are commonly used to represent part models [9]. However, these representations have difficulty scaling with AM. Each toolpath necessarily has its own explicit boundary which represents the bead geometry along it. The explicit boundary representation of the entire part model must be computed from the union of all toolpath boundaries. The number of CSG unions can get into the hundreds of thousands quickly, taking unreasonable amounts of time to compute on even the best hardware. Furthermore, the number of facets of the final representation can grow unreasonably large, resulting in part models that consume gigabytes of disk space, even for a relatively small component (Fig. 1).

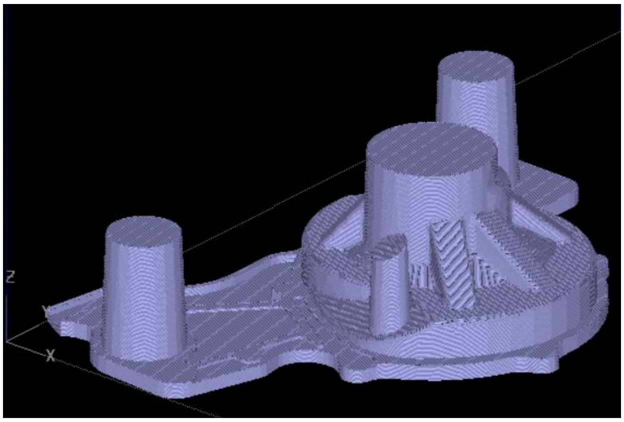

Fig. 1: Water pump casting pattern *.STL (174 x 236 x 62 mm @ 0.25 mm slice height): 1,018,892 KB.

Main Idea:

We demonstrate a scalable alternative for generating AM part models using an implicit boundary representation. Whereas an explicit boundary representation uses polygonal meshes or spatial subdivision to define the boundaries of an object, an implicit boundary representation uses a scalar field. The boundary of the object is the set of points in that field, which correspond to a specific isovalue—this is called an implicit surface or isosurface.

The primary benefit of an explicit boundary representation (Fig. 2) is that the result can be computed with exact arithmetic, improving robustness. Indeed, most state-of-the-art research in this area uses exact arithmetic for this reason [1]. The exactness of explicit boundary representations comes at a cost, however: it requires multiprecision arithmetic. This means that for large calculations, many memory management operations will need to be performed, and each calculation will involve much more computation than with a datatype native to the processor, resulting in poor performance.

Number of Meshes	*Processing time (approx.)*
10	20 seconds
100	4 minutes
1000	40 minutes
10,000	6 hours

Tab. 1: Processing time required for union of explicit boundaries using exact arithmetic.

IR Size	*Part model size (STL)*
2,509 KB	16,446 KB
6,547 KB	176,202 KB
8,256 KB	210,345 KB
93,387 KB	2,242,654 KB

Tab. 2: Part model sizes in relation to their toolpath Intermediate Representation (IR) size using explicit boundary representation.

Fortunately, exact results are not required for part model generation. An imprecise result within a confidence threshold is good enough for most visualization tasks, and even some hybrid manufacturing (machining operations performed on an additive manufactured model) tasks. One might be tempted to use a floating-point format in combination with an explicit boundary technique to achieve this; however, floating point arithmetic is non-associative and hence not suitable for performing geometric operations [4][8]. Even non-degenerate input can result in degenerate output due to a sign flip or a rounding error. Instead, we turn to an implicit boundary representation to address these problems.

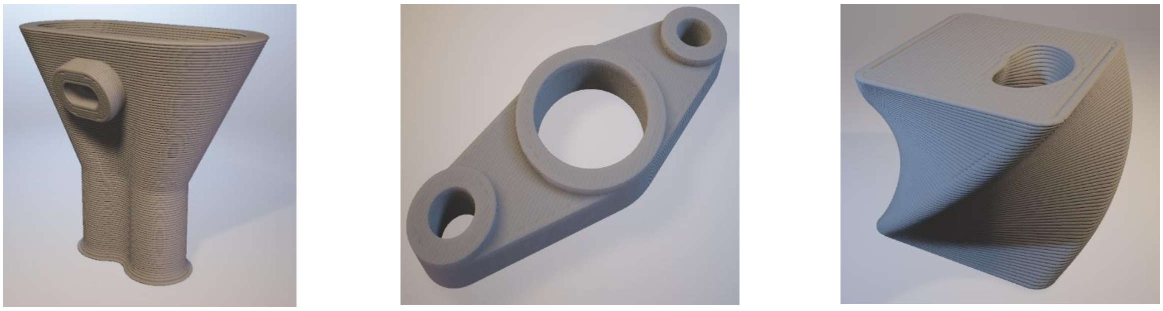

Fig. 2: Explicit boundary representations of three-part models.

Briefly, we represent the AM part in a field where each point has a nonpositive value if it is “outside” the part, and a positive value if it is “inside” the part. Locations in the field that have a value of 0 therefore represent the boundary of the part model. We use the Generalized Winding Number [3] to induce this field, although any function which maintains this property could be used as well, including a Signed Distance Function (SDF) [5-6].

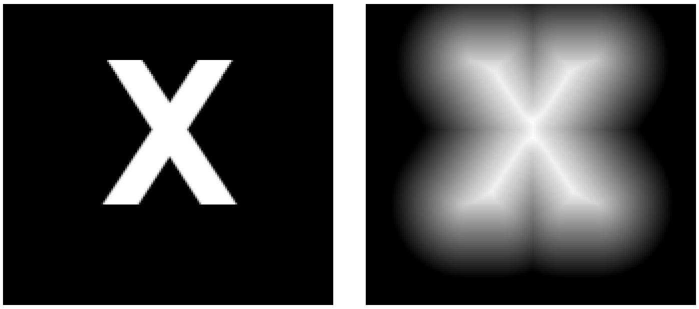

Fig. 3: Boundary representations of the character ‘X’. Left: explicit representation, Right: Implicit representation using a Euclidean SDF. [5]

The Generalized Winding Number (GWN) is a generalization of the winding number to three dimensions. Briefly, it provides a measure of “insideness” of a point with respect to an object. For a watertight mesh, the GWN of every point inside the mesh is 1, and the GWN of every point outside the mesh is 0. Where the GWN really shines is in handling degenerate meshes: meshes that contain self-intersecting edges, have facets that are not oriented correctly, or meshes that are not watertight. In these cases, the GWN provides a smooth transition from regions of the mesh that are “inside” to ones that are “outside”, providing robustness where an SDF would have trouble. We use a special case of the GWN [10] which can be used to efficiently compute the GWN for triangle-based meshes.

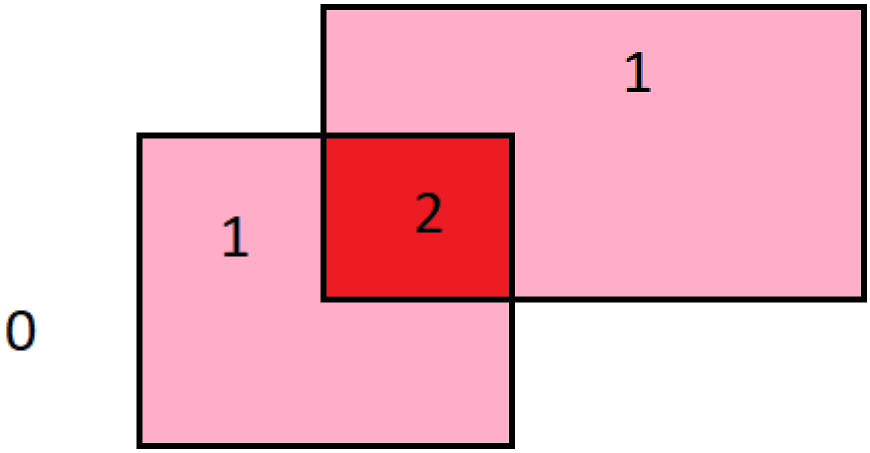

Fig. 4: Winding numbers of points with respect to the two overlapping boxes. Notice that the overlapping region still has a positive winding number.

Our approach enables the bead geometry of the toolpaths to be defined explicitly using a swept cross section, which is convenient for designers. However, the Boolean union of these solids are not directly computed as in an explicit boundary approach. Instead, we sample the GWN field induced by all explicit boundaries of all toolpaths and locate the implicit surface at isovalue 0.5. This implicit surface is the boundary of the part model. There are two approaches to sampling this field: a dense method and a sparse method. The terms “dense” and “sparse” refer to the assumptions made about the model’s distribution of information. If the field has a distribution of information where there are many implicit boundaries, a dense method is preferable. On the other hand, if the field has a distribution of information where there are few implicit boundaries, then a sparse method is preferable.

With a dense approach, the field is sampled at fixed intervals in a grid-like fashion. This is more amenable to direct implementation, however can suffer from scalability problems with larger part models. It does have the benefit of being readily computed on a GPU, using the built-in trilinear interpolation hardware within it to interpolate within the already obtained samples for further processing and visualization. Here, the model is represented as a multidimensional array of samples. It can be thought of as a “brute force” approach to sampling data—although one that can outperform a sparse approach in some cases with a powerful GPU.

With a sparse approach, the field is sampled in dynamic intervals. A Sparse Voxel Octree (SVO) is constructed which samples the GWN field, hierarchically honing down towards the boundaries of the part model. The SVO is Hermite tagged with gradients as is common with dual contouring approaches [7] to preserve sharp features when constructing isosurfaces. The depth of the SVO is directly related to how much information and detail is preserved. Few samples are made in regions on either side of the chosen isovalue. However, samples are much more frequently made in areas near boundaries. With a sparse approach, the model is represented as a hierarchical tree of samples with associated metadata.

Both approaches have diminishing returns with the desired resolution chosen. For purely random data, a sparse approach will perform worse than a dense approach, since the tree will become excessively large compared to an equivalent array in a dense approach. However, sparse approaches have the main benefit of performing fewer samples overall compared to dense approaches when there are few implicit boundaries. We have chosen to use a sparse approach in our implementation as we’ve found this to be the case for AM parts.

Finally, an isosurface is constructed at the implicit boundary where the GWN field is 0.5. This represents the boundary of the part model, which can be saved in the conventional STL format for compatibility with major software packages. The resolution of the final mesh will depend on the sampling resolution and isosurface generation method chosen. This gives flexibility to users, as they can choose their own balance between part model size and detail preservation. We use Manifold Dual Contouring in our implementation [7], which guarantees that the extracted isosurface is indeed manifold.

Future Work:

Using our approach enables floating point hardware to be used throughout the calculations, providing a large speed improvement in comparison to exact approaches. Memory can be allocated in large chunks, providing even more speed benefits. There are still open problems to be addressed, however. Although IEEE floating point formats are not suitable for solving geometric problems, it is possible that alternative floating-point formats may yield better results. Posits [2], a proposed alternative to IEEE floating point, are associative and hence may be a better fit here. If this is the case, it may be possible to construct an explicit boundary representation using floating point that is robust and much faster than existing methods, although with inexact results. Since no hardware exists yet that can handle these formats, the first practical implementation will likely need to use an FPGA for acceptable performance.

Conclusions:

Part model generation and visualization for additively manufactured parts poses unique challenges that prevent explicit boundary representations from being used. We discussed the benefits and drawbacks of explicit boundary representations, including exact calculations commonly used with them. We demonstrated an alternative method for generating and visualizing part models using implicit boundary representations, taking advantage of the fact that exact results are not strictly required for these tasks. Our approach addresses both the processing time and part model size problems that are present with explicit boundary approaches. We showed that our method is scalable and provides opportunities for fine tuning and further processing, and that it can be accelerated with common hardware.

Acknowledgements:
An Ontario Centres of Excellence (OCE) TalentEdge Academic Internship grant with CAMufacturing Solutions Inc. funded this research.

References:

[1] Fabri, A.; Pion, S.: The Exact Computation Paradigm, https://www.cgal.org/exact.html.

[2] Gustafson, J.L.; Yonemoto, I.T.: Beating Floating Point at its Own Game: Posit Arithmetic, Supercomputing Frontiers and Innovations, 4(2), 2017, 71-86.

[3] Jacobson, A.; Kavan L.; Sorkine-Hornung, O.: Robust inside-outside segmentation using generalized winding numbers, ACM Transactions on Graphics, 32(4), 2013, 33.

[4] Kettner, L.; Mehlhorn, K.; Pion, S.; Schirra, S.; Yap, C.: Classroom Examples of Robustness Problems in Geometric Computations, Computational Geometry: Theory and Algorithms, 40(1), 2008, 61-78.

[5] Node.JS package bitmap-sdf.: https://www.npmjs.com/package/bitmap-sdf.

[6] Osher, S.; Fedkiw, R.: Signed distance functions, Level set methods and dynamic implicit surfaces, Springer, 2003, 17-22. https://doi.org/10.1007/0-387-22746-6_2

[7] Schaefer, S.; Ju, T.; Warren, J.: Manifold dual contouring, IEEE Transactions on Visualization and Computer Graphics, 13(3), 2007, 610-619. https://doi.org/10.1109/TVCG.2007.1012

[8] Schirra, S.: Robustness and precision issues in geometric computation, Handbook of Computational Geometry, Elsevier Science, 1999.

[9] Urbanic, J.; Hedrick, R: Developing a virtual model for the fused deposition rapid prototyping process, Proceedings of the Life Cycle Engineering Conference, 2009, 131-137.

[10] Zhou, Q.; Grinspun, E., Zorin, D.; Jacobson, A.: Mesh arrangements for solid geometry, ACM Transactions on Graphics, 35(4), 2016, 39. https://doi.org/10.1145/2897824.2925901