Spatially Adaptive Volume Tools in Bifrost

Morten Bojsen-Hansen
morten.bojsen-hansen@autodesk.com
Autodesk
Canada

Michael B. Nielsen
michael.nielsen@autodesk.com
Autodesk
Canada

Konstantinos Stamatelos
konstantinos.stamatelos@autodesk.com
Autodesk
Canada

Robert Bridson
robert.bridson@autodesk.com
Autodesk
Canada

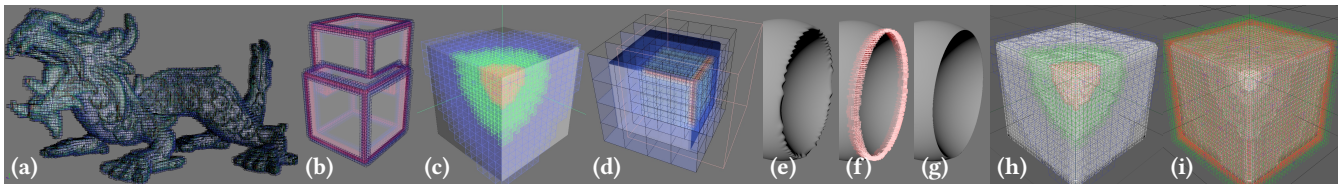

Figure 1: (a) An adaptive level set with three levels of resolution along the surface reduces the voxel count by sixty percent compared to the input OpenVDB narrow band level set. Adaptivity can be controlled via (b) an error predicate, (c) an arbitrary function such as the distance to a point or (d) an AABB. (e–g) Spatial adaptivity benefits CSG operations by automatically resolving new features. Our mesher allows both (h) meshing of adaptive volumes and (i) adaptive meshing of uniform volumes.

ABSTRACT

The level set method offers many advantages over *e.g.* meshes for modelling and visual effects, but a naïve implementation is both computationally expensive and memory intensive. Narrow band level sets alleviate both issues but are still limited by the finest detail resolved due to uniform resolution along the surface. Voxel structures that are adaptive along the surface improve this [Frisken et al. 2000], but have not seen wide adoption. This is presumably due to difficulties matching the performance of optimized narrow band implementations like industry standard OpenVDB [Museth 2013]. We present the adaptive level set implementation in Bifrost which is competitive with OpenVDB in speed while offering lower memory usage thanks to spatial adaptivity. Our contributions include novel algorithms for adaptive sharpened B-spline interpolation of volumes in general, voxelizing meshes and points into adaptive level set volumes, and meshing adaptive level sets.

CCS CONCEPTS

• **Computing methodologies** → **Volumetric models**.

KEYWORDS

level sets, spatial adaptivity, sampling, meshing, FX

ACM Reference Format:
Morten Bojsen-Hansen, Michael B. Nielsen, Konstantinos Stamatelos, and Robert Bridson. 2021. Spatially Adaptive Volume Tools in Bifrost. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks (SIGGRAPH '21 Talks), August 09-13, 2021.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3450623.3464642

SIGGRAPH '21 Talks, August 09-13, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8373-8/21/08.
https://doi.org/10.1145/3450623.3464642

1 SPATIALLY ADAPTIVE DATA STRUCTURE

We represent spatially adaptive volume data using the *tile tree* LOD voxel data structure of Nielsen and Bridson [2016]. Data is stored at voxel corners in tiles of 4^3 voxels while using a branch factor of 2^3 similar to an octree. Leaf voxels can exist at any depth, facilitating level sets with varying resolution along the surface. Several of our algorithms leverage the error predicate introduced by Frisken *et al.* [2000] modified to account for the tile tree's decoupling of tile size and branch factor. To determine if a branch of the tree should be or stay refined, we test if it is within a threshold distance from the surface. If not, the branch is coarsened or remains coarse. If true, we evaluate if the $(2+1)^3$ samples at the corners of the voxels in the parent tile that cover the child branch fail to accurately predict any of the unique $(4+1)^3-(2+1)^3$ corner samples in the child tile by linear interpolation. If the error predicate fails, we retain or refine the branch. In practice, an error between five and ten percent of the voxel size leads to an adaptive representation indistinguishable from a narrow band with uniform resolution along the surface. Similar to Frisken *et al.*, we use this to perform top-down CSG operations of adaptive level sets, as well as bottom-up coarsening of level set volumes (in our case when loading VDBs).

2 VOXELIZING MESHES AND POINT CLOUDS

The input mesh is assumed to be a soup of triangles. No assumptions are made about connectivity or orientation and triangles are allowed to intersect. When voxelizing the input mesh, we exploit the tile tree we build simultaneously as an AABB structure to accelerate operations. Proceeding depth-first from the root(s) of the tile tree, evaluating different branches in parallel, we dynamically split the

AABB of the mesh elements in the parent tile into AABBs of mesh elements for the child tiles. Recursion stops if the AABB is empty or a coarsening criterion is met. As criteria we use the modified error predicate of Frisken [2000] as well as criteria based on user-defined functions (Figure 1c) and resolution regions (Figure 1d). AABBs are only stored for the branches of the tile tree that are currently being evaluated and within each branch only for two levels at a time. During the depth-first pass we compute unsigned distances and optionally edge-triangle intersection counts with a robust edge-triangle intersection test based on integer arithmetic.[1]

To generate signs, we take one of two approaches depending on if the mesh has a consistent interior. If yes, signs are generated by a parallel graph traversal of all the edges in the tile tree starting from the outside and changing sign when traversing an edge with an odd number of triangle intersections. Meshes with holes or self-intersections are instead handled by dilating the unsigned distance by a user-specific maximum hole size and then flood-filling the signs from the outside using a parallel union-find data structure and stopping as soon as the zero-crossing is encountered [Xu and Barbič 2014]. Point clouds are voxelized in a similar fashion. The points are CSG unioned based on their position and radii, and we avoid refining interior regions where the particle cloud is dense, but where the adaptive level set should remain coarse.

Voxelization into adaptive level sets adds considerable computational complexity compared to a narrow band approach: the top-down construction computes distances at coarser levels of the tree, and more expensive robust edge-triangle intersection tests are needed for sign-generation. The extra overhead is often balanced by the reduction in voxel count, in which case the conversion is faster than state-of-the-art, OpenVDB. In other cases, conversion using OpenVDB is faster (see video). However, while this initial conversion step may be a bottleneck, the reduction in voxel count can significantly reduce memory bandwidth requirements and computation for subsequent level set operations.

3 ADAPTIVE B-SPLINE SAMPLING

Even with adaptivity, the finest voxels are likely larger than pixels in final renders, so care must be taken with sampling. Nielsen and Bridson [2016] showed it was possible to construct triquadratic B-splines with C^1 smoothness even across resolution jumps, which is reasonable for matte level sets (silhouettes will be smooth and normals will be continuous) but not quite adequate for specular surfaces requiring continuous curvature. Density fields also may require higher order smoothness: even with anti-aliased voxel values, grid artifacts are quite apparent with *e.g.* Catmull-Rom interpolation. We faced two problems, however: implementing cubic B-splines efficiently on adaptive grids (with T-junctions at resolution jumps), and avoiding excessive blurring in detailed regions.

For the former, we create haloed tiles, with extra layers of stored values around the normal tile size so that the B-spline can be evaluated in the domain of the tile without needing values beyond the halo. We construct these haloed values in a top-down manner (coarsest root tile first), either by copying data from neighbouring tiles of the same resolution where they exist, extrapolating at the edge of the entire domain, or computing values in a 3-wide band along a resolution jump (to a coarser tile) which will continuously match the value and two derivatives at the jump. To evaluate in a given voxel, we find the finest resolution tile which contains it and use the regular uniform B-spline sampling in that halo tile.

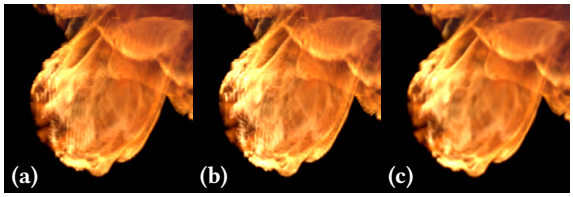

Figure 2: (a) Tri-linear and (b) tri-Catmull-Rom can show grid artifacts, reduced by our (c) tri-cubic B-spline sampler.

To mitigate blurring, we designed a $3 \times 3 \times 3$ prefilter to sharpen the data so that the subsequent B-spline sampler exactly interpolate *cubic* (not just affine) regions of the original function, while behaving as isotropically as possible on quartics to reduce grid-aligned artifacts. This filter can easily be run across all tiles of the raw input in parallel, ignoring resolution jumps since the halo computation overwrites the values too close to the jump to need special treatment. The supplemental document provides full details.

4 MESHING ADAPTIVE LEVEL SETS

To mesh adaptive level sets we developed a parallel lock-free version of the Dual Marching Cubes method of Schaefer and Warren [2004] adapted to work on tiles of any dimension. The algorithm is divided into four parallel loops over each tile. First, we place a dual vertex in each voxel and compute the depth of each voxel. This information is enough to derive the dual grid on-the-fly in subsequent passes. Secondly, we count the number of mesh vertices created by Marching Cubes in each tile. We use vertex counts to derive an offset per tile into a flat array of mesh vertices. In the third pass, we use the offsets to directly create the mesh vertices in the array. Finally, we use a parallel reduction to create the mesh faces. We avoid duplicate faces by making sure any face that crosses a tile boundary is assigned to a unique tile: faces that cross a resolution jump are created by the tile with finer resolution with a tie-breaker in case of tiles that have the same resolution going to the tile with the lowest coordinates. By modifying the per voxel depth, we also allow on-the-fly adaptive meshing even for uniformly sampled level sets. We demonstrate this in the supplemental video.

[1] We transform coordinates to 64-bit fixed-point, then use exact 128-bit integer arithmetic, to avoid problems with unsafe compiler optimizations of floating-point. We also use Simulation of Simplicity [Edelsbrunner and Mücke 1988] to avoid special cases when mesh elements exactly align with the adaptive grid.

REFERENCES

H. Edelsbrunner and E. P. Mücke. 1988. Simulation of Simplicity: A Technique to Cope with Degenerate Cases in Geometric Algorithms. In *Proc. Fourth Annual Symp. Computational Geometry*. ACM, 118–133.

Sarah F. Frisken, Ronald N. Perry, Alyn P. Rockwood, and Thouis R. Jones. 2000. Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics. In *Proc. SIGGRAPH 2000*. 249–254.

Ken Museth. 2013. VDB: High-Resolution Sparse Volumes with Dynamic Topology. *ACM Trans. Graph.* 32, 3, Article 27 (July 2013).

Michael B. Nielsen and Robert Bridson. 2016. Spatially Adaptive FLIP Fluid Simulations in Bifrost. In *ACM SIGGRAPH 2016 Talks*. ACM, Article 41.

Scott Schaefer and Joe Warren. 2004. Dual Marching Cubes: Primal Contouring of Dual Grids. In *Proc. 12th Pacific Conference on Computer Graphics and Applications (PG 2004)*. 70–76.

Hongyi Xu and Jernej Barbič. 2014. Signed Distance Fields for Polygon Soup Meshes. In *Proc. Graphics Interface 2014 (GI '14)*. CAPS, 35–41.