Isosurface Extraction in a Simple C/C++ Library

Jonathan Glines, with undergraduate advisor Dr. John Edwards

Idaho State University, College of Science and Engineering

Isosurface Extraction

Many applications, such as medical imaging, mathematical visualization, and artificial terrain generation, require the generation of isosurfaces from sample data. An isosurface in three dimensions is simply the surface contour of a three-dimensional scalar field for a given contour value. These surface contours are analogous to the contour lines that we are familiar with from geological maps that represent elevation.
The process of isousface extraction begins with a scalar field function $f : \mathbf{R}^3 \mapsto \mathbf{R}$. This function might be defined by empirical data, such as samples from a CT scan, or by a mathematical formula. It is assumed that this function is relatively continuous and that it does not oscillate faster than the resolution we are sampling at (to satisfy the Nyquist limit).
Once a scalar field function has been selected, an isosurface value $s \in \mathbf{R}$ is also selected. This isosurface value is analogous to a specific contour elevation in a geological elevation map. While the value s can in principle be any real number, it is often zero in practice. This is because we can easily manipulate the scalar field function f by adding a constant offset.
The isosurface itself is defined as the set of points $S := \{x : f(x) = s\}$. The ultimate goal of isosurface extraction is to generate a triangle mesh representing S.

Original Marching Cubes

With the scalar field function f in hand, we can readily generate a lattice of sample values. The original marching cubes algorithm, along with most of its variations, considers one voxel cube (i.e. eight samples arranged in a cube) from this sample lattice at a time. From the eight sample values in this voxel, we compute an 8-bit index according to the samples that are above or below the isosurface. The primary innovation of the marching cubes algorithm is the $2^8 = 256$ entry lookup table that enable us to quickly generate vertex positions and triangles for our isosurface mesh.

Accounting for Ambiguity

The original marching cubes algorithm as described by Lorensen and Cline uses 15 equivalence classes to generate the lookup tables. It was shown early in the use of marching cubes that certain functions, such as those with saddle points, could expose ambiguities in the way marching cubes was originally described. More advanced treatments of the algorithm, such as Chernyaev's Marching Cubes 33 and Nielson's Asymptotic Decider, use additional equivalence classes that resolve these ambiguities.

Dual Marching Cubes

The dual of the marching cubes algorithm is easiest to understand by looking at the popular voxel game Minecraft, since cube meshes generated by Minecraft are topologically the same as any dual marching cubes mesh. The algorithm that generates Minecraft-style meshes, called "cuberille," does not perform any interpolating or smoothing. More advanced dual methods will "relax" or "smooth" the vertex positions of the cuberille mesh to produce a mesh much closer to the actual isosurface.

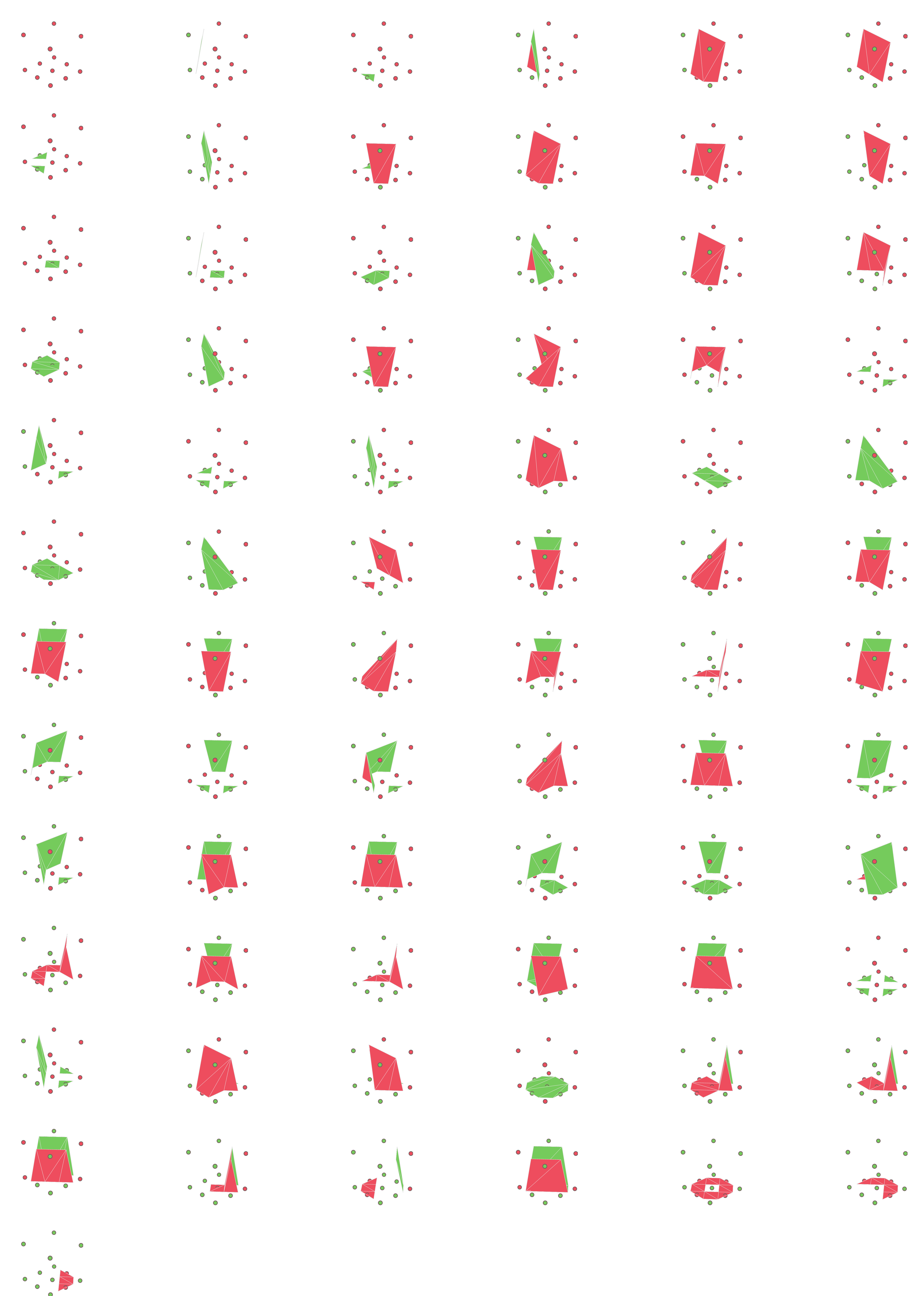
Transvoxel Algorithm

The Transvoxel isosurface extraction algorithm sets itself apart from ordinary marching cubes by allowing seamless transitions between half and full resolution samples. This algorithm was developed by Eric Lengyel in his PhD dissertation titled "Transition Cells for Dynamic Multiresolution Marching Cubes."

Cascading Transition Voxels

We are working to generalize the Transvoxel algorithm for transitioning between n levels of detail within the adaptive sample space of an octree. We hope to apply this new algorithm to our research on generalized Voronoi diagram computation.

73 Transition Cell Cases



Implementation

We have implemented a number of these isosurface extraction algorithms in an easy-to-use library. The result is a C/C++ library we call `libmc`. We are developing `libmc` in parallel with our other undergraduate projects that should benefit from an isosurface extraction library, such as LIDAR mapping and generalized Voronoi diagram computation.
`libmc` is still in the early stages of development. Source code for the library is available on GitHub at the following URL: `https://github.com/auntieNeo/libmc`

Example Output

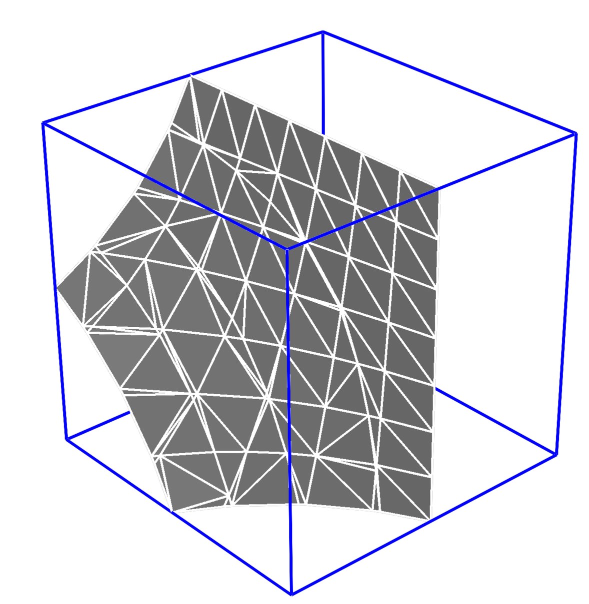


Fig. 1: Original Marching Cubes

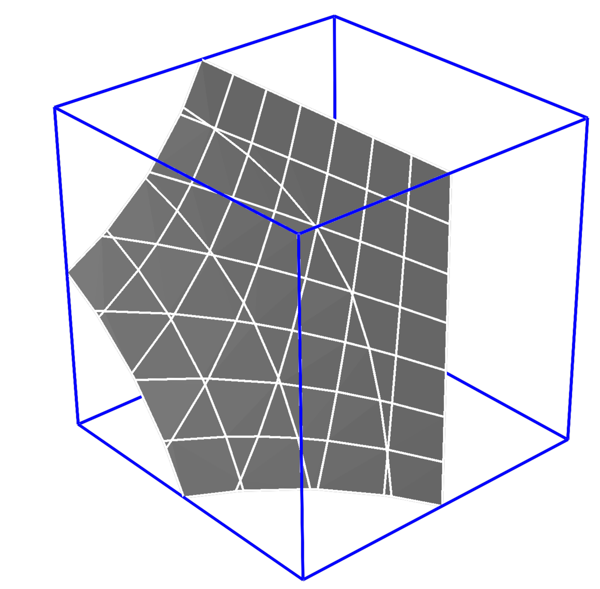


Fig. 2: Marching Cubes Patch

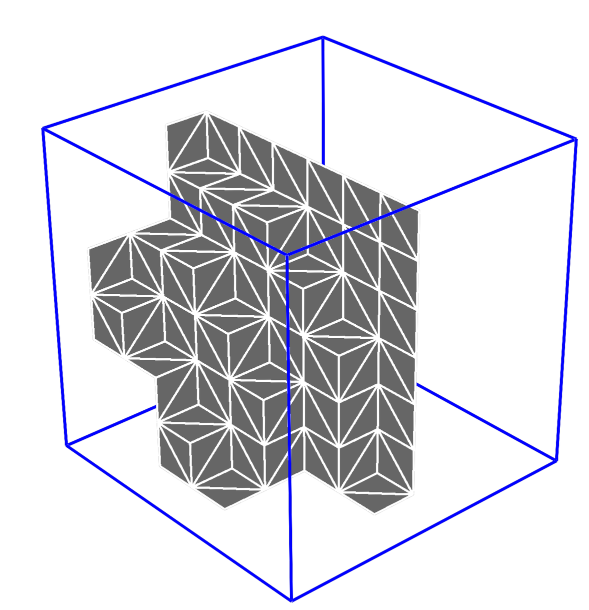


Fig. 3: Cuberille

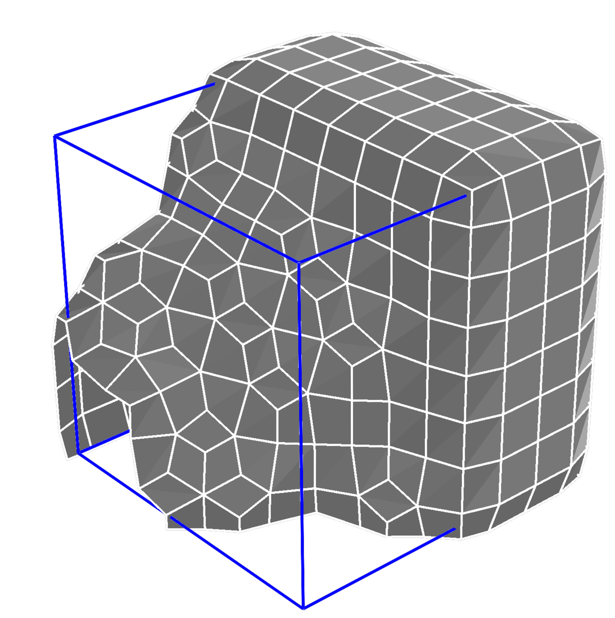


Fig. 4: Nielson's Dual Marching Cubes

Contact

I can be reached at glinjona@isu.edu. I also have a personal web page at `http://glines.net`.
My research advisor, Dr. John Edwards, keeps a faculty page at the URL `http://www2.cose.isu.edu/~edwajohn/` and can be contacted via email at edwajohn@isu.edu.