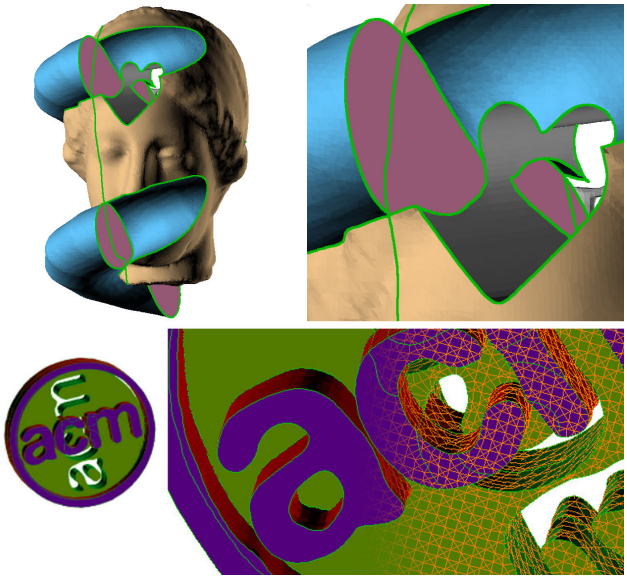
Extracting Boolean Isosurfaces from Tetrahedral Meshes

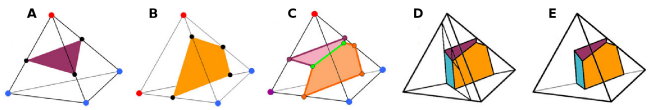
Gabriel Taubin*
Brown University

Peter G. Sibley†
Brown University



Surfaces separating the inside from the outside of most solid objects of interest are piecewise smooth: they can be decomposed into smooth surface patches meeting along smooth boundary curves called creases. Normal vectors are discontinuous across creases, and creases join at their ends forming corners. Rather than identifying sharp features in an isosurface with additional Hermite data or detecting large changes in surface normals (see [Ju et al. 2002] and references within), we consider the Constructive Solid Geometry (CSG) representation of surfaces as boundaries of regularized Boolean combinations of half spaces. But here each half space is defined by one piecewise linear implicit function inequality. The Boolean Isosurface Algorithm for tetrahedral meshes (TBIso) outlined in this sketch extends the classical isosurface algorithms for tetrahedral grids from smooth implicit surfaces to volumetric sampled CSG surfaces in such a way that the feature lines are identified and extracted as poly-line networks. These feature lines and corners can be preserved during subsequent smoothing and simplification.

The figure below shows a Boolean combination of three half spaces within a single tetrahedral cell. A and B show the surface boundaries of two of the three half spaces (red are positive function values , blue are negative values). C, D, and E show a series of added vertices, edges, and faces which are the boundary of the intersection of the three half spaces. Notice that we can have vertices that lie in the interior, faces, edges or even vertices of cells. E shows the final boundary of the CSG combination.



Marching Tetrahedra (MT) [Doi and Koide 1991] operates on a tetrahedral grid and one piecewise linear function $f(x)$ defined by its values at the grid vertices. The signs (positive or negative) of the function f at the four grid vertices determines the connectivity (triangle, quadrilateral, or empty) of the isosurface within each tetrahedron. There are $16 = 2^4$ cases,which reduces via symmetries to 3 unique cases, these can be stored in a look up table (LUT) indexed by a 4-bit signature word composed concatenating the signs of f at the four corners. The locations of the vertices along the grid edges are determined via linear interpolation.

TBIso takes as input: (i) a tetrahedral grid, (ii) a collection $f_1 \dots f_N$ of scalar fields defined at the vertices of the tet-grid (i.e., piece-wise linear functions), and (iii) a Boolean expression b of N variables in disjunctive normal form (b corresponds to a CSG tree whose leaves are functions $\{f_i\}$). The output is: (i) a polygonal mesh, (ii) a list of crease edges, and (iii) a list of corner vertices. The MT approach of precomputing and storing topology information using a single LUT isn't feasible here. The possible number of cases for non-trivial Boolean expressions is $2^{(2^N-1)}-1$, thus precomputing all cases for even small N is infeasible. Furthermore the topology of the Boolean isosurface depends on function magnitude *not just their signs* at the vertices of a tetrahedral cell. At run time, we compute three tables which depend on b and the number of variables N and contain partial topology information. The first table, which we refer to as the cell to face (C2F) table, exploits the relation that a pair of terms (conjunctions) in b may share a face if they differ a 1-bit. Similar relations for pairs (shared edges, F2E) and triplets (shared vertices, E2V) of bits in terms of b are used to form two other look up tables. These tables use the structure of b to avoid the N^2 overhead of naively clipping every plane determined by f_i in a cell against all others for each term in b and forming the union. Then within each cell we *evaluate the tables* incorporating the function values. We work backwards through E2V, F2E, and C2F. Finally hash tables are used to identify common vertices that lie on cell faces, cell edges or cell vertices to produce an indexed face set representation. Our current implementation uses unstructured tet-grids, however an out-of-core extension would simply require z-sorting the tetrahedral grid then working on slices. Our method also handles singular cases when the functions f_i and corresponding half spaces aren't in general position.

We've performed some initial experiments on several examples. The top figure is a $139K$ face mesh corresponding to the Boolean isosurface of 4 functions and a boolean expression (in DNF) consisting of a 6 terms defined over $12,290,298$ tet-grid. This tet-grid was generated from a 128^3 regular hexahedral grid. The table construction time was $8s$ and extraction took $15m$, on a Xeon 2GHz with 1.5GB of ram. The tagged feature line network shown in green is where halfspaces intersect and are used to preserve creases when applying hierarchical $\lambda|\mu$ smoothing. The faces of the meshes are colored depending on which of the functions they bound. The ACM figure, shows 6 functions on $108,045$ tets with 15 Boolean terms, table construction took $29ms$ and extraction took $53s$.

DOI, A., AND KOIDE, A. 1991. An Efficient Method of Triangulating Equivalued Surfaces by Using Tetrahedral Cells. *IEICE Transactions on Communications and Electronics Information Systems E74*, 1, 214–224.

JU, T., LOSASSO, F., SCHAEFER, S., AND WARREN, J. 2002. Dual contouring of hermite data. In *Siggraph'02*, 339–346.

*e-mail:taubin@brown.edu
†e-mail: pgs@cs.brown.edu