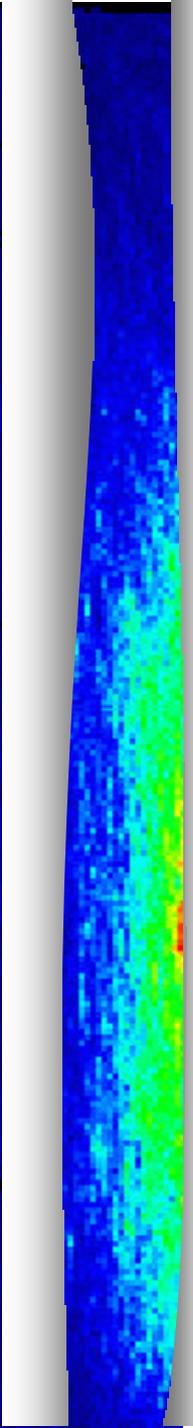


Where and When to Look in Time-varying Volume Visualization

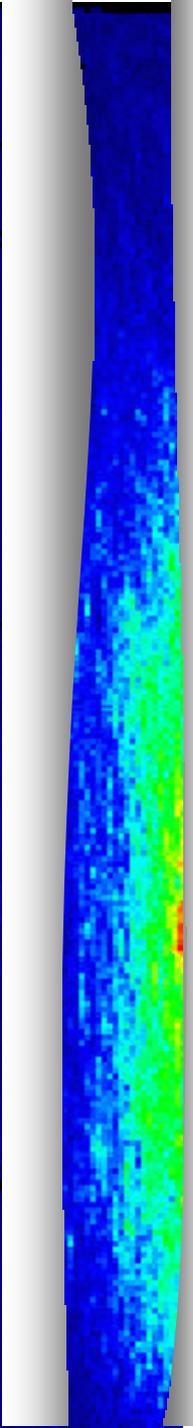
Jack Snoeyink
UNC-Chapel Hill

Jarek Rossignac
Georgia Tech



CARGO incubation project

- **Computational Topology for Exploring Time-varying Volume Data**
 - Jarek Rossignac, Georgia Tech
 - Jack Snoeyink, UNC Chapel Hill
 - Valerio Pascucci, LLNL
 - Peter Lindstrom, LLNL
- **Thesis:** Computed topological structure can provide global context for exploring large data sets from scientific simulations
- **Goal:** prototype tool for visualizing iso-surfaces in time-varying volume data



Applying computational topology

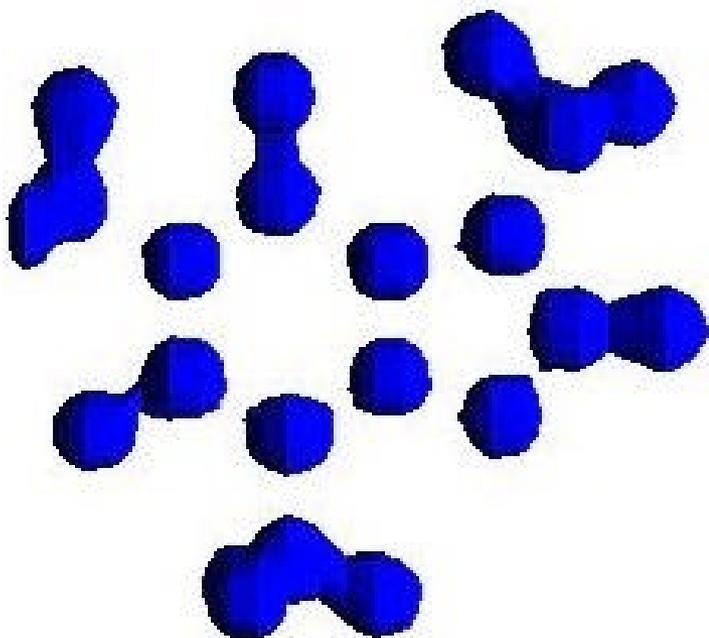
- Illustrating our thesis: *contour trees in 3D*
- Time-varying volume visualization
 - Sample data sets: *scientific & eng. simulation*
 - Prototype interface: *contour spectra & iso-surfaces*
- Pentatope meshes
 - A data structure supporting iso-surfaces
 - Implementing the incremental flip algorithm
 - Arithmetic complexity
 - Handling degeneracies

Level sets in density data



- Many technologies give density samples: $f(p)$ on a 3-D lattice.
- Define **Level Sets**
 $L(v) = \{p \mid f(p) = v\}$.

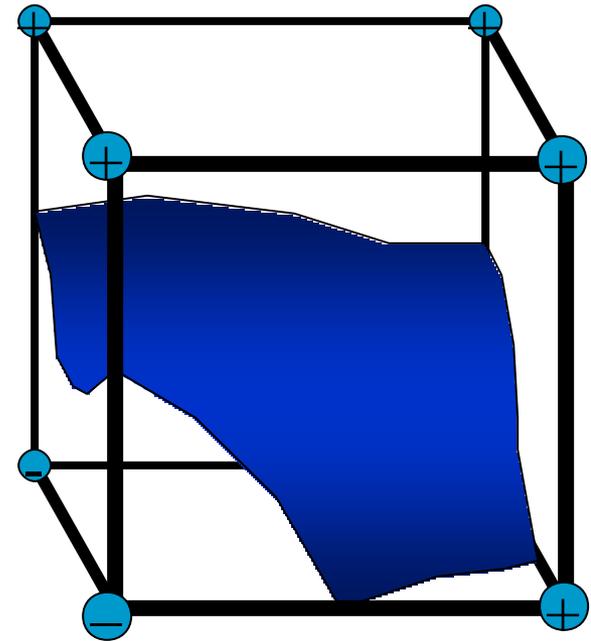
Level sets in density data



- Many technologies give density samples: $f(p)$ on a 3-D lattice.
- Define **Level Sets**
 $L(v) = \{p \mid f(p) = v\}$.
- Define **Contour** as a connected component of the level set.

Marching cubes

- Check every cube,
- If it hits surface, make portion of the level set.
- 14-16 cases

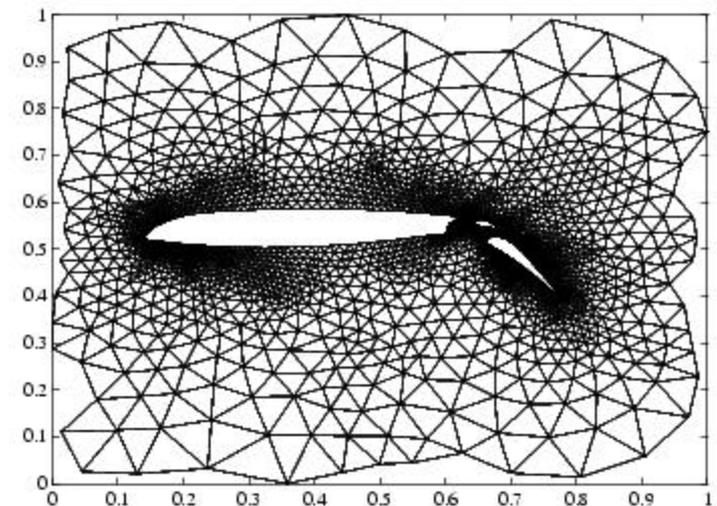
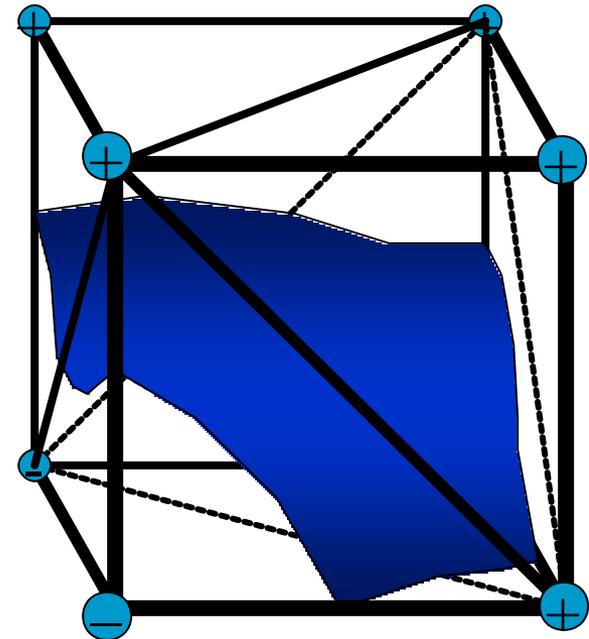


- Lorensen & Cline 87, Marching cubes
- Nielsen & Hamann 91, fix ambiguities
- Van Gelder & Williams 92, octrees

Marching tetrahedra

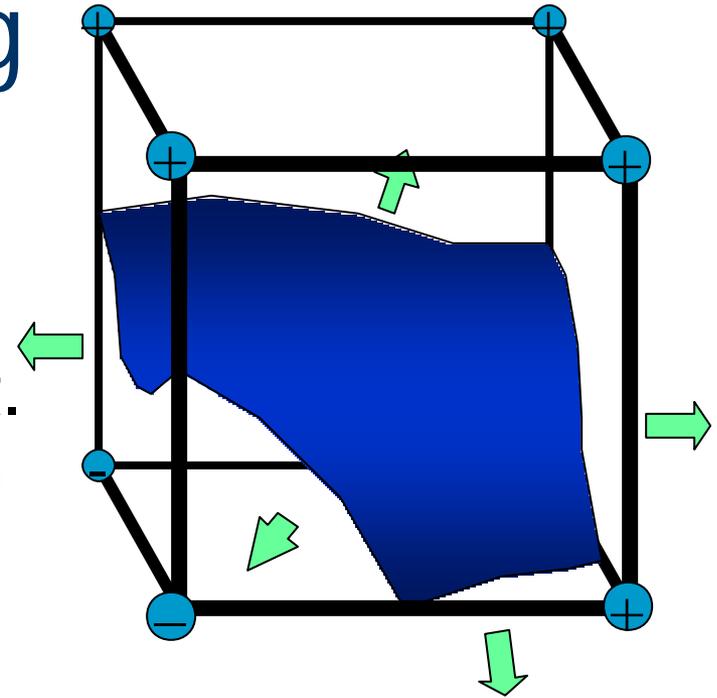
- An easy variant on MC
- Removes ambiguities
- Subdivide cubes into 5 to 24 tetrahedra [CMS01]

- Or use an irregular mesh & detail-dependent resolution



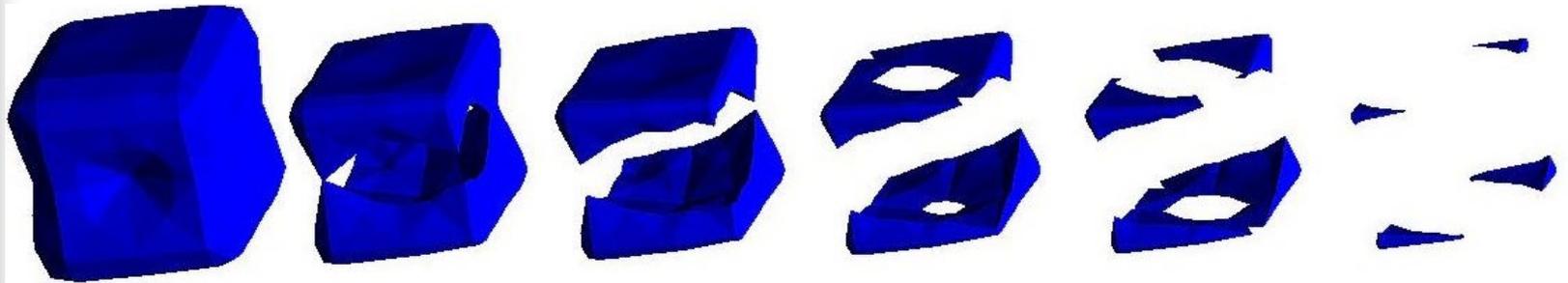
Contour following

- Can follow the contour *if* we know a start point.
- How to find start point?



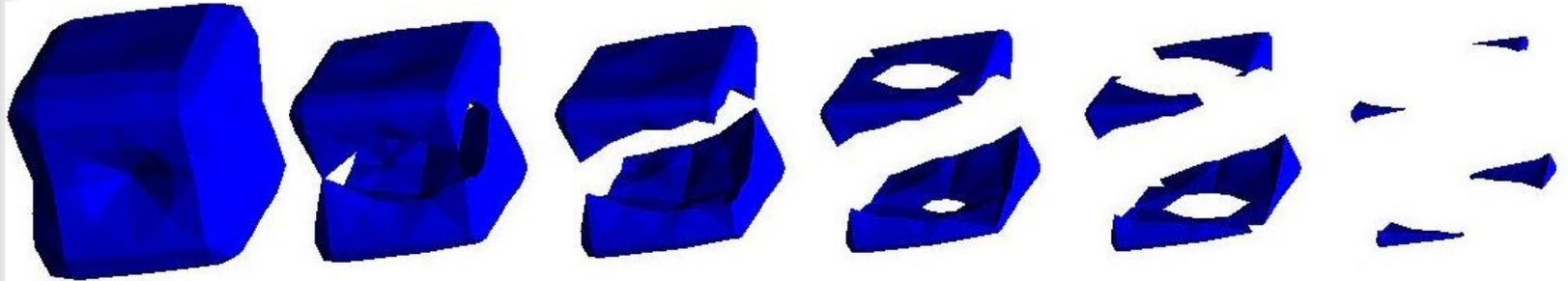
- Shinagawa et al. 91, extreme graph
- Cignoni et al. 95, interval tree
- Shen & Johnson 96, span space

Morse theory & Contour trees

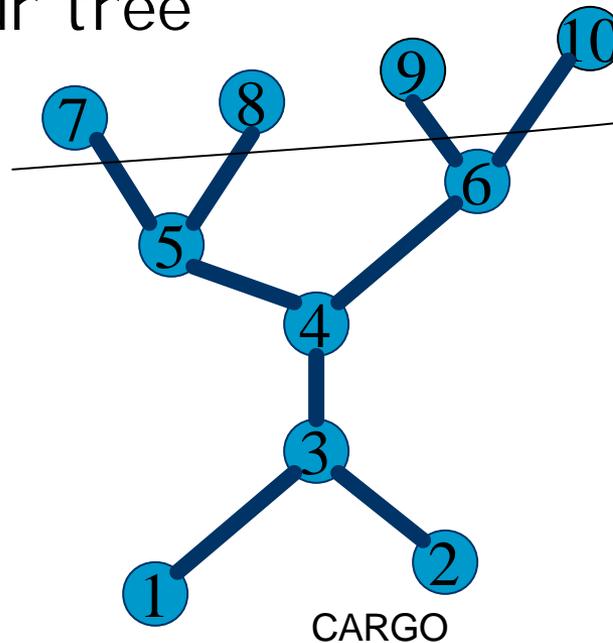


- Karron et al 95, digital Morse theory
- Bajaj et al 97-99, seed sets
- Van Kreveld et al. 97, 2-D contour tree
- Tarasov & Vyalyi 98, 3-D contour tree
- Carr et al 00, n -D contour tree
- Edelsbrunner, Harer et al. 01-

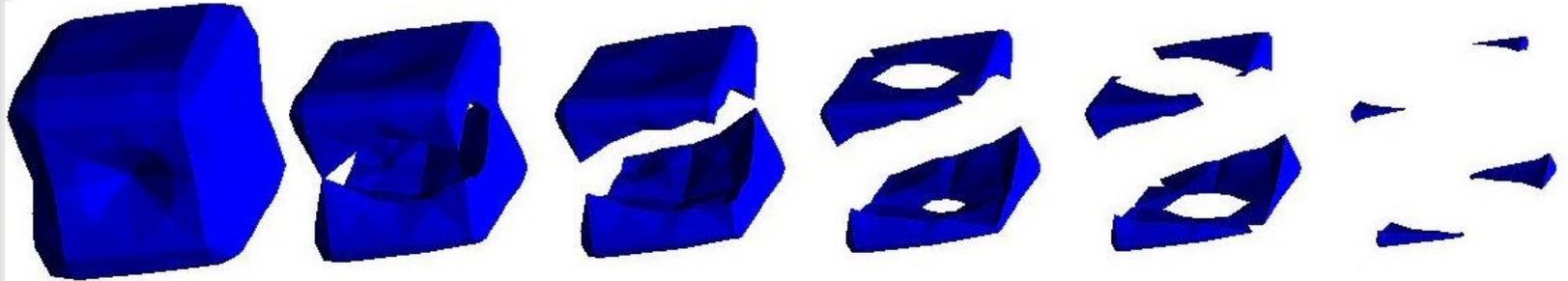
Evolution of level sets



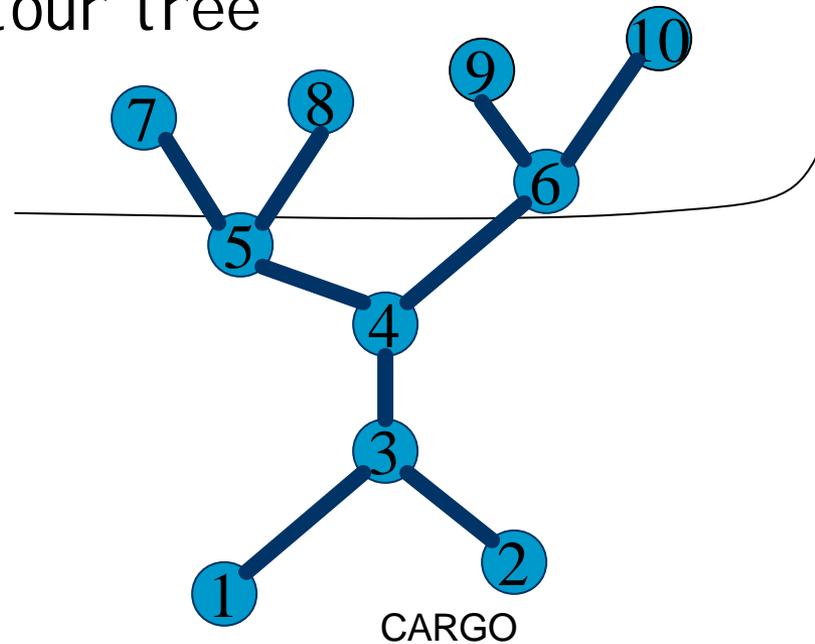
Contour tree



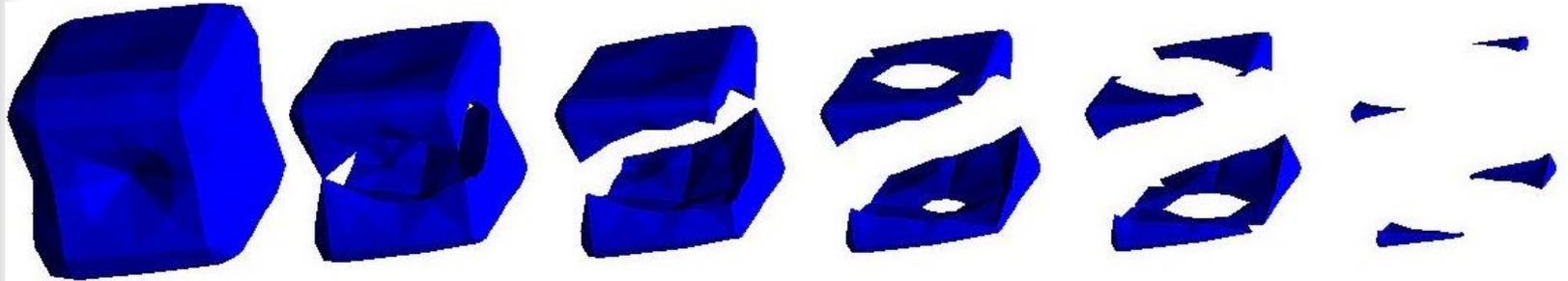
Evolution of level sets



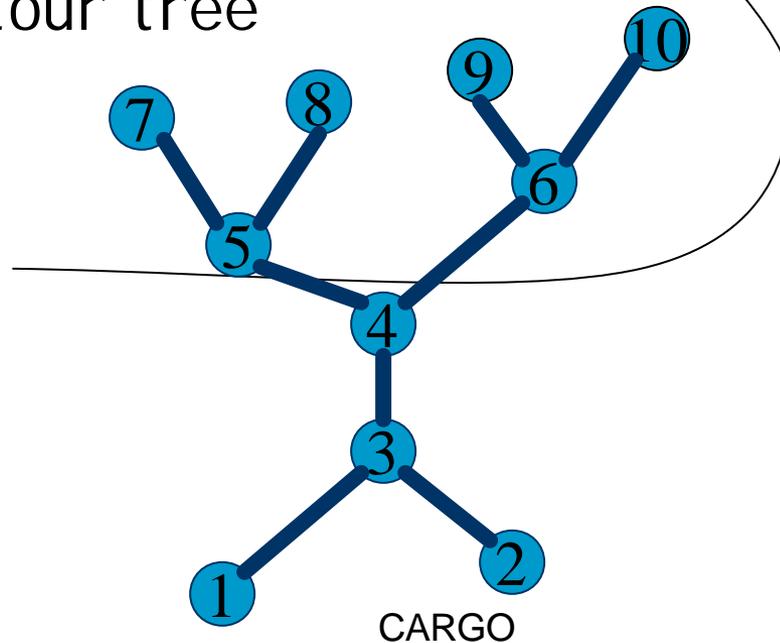
Contour tree



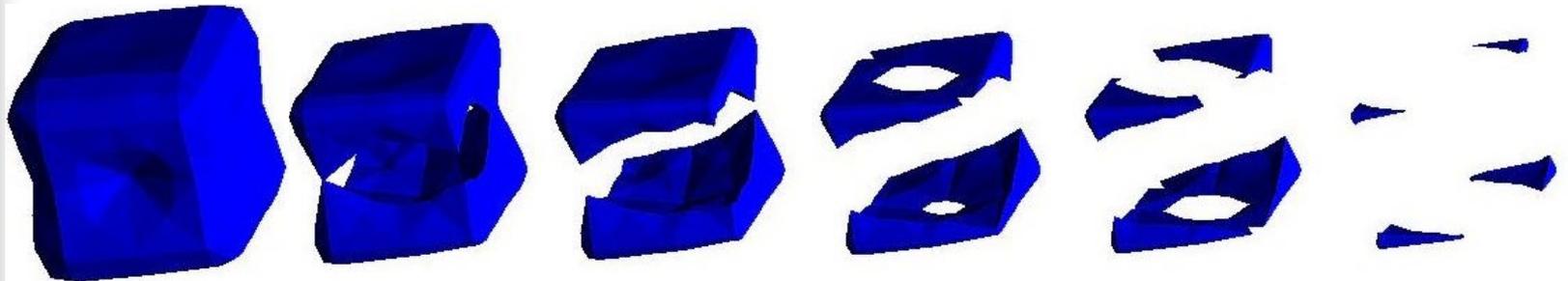
Evolution of level sets



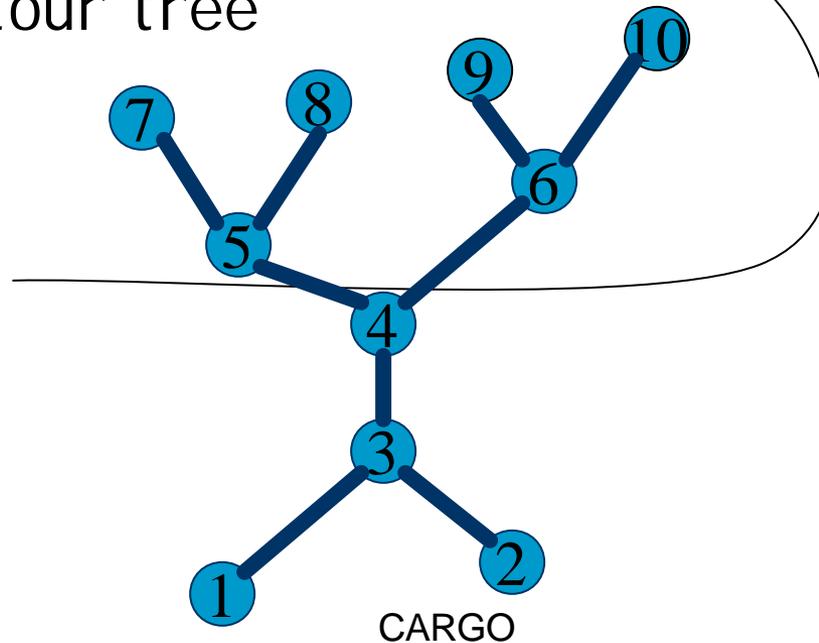
Contour tree



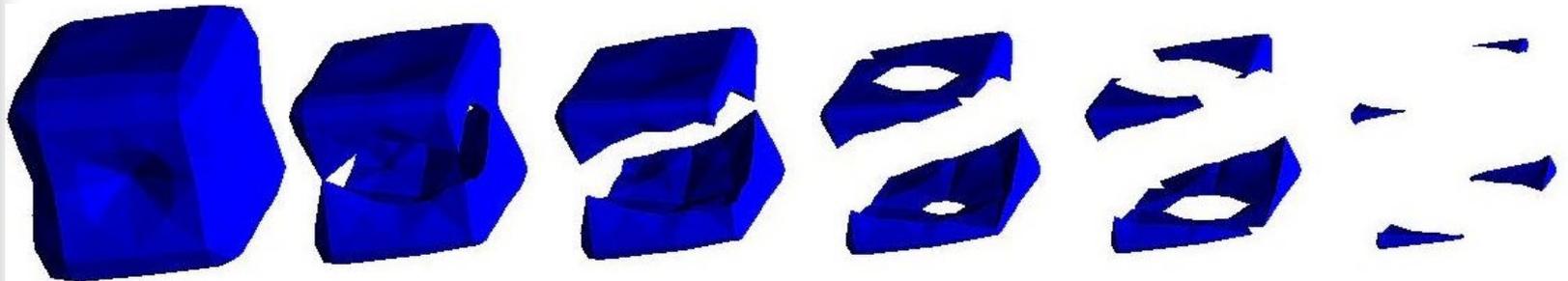
Evolution of level sets



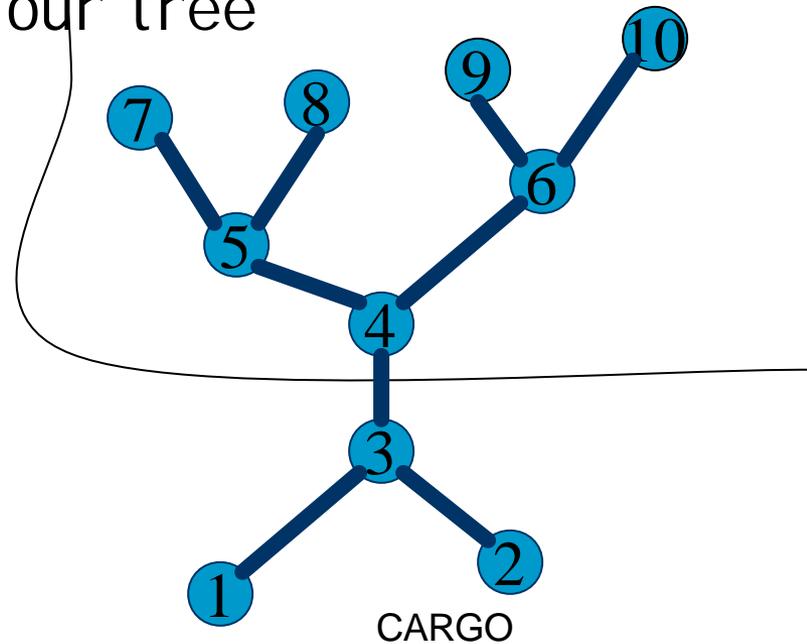
Contour tree



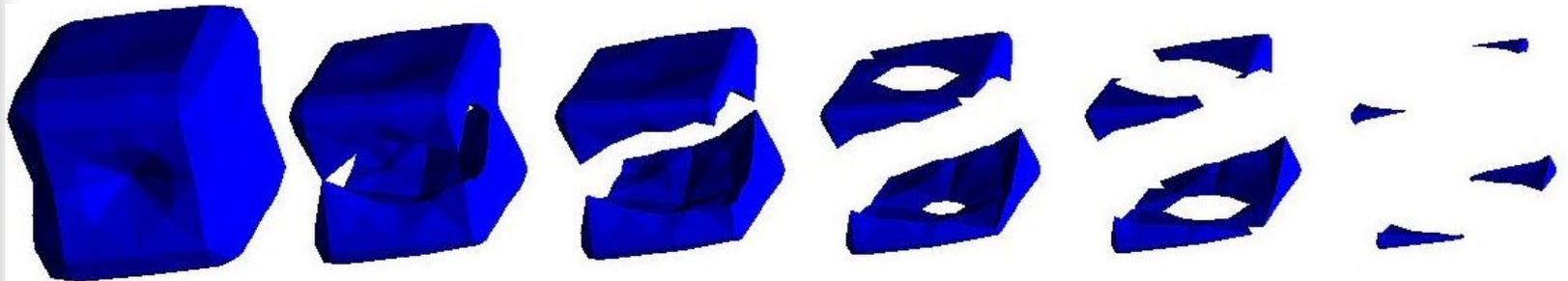
Evolution of level sets



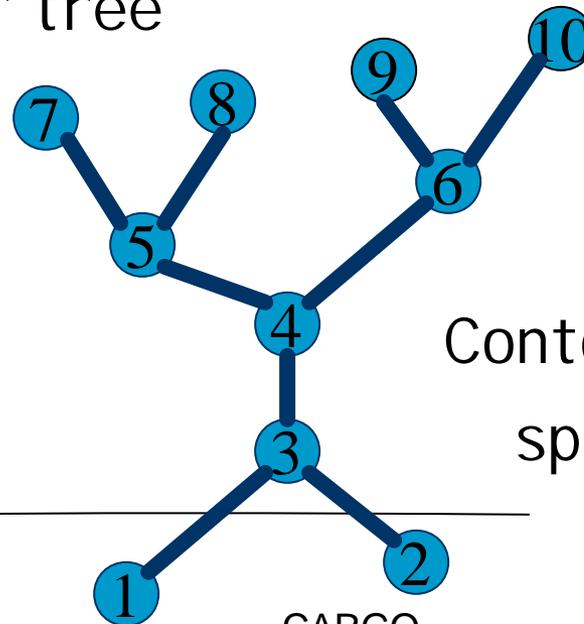
Contour tree



Evolution of level sets



Contour tree



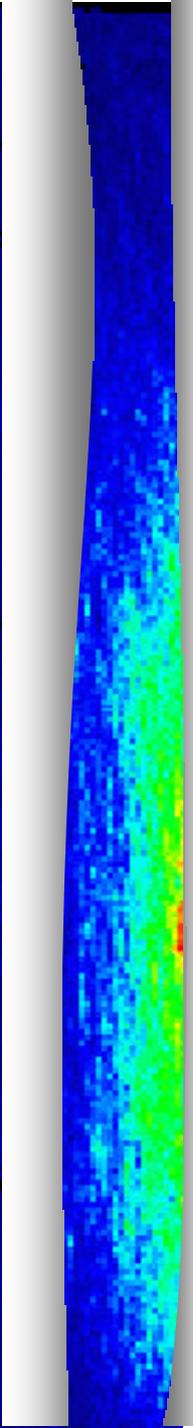
Contours appear, merge,
split, & vanish

May 2002

CARGO

Some references

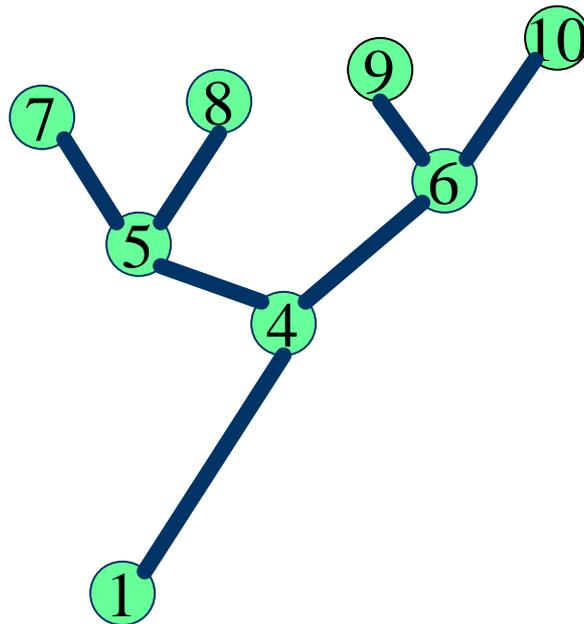
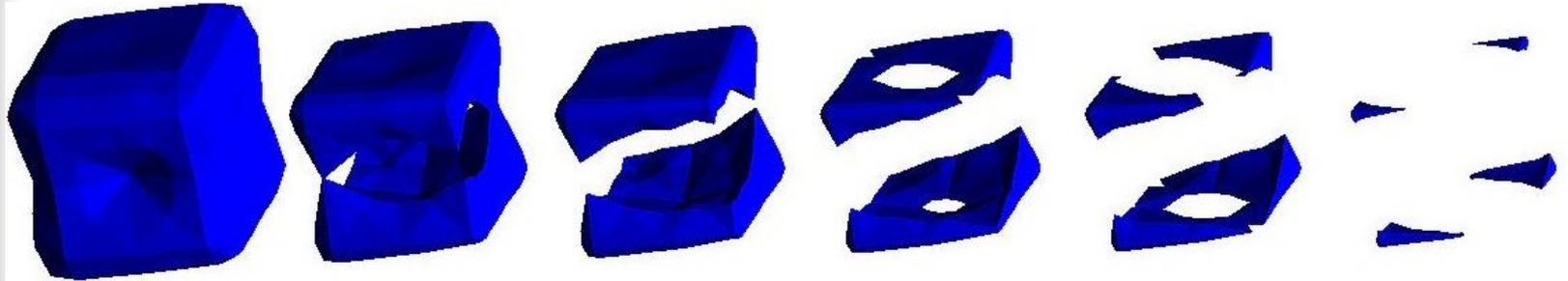
- Lorensen & Cline 87, Marching cubes
- Nielsen & Hamann 91, fix ambiguities
- Shinagawa et al. 91, extreme graph
- Van Gelder & Williams 92, octrees for isosurf.
- Shen & Johnson 96, span space
- Cignoni et al. 97, interval trees
- Van Kreveld et al. 97, 2-D contour tree
- Tarasov & Vyalyi 98, 3-D contour tree
- Carr et al. 00, n -D contour tree, simpler



Building contour trees without constructing all level sets

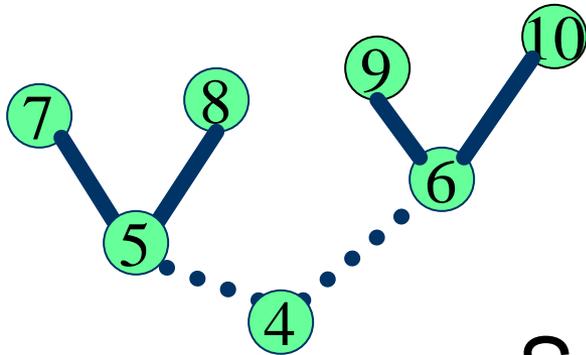
- Build join and split trees
- Merge to form contour tree
- Works in all dimensions...

The join tree



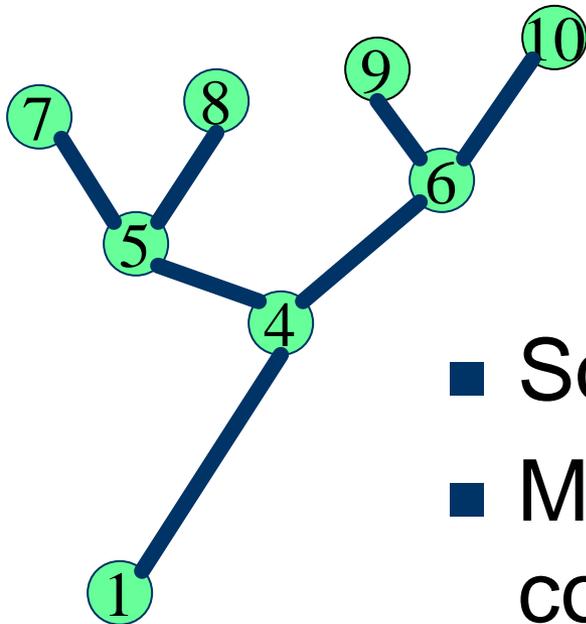
- Represents set $\{p \mid f(p) \geq x\}$
- Can be computed w/out the surface of the level set.

Join tree construction



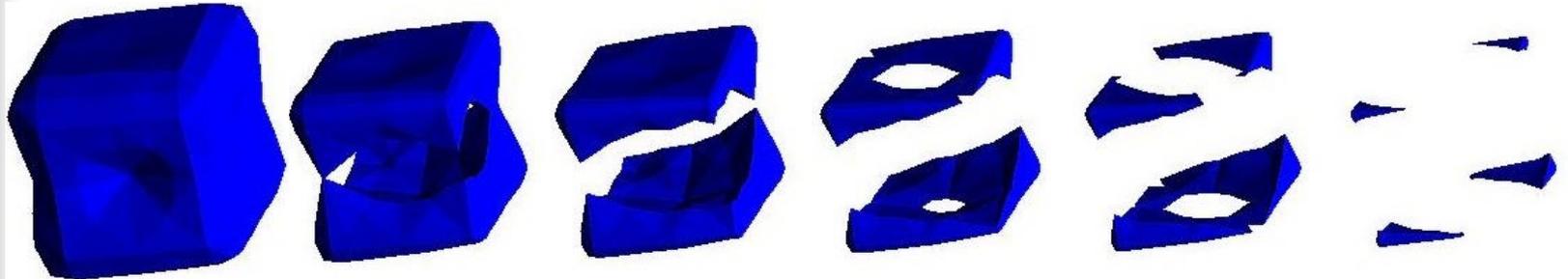
- Sort data by \downarrow value
- Maintain union/find for comp. of $\{p \mid f(p) \geq x\}$

Join tree construction

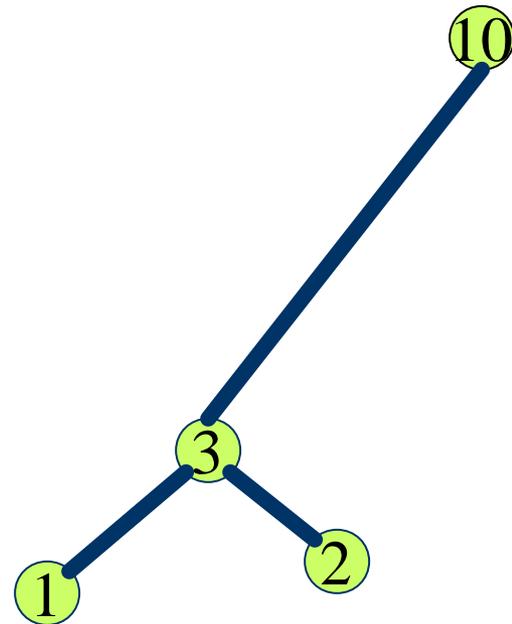


- Sort data by \downarrow value
- Maintain union/find for comp. of $\{p \mid f(p) \geq x\}$
- $O(n + t \cdot a(t))$ after sort

The split tree

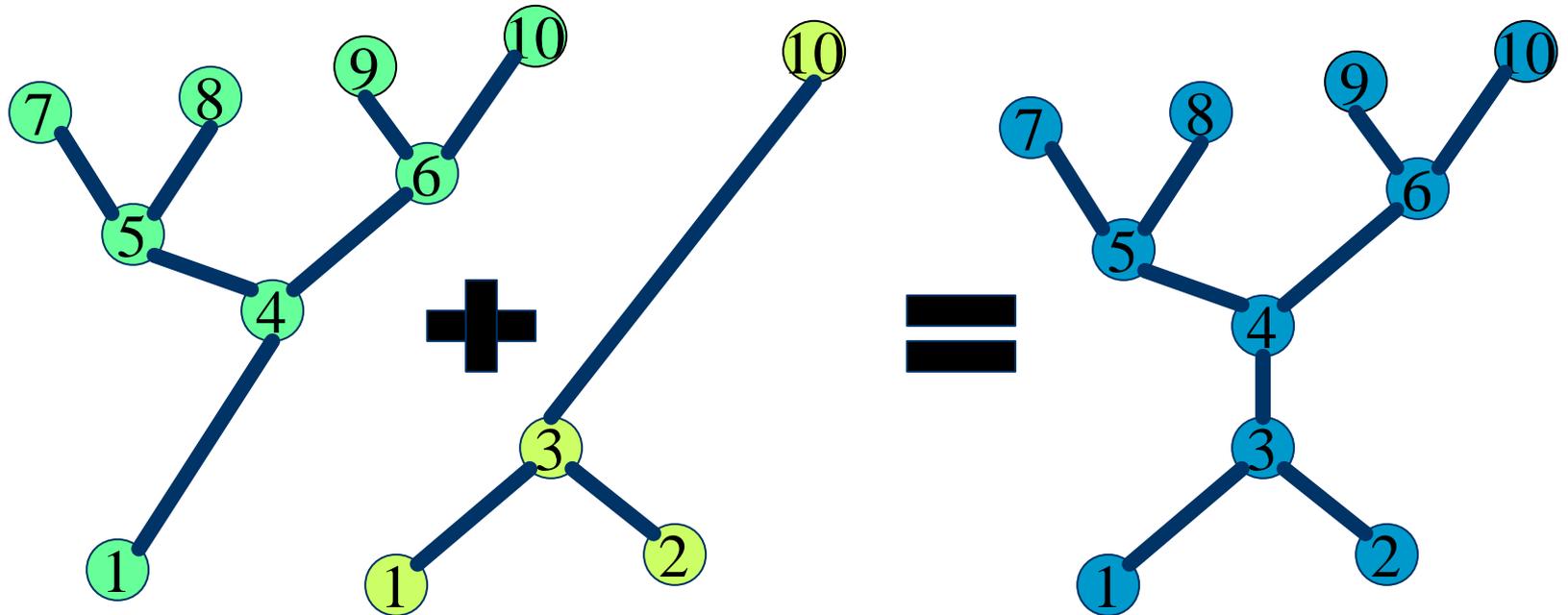


- Represents the set $\{p \mid f(p) \leq x\}$
- Reverse join tree computation



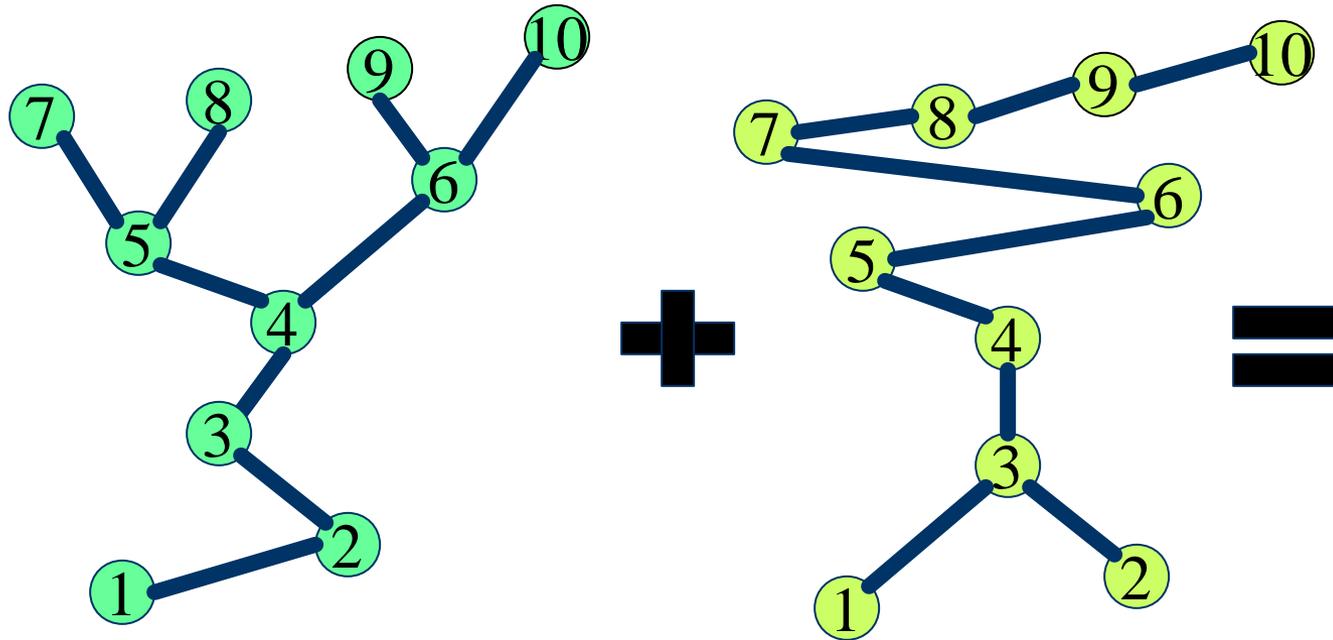
CARGO

Merge join and split trees



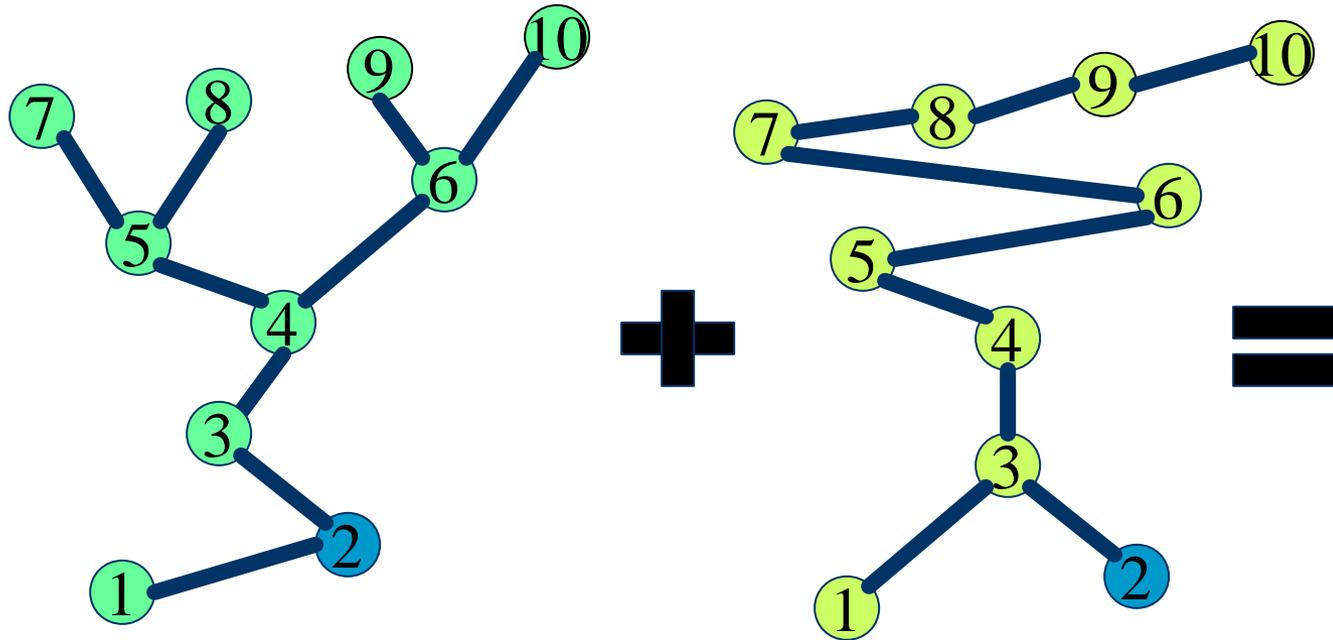
- A four step process...

1. Add nodes from other tree



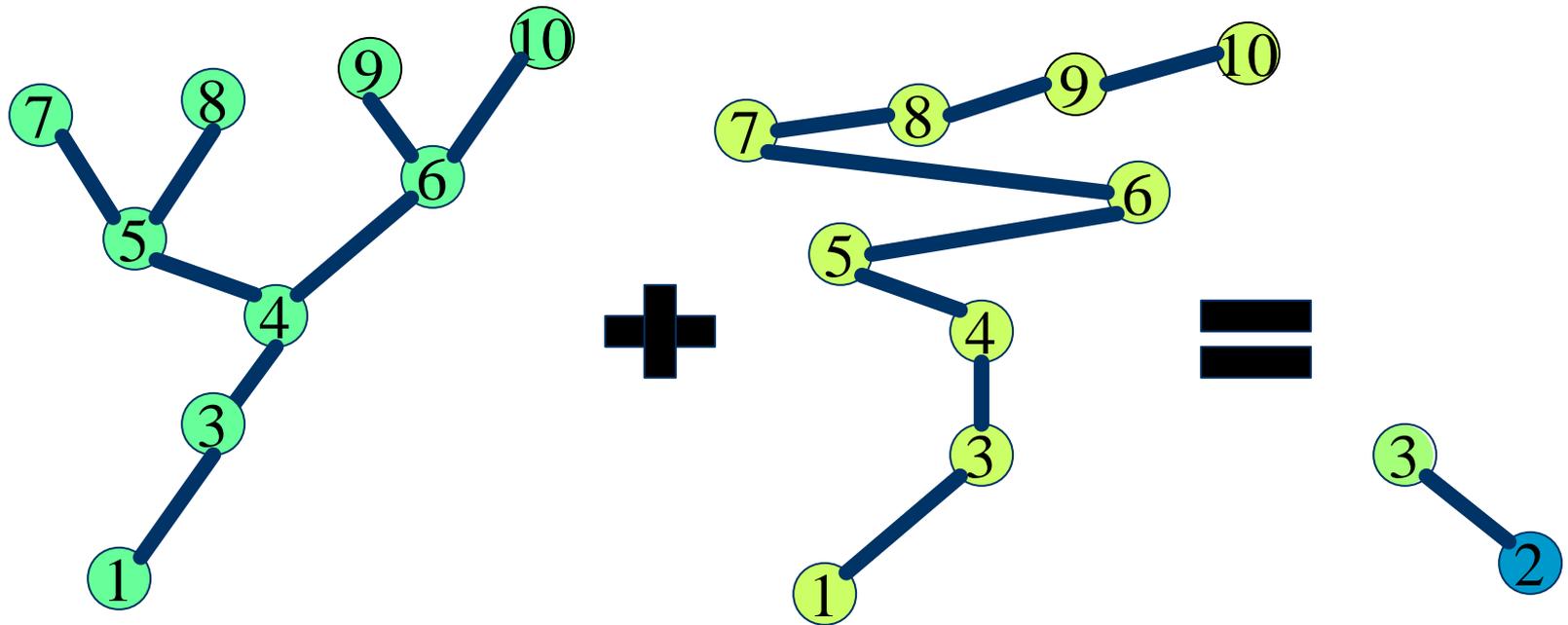
- Nodes have correct up degree in the join tree & correct down degree in the split tree.

2. I identify leaves



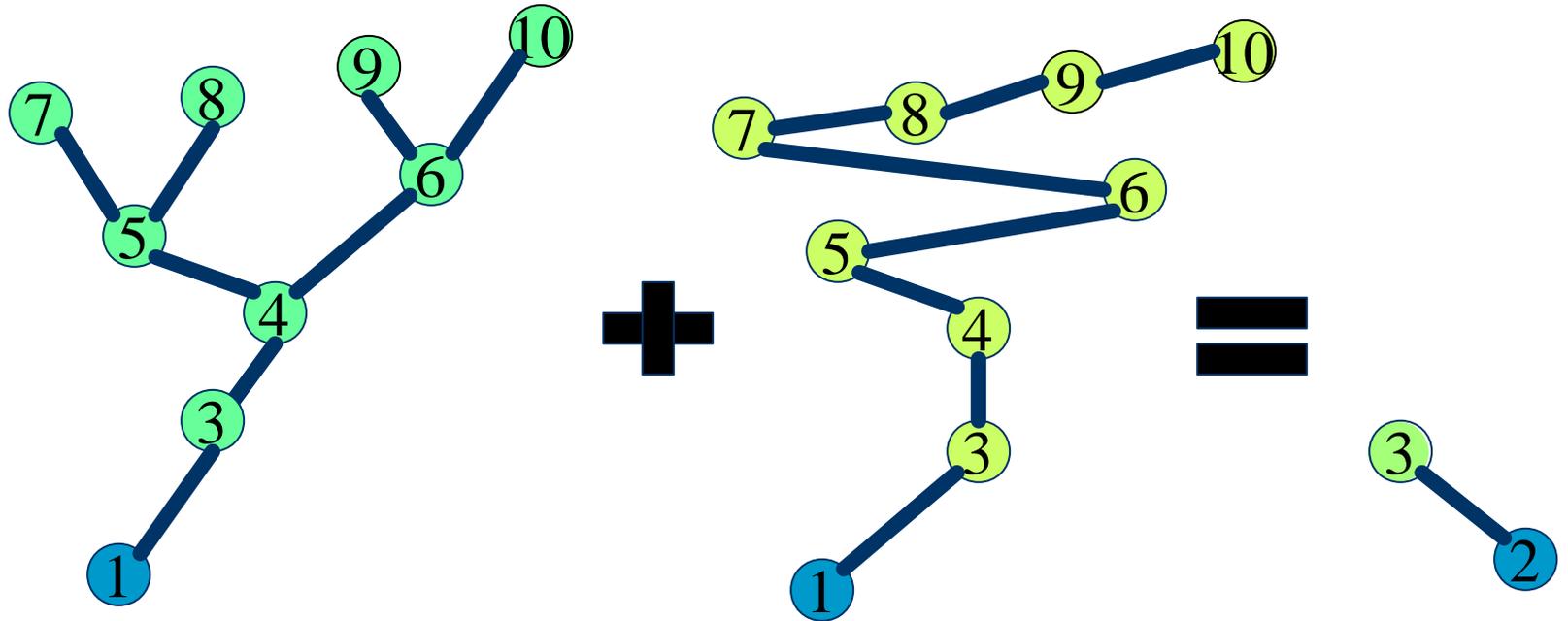
- A leaf in one tree, with up/down degree ≤ 1 in other tree

3. Add to contour tree

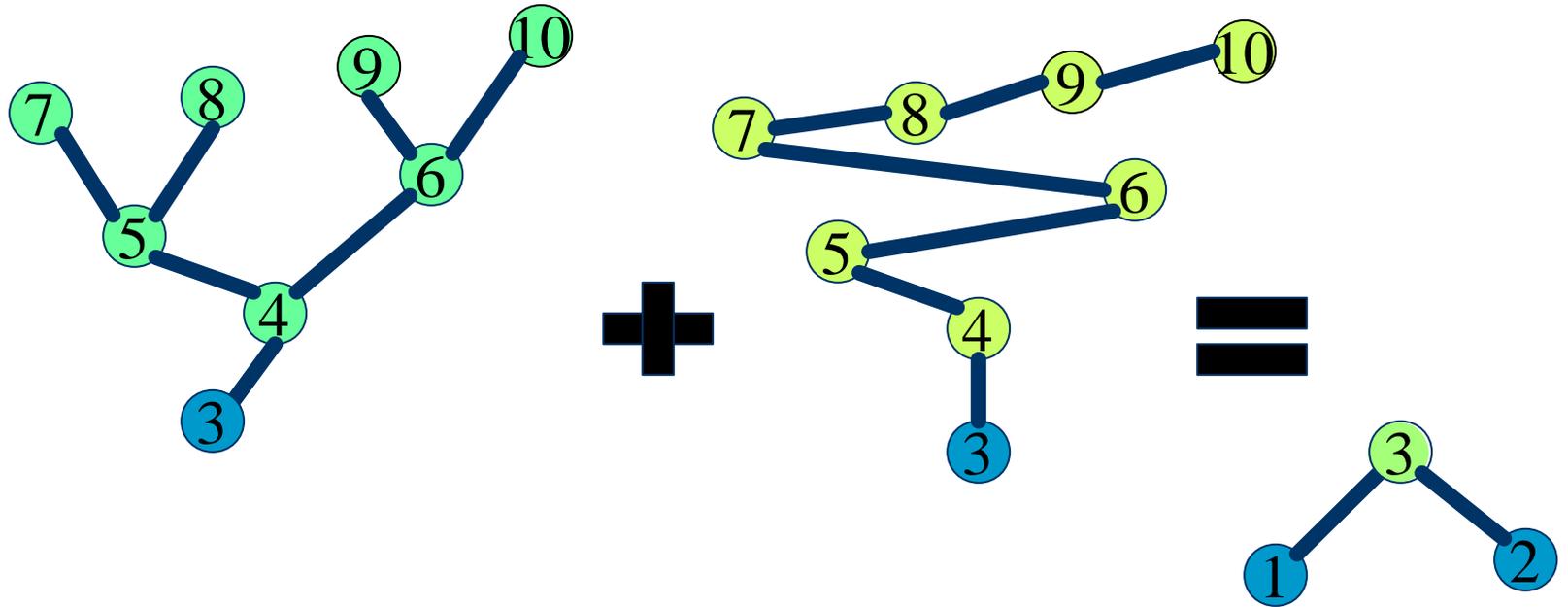


- A leaf in one tree, with up/down degree ≤ 1 in other tree

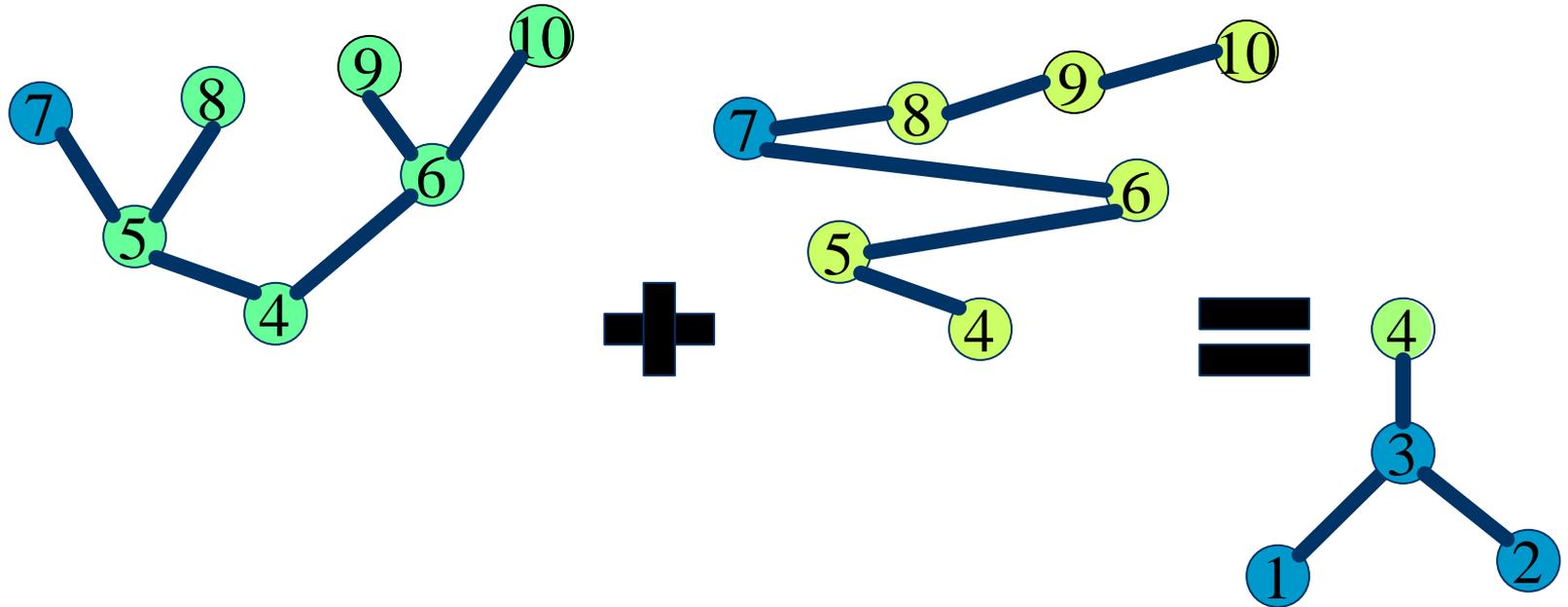
3. Add to contour tree



3. Add to contour tree

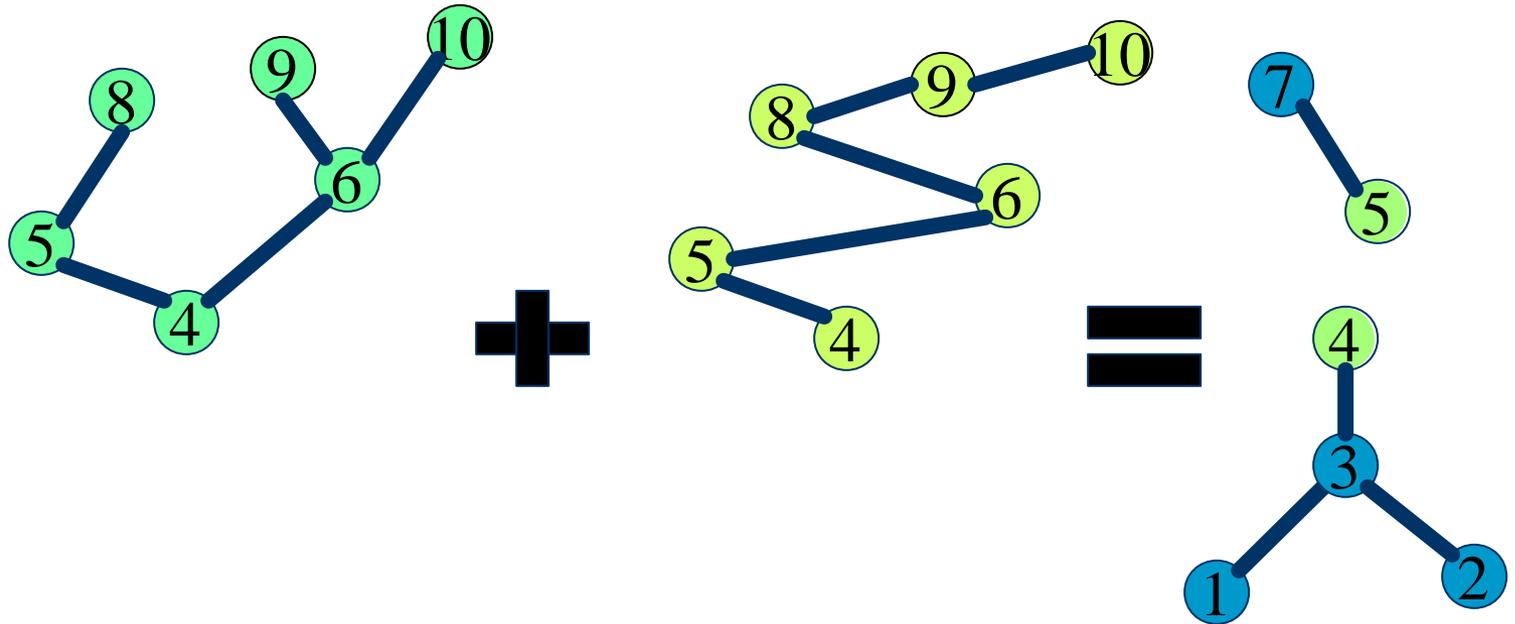


3. Add to contour tree



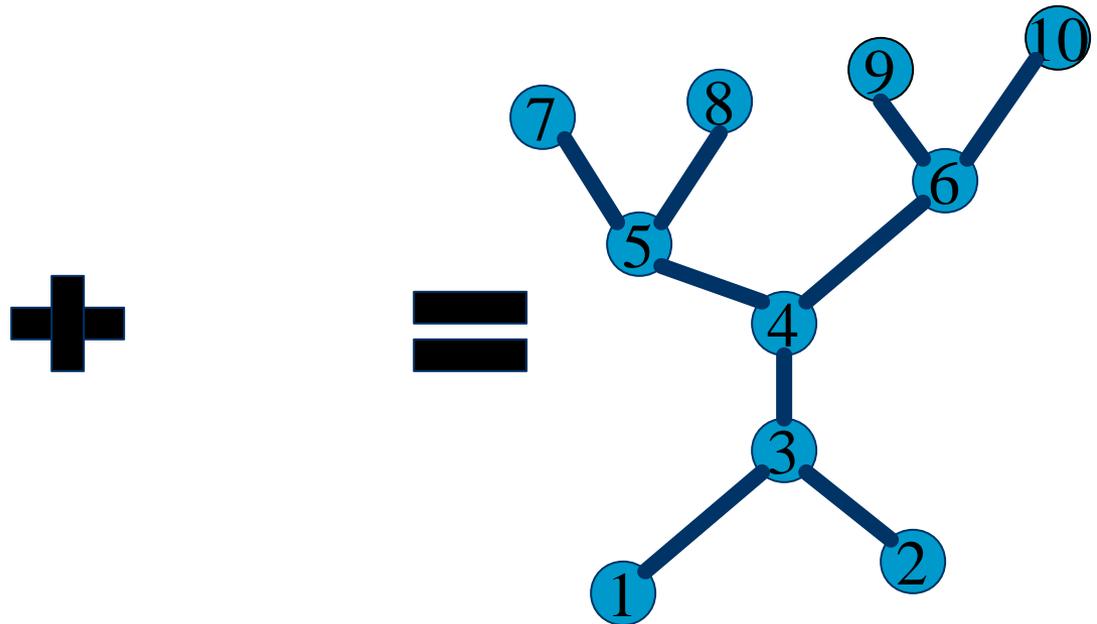
- Note that 4 can not be added, but any of 7, 8, 9, or 10 can.

3. Add to contour tree



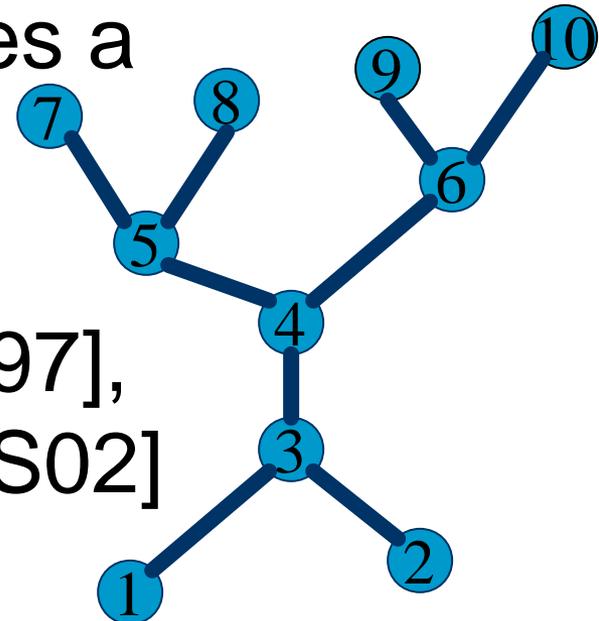
- The merge time is proportional to the tree size; much smaller than the data

4. Ends with Contour tree



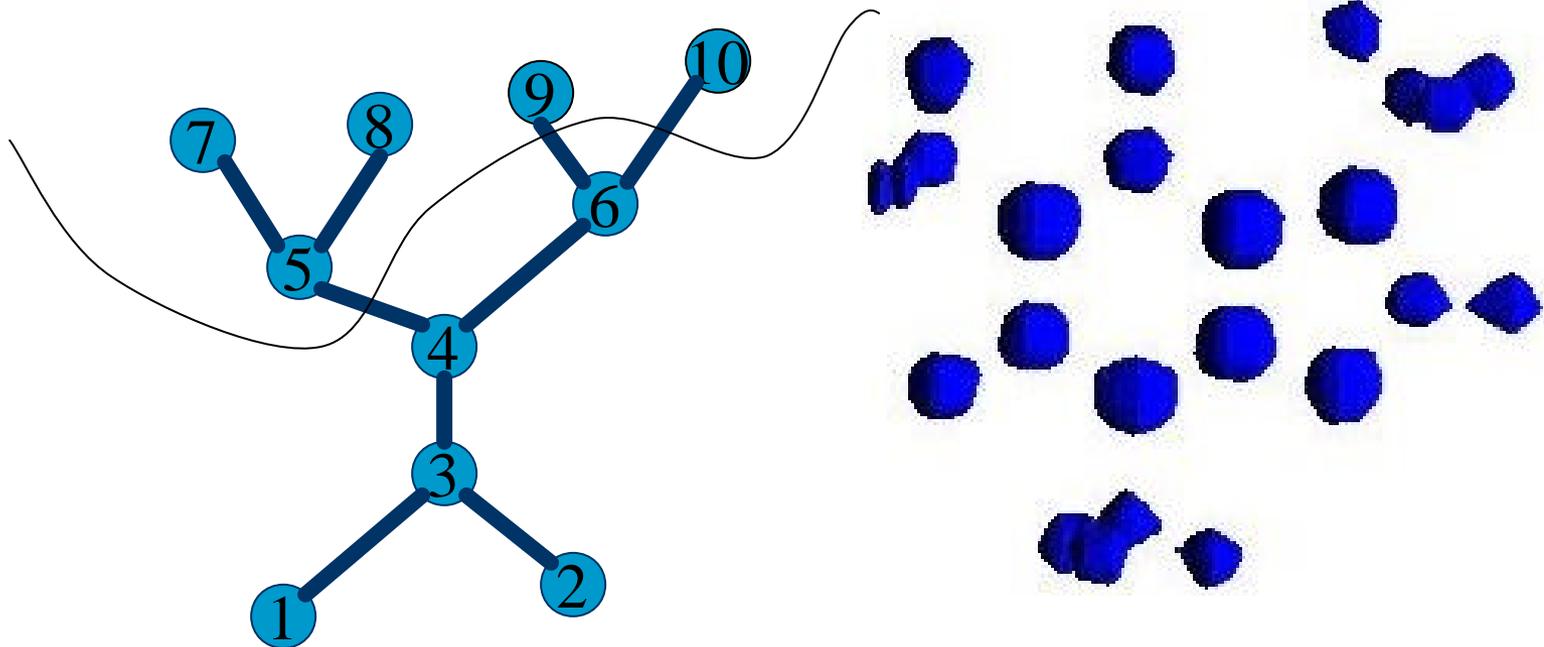
Advantages

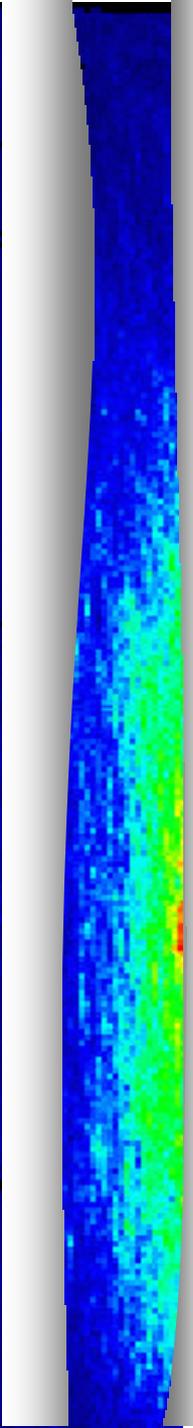
- Doesn't need iso-surface to build contour tree
- Merge is simple & uses a subset of points
- Tree gives seed sets, contour spectra [BPS97], & flexible contours [CS02]



Flexible contouring

- Threshold diff. areas at diff. values
- Guided by the Contour Tree

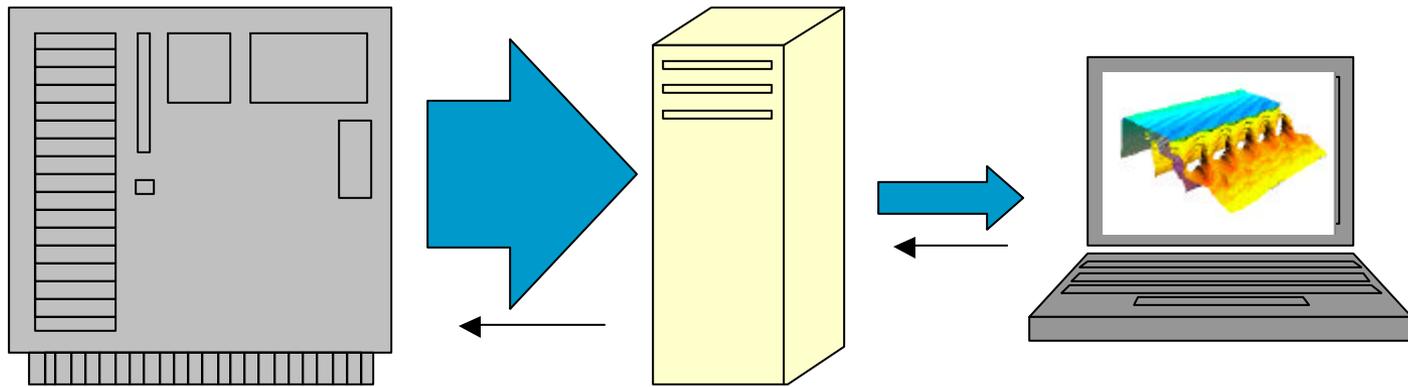




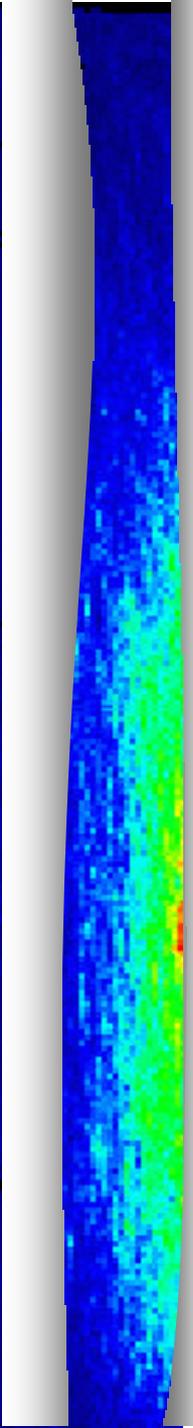
Applying computational topology

- Illustrating our thesis: *contour trees in 3D*
- Time-varying volume visualization
 - Sample data sets: *scientific & eng. simulation*
 - Prototype interface: *contour spectra & iso-surfaces*
- Pentatope meshes
 - A data structure supporting iso-surfaces
 - Implementing the incremental flip algorithm
 - Arithmetic complexity
 - Handling degeneracies

4-D Simulation Data Sets



What tools can we provide to suggest
when & where to look
in large simulation data sets?

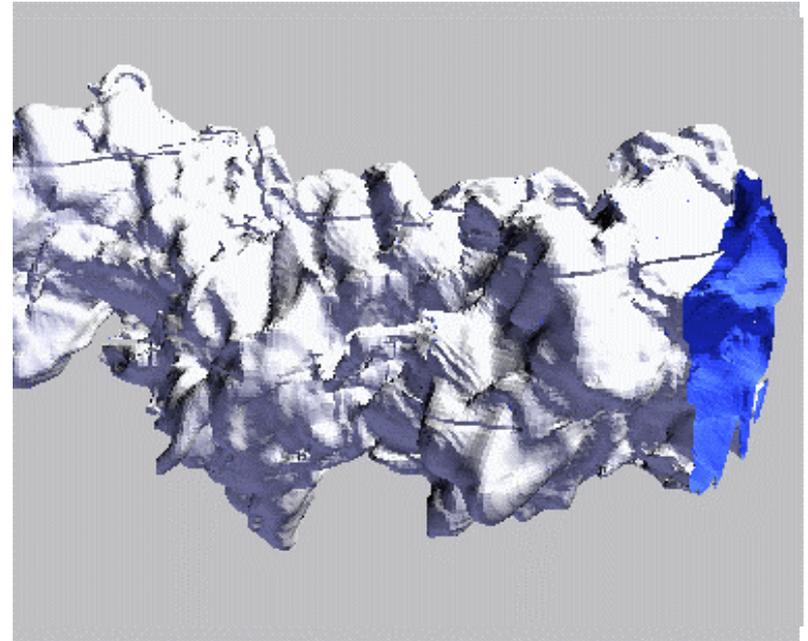


4-D Simulation Data Sets

- Samples from pressure $p = f(x, y, z, t)$.
- Level sets defined by two parameters:
$$L(P, T) = \{ (x, y, z) : P = f(x, y, z, T) \}$$
- Consider some examples...

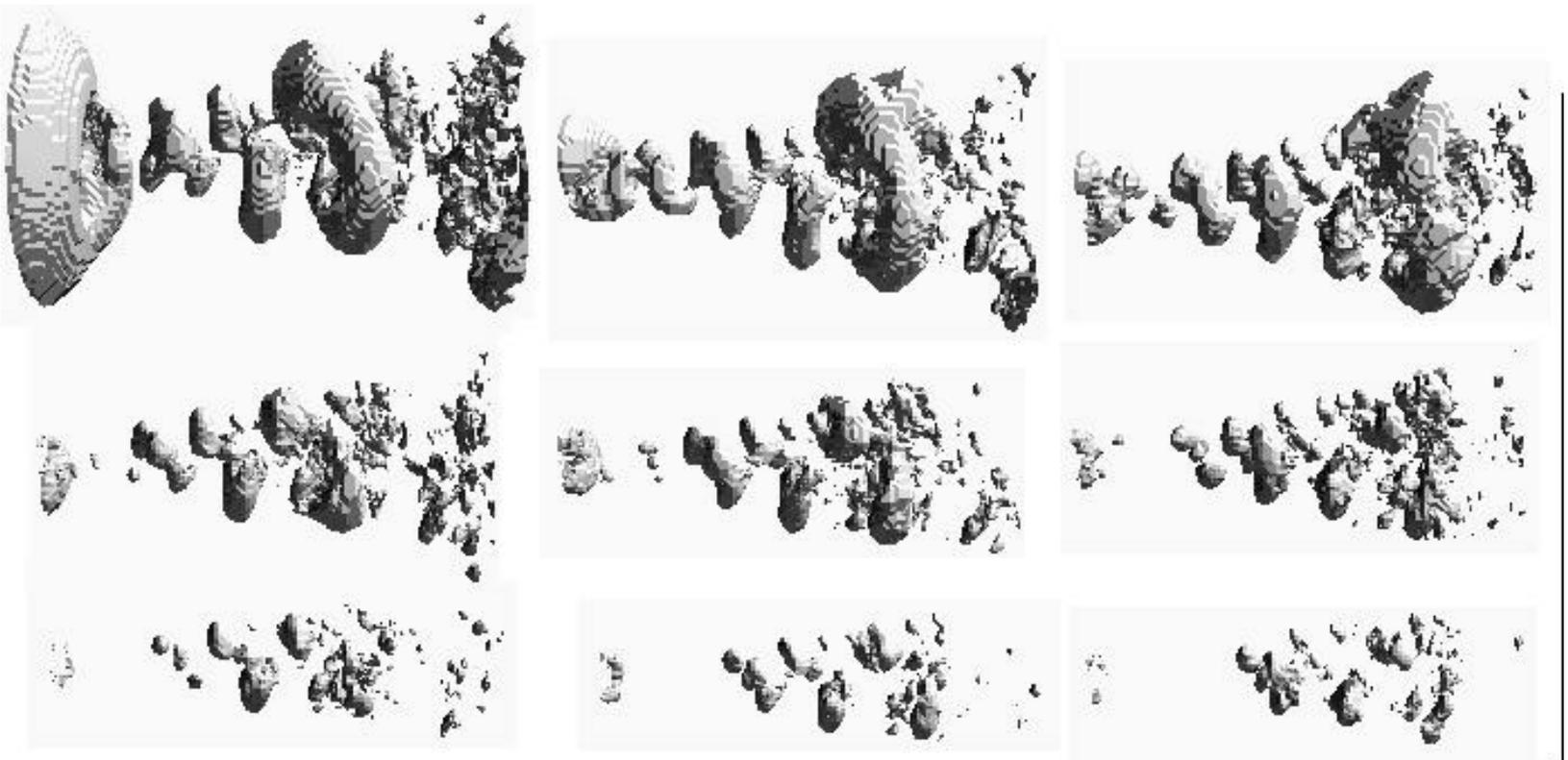
Supersonic flow

- Introduced into still medium to study Helmholtz instability
- Density of tracer
- 256 x256 x256 x100
- 1680 MB



Pulsed flow

- $104 \times 129 \times 129 \times 150 = 1040 \text{ MB}$

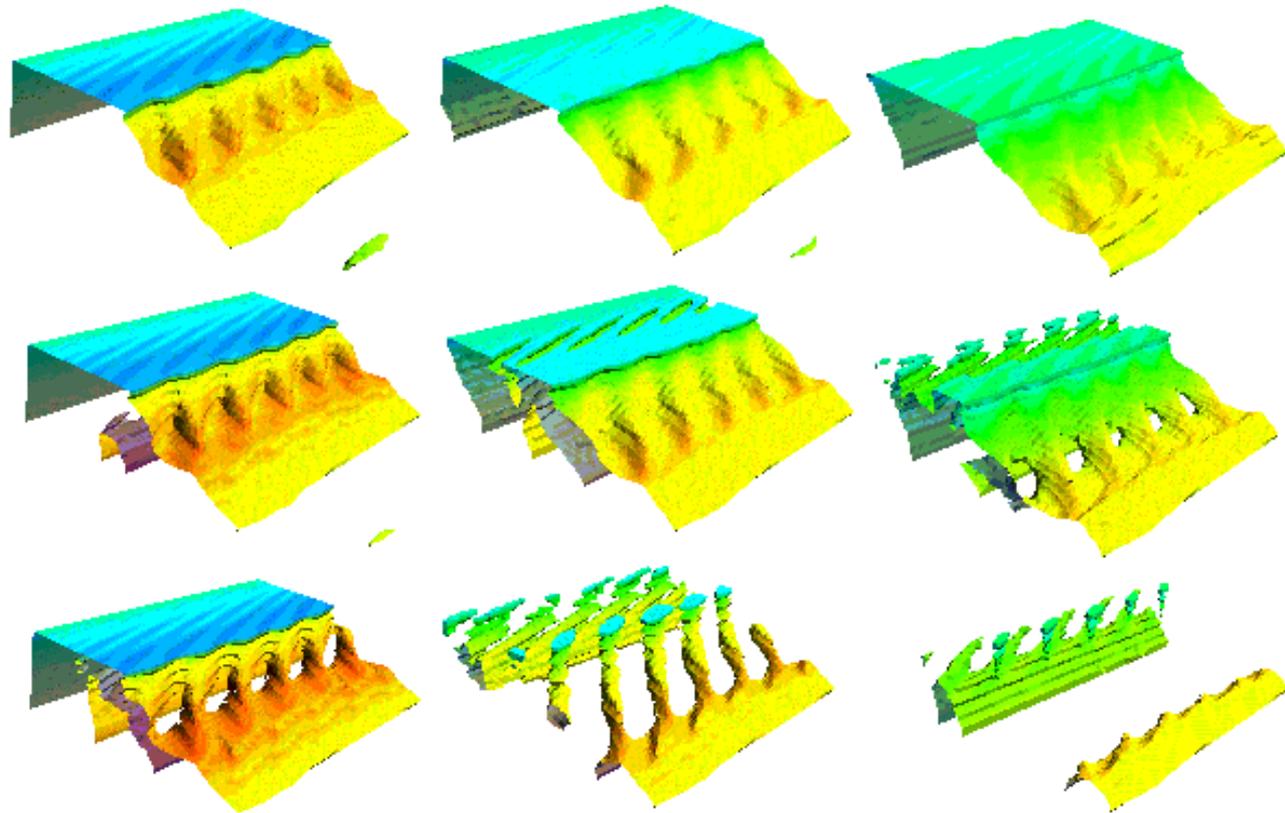


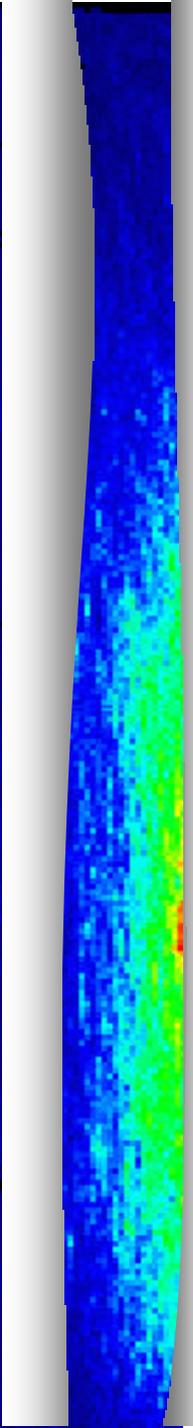
May 2002

CARGO

Combustion simulation

- Suresh Menon
GATech
- 110x64x81
x 20 x 7
- 320 MB

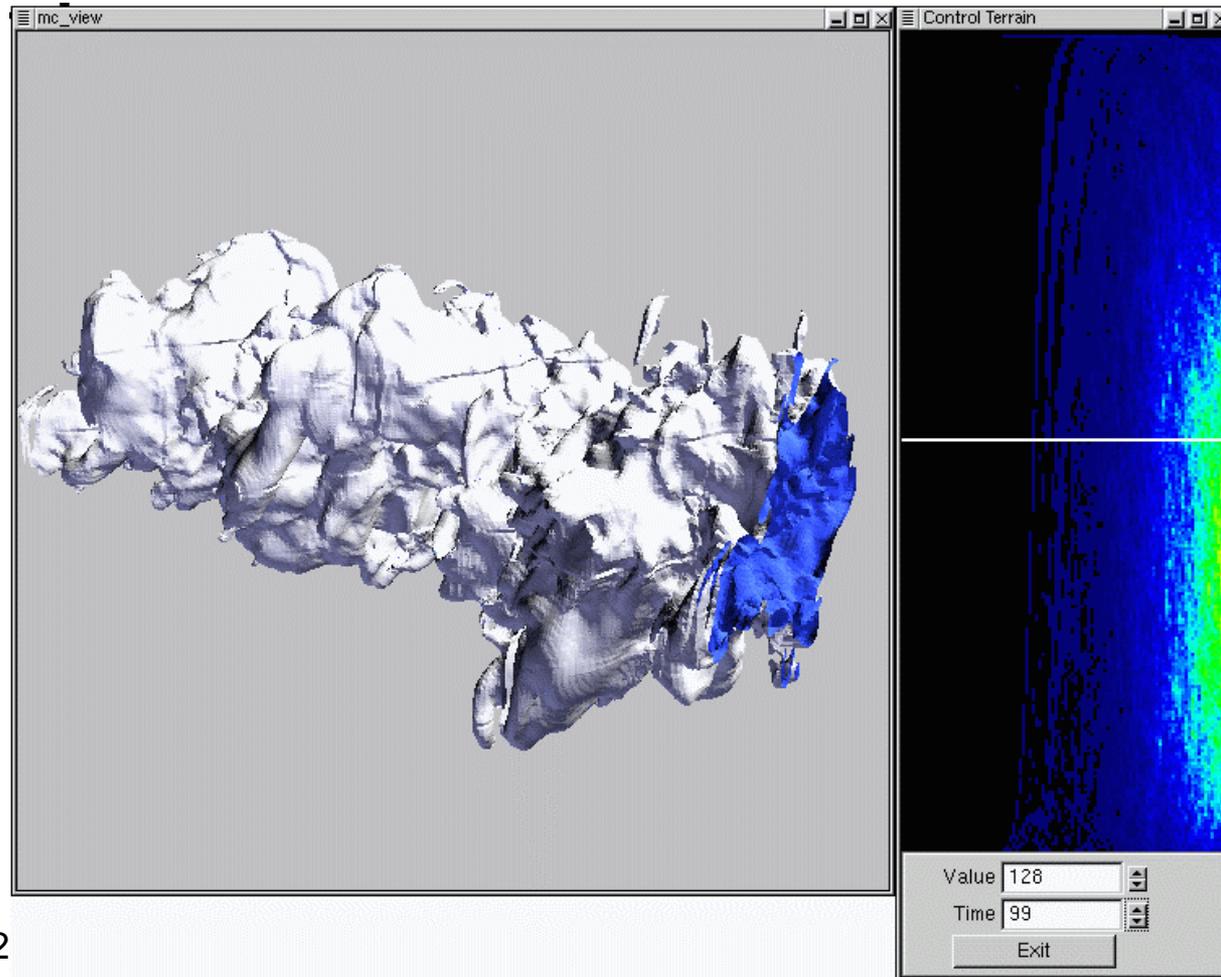




4-D Simulation Data Sets

- Samples from pressure $p = f(x, y, z, t)$.
- Level sets defined by two parameters:
$$L(P, T) = \{ (x, y, z) : P = f(x, y, z, T) \}$$
- Partition the data dimensions
 - Viewing volume (x, y, z)
 - Control plane (p, t)

Viewing volume & control plane

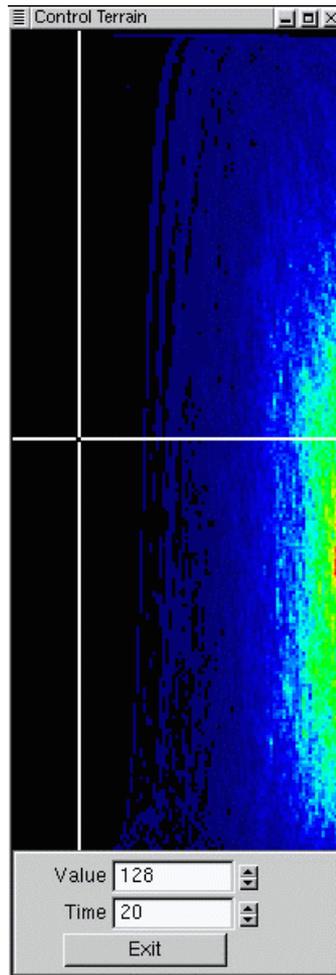


May 2

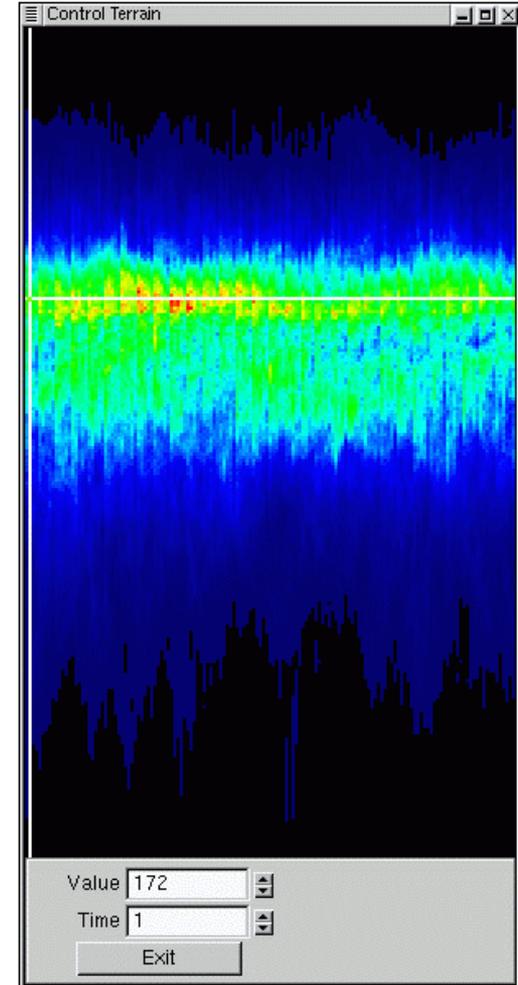
Comparing control planes

⇐ Supersonic
flow data set

Pulsed flow
data set ⇒

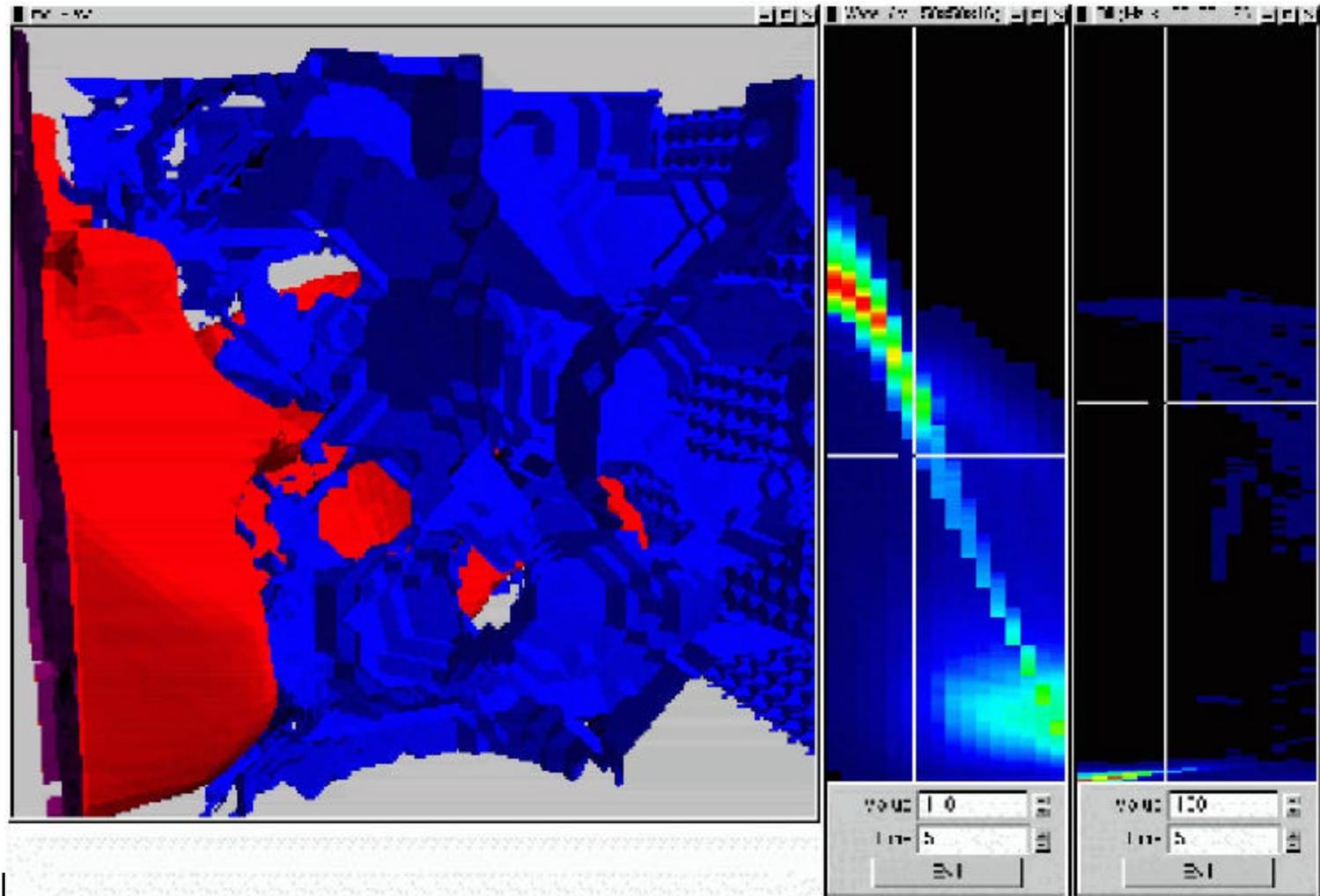


May 2002



CARGO

Water and Oil in gravel soil

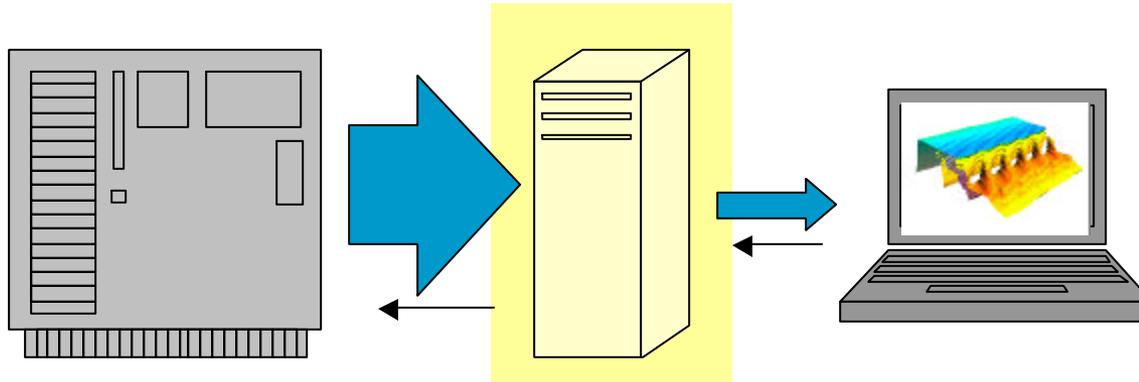


Video of Example Data Sets

May 2002

CARGO

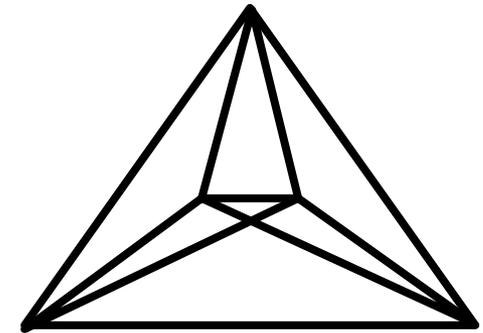
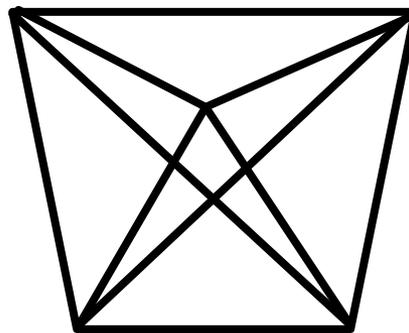
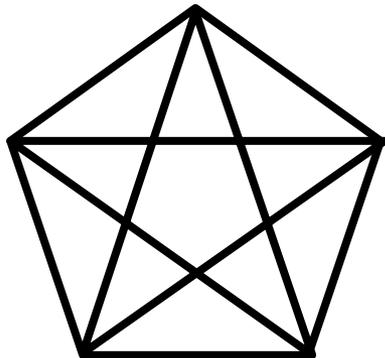
A geometric basis: pentamesh



- Mesh domain with pentatopes (4-simplices)
 - Expensive, but supports refinement
- Isosurfaces from pentatope meshes
 - graph structure [network topology]
 - connectivity summary [$\mathbf{b}_o(t,p)$]
 - contour following [seeds(t,p)]

Isosurfaces from pentatopes

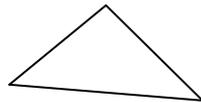
- Mesh of 5-simplices are 4d surface in 5d:
 - Pentatopes (4d)
 - Tetrahedra (3d)
 - Triangles (2d)
- Slice away 2 dimens to form isosurface:
 - Faces (2d)
 - Edges (1d)
 - Vertices (0d)



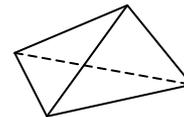
Yes, this is crazy...

- Rectilinear mesh stores values only
- Simplicial stores coords & neighbors

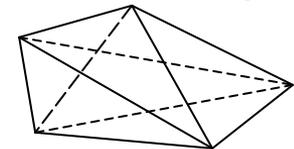
2d: Triangle



3d: Tetrahedron



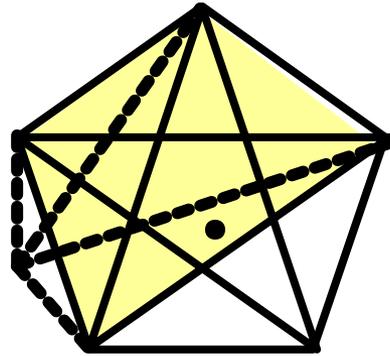
4d: Pentatope



Simp/vertex	2	8	30
Words/vertex	15	51	244

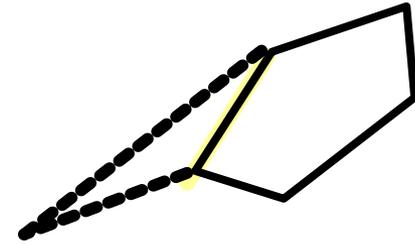
- Allows local refinement, so we try anyway...

Mesh Topology



Pentamesh

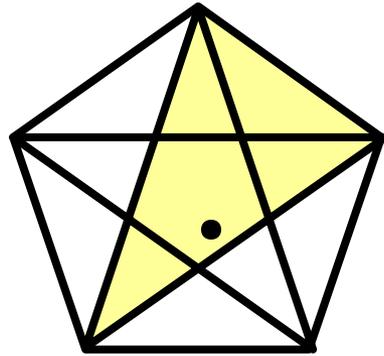
- each tetrahedron bounds two adj. pentatopes
- each triangle is in two tetrahedra per pentatope
- each triangle has a cycle of pentatopes, each sharing a tetrahedron with the next pentatope



Isosurface

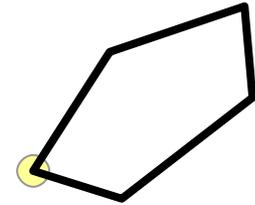
- each edge bounds two adjacent polygons
- each vertex is in two edges per polygon
- each vertex has a cycle of polygons, each sharing an edge with the next polygon

Mesh Topology



Pentamesh

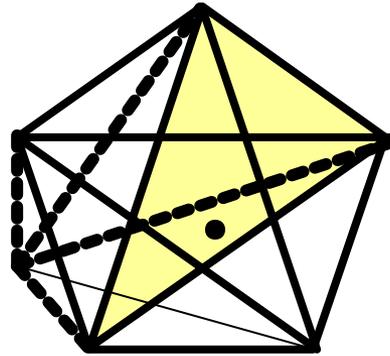
- each tetrahedron bounds two adj. pentatopes
- each triangle is in two tetrahedra per pentatope
- each triangle has a cycle of pentatopes, each sharing a tetrahedron with the next pentatope



Isosurface

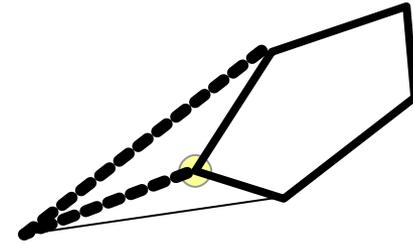
- each edge bounds two adjacent polygons
- each vertex is in two edges per polygon
- each vertex has a cycle of polygons, each sharing an edge with the next polygon

Mesh Topology



Pentamesh

- each tetrahedron bounds two adj. pentatopes
- each triangle is in two tetrahedra per pentatope
- each triangle has a cycle of pentatopes, each sharing a tetrahedron with the next pentatope



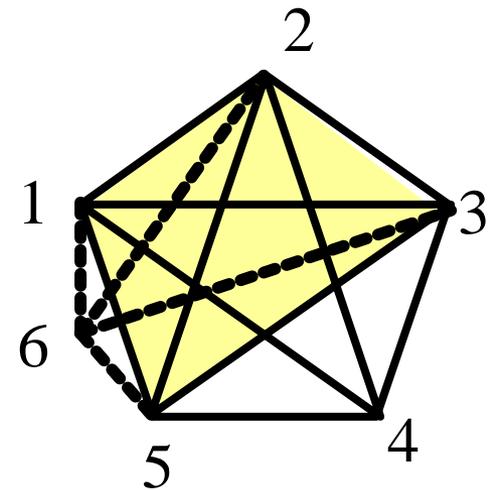
Isosurface

- each edge bounds two adjacent polygons
- each vertex is in two edges per polygon
- each vertex has a cycle of polygons, each sharing an edge with the next polygon

Corner table data structure

Don't need facial lattice

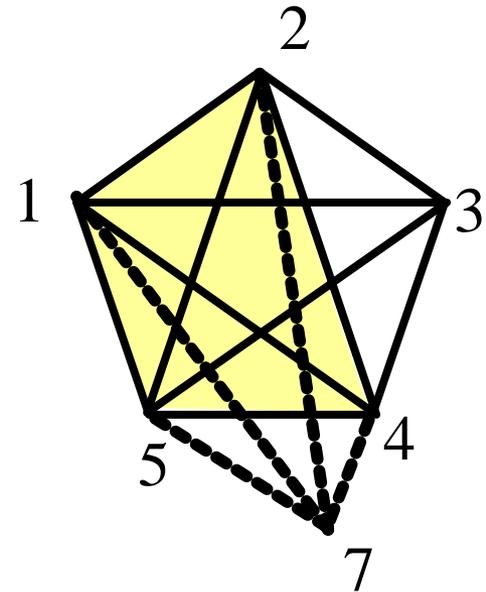
- Pentatope stored in table as block of 5 corners in lex-min order having positive orientation
- Store *opposite(corner)* & *vertex(corner)* pointers
 $opposite(12345) = 12356;$
 $opposite(12356) = 12345;$
- Data structure supports walk around fixed triangle without search



Corner table data structure

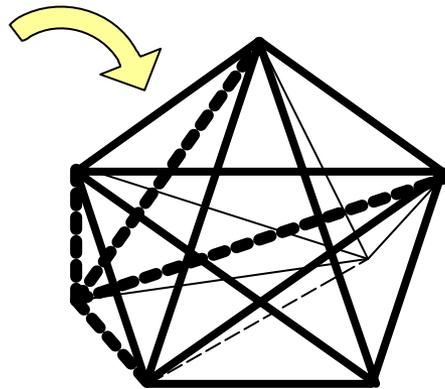
Don't need facial lattice

- Pentatope stored in table as block of 5 corners in lex-min order having positive orientation
- Store *opposite(corner)* & *vertex(corner)* pointers
 $opposite(12345) = 21357$;
 $opposite(21357) = 12345$;
- Data structure supports walk around fixed triangle without search

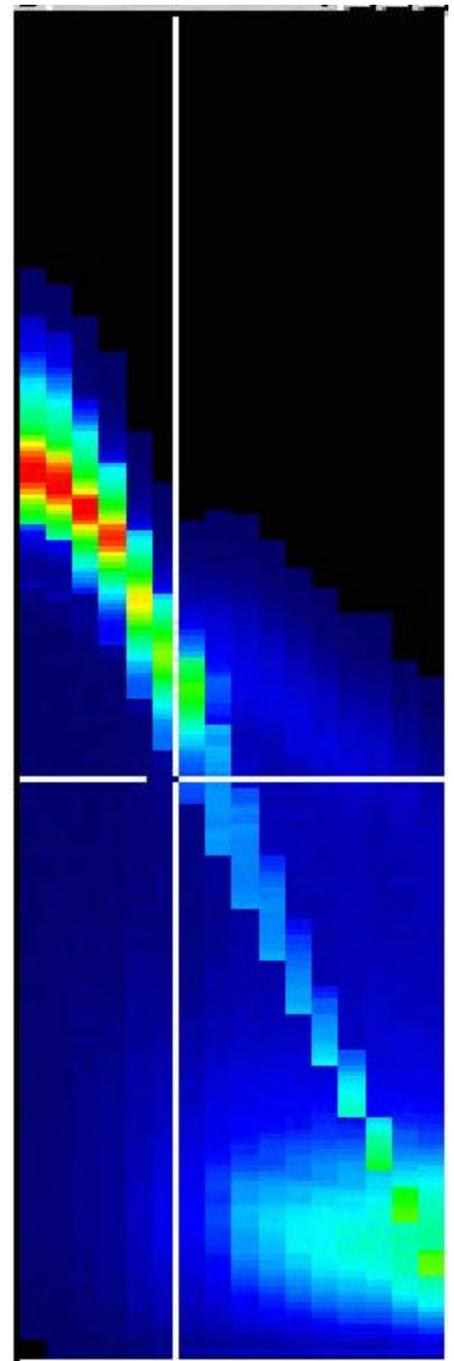


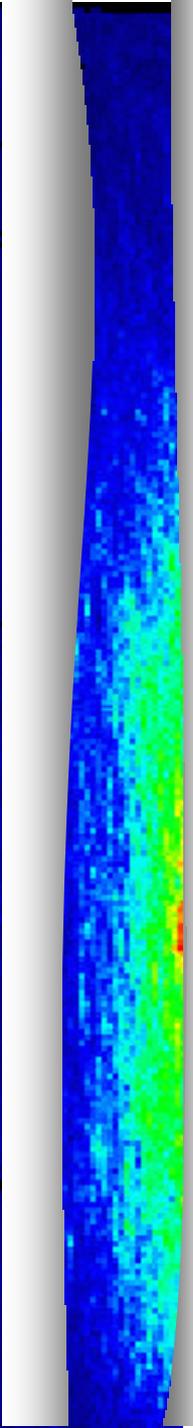
Counting contours [b_0]

- Project silhouette edges onto (p, t) control plane



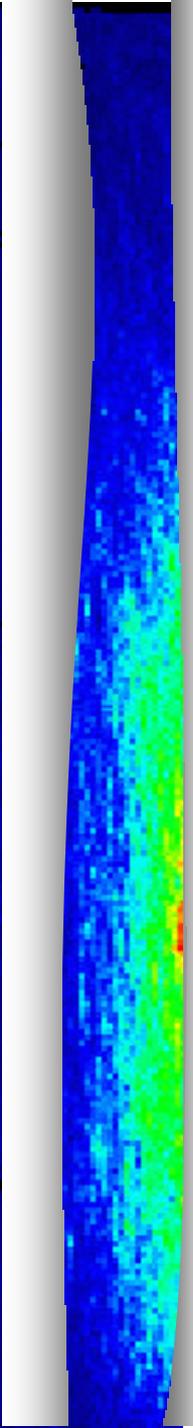
- Color the arrangement





Evolution of contour trees?

- Would like a family of contour trees parameterized by time t .
- Can find families of join & split trees.
- Morse theory with two parameters?

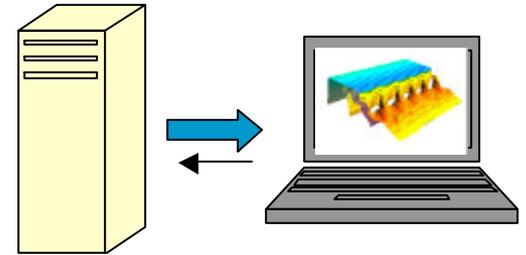


Applying computational topology

- Illustrating our thesis: *contour trees in 3D*
- Time-varying volume visualization
 - Sample data sets: *scientific & eng. simulation*
 - Prototype interface: *contour spectra & iso-surfaces*
- Pentatope meshes
 - A data structure supporting iso-surfaces
 - Implementing the incremental flip algorithm
 - Arithmetic complexity
 - Handling degeneracies

A "Safari" scenario

<http://www.cs.unc.edu/~compgeom>



1. Compute server outputs (hierarchical) pentamesh
2. Visualization server computes control plane image
3. User chooses (p, t)
4. Vis server sends initial isosurface/pentatopes
5. User views isosurface and moves to nearby (p', t')
6. Vis server sends updates to isosurf/pentas
7. User rotates or zooms current isosurface
8. Vis server sends refinements to isosurf/pentas
9. User continues with 3, 4, or 7.

Computing pentatope meshes

- Corner-based data structure
- Compute 4d Delaunay by incremental algorithm of Edelsbrunner and Shah [ES96]
- Exact arithmetic in IEEE doubles [P91]
- Degeneracy handling extending [ADS99]
- Code to be released when validation is complete

Incremental Delaunay [ES96]

- Start with an empty mesh M
- Add points $p_1 \dots p_n$ in order, update M to preserve the Empty Sphere property
- Flip operations suffice for the update

Empty Sphere determinant

- Delaunay: Every simplex has an Empty Sphere property w.r.t. neighbor points

$$\text{InSphere}(abcdep) = \begin{vmatrix} 1 & a_x & a_y & a_z & a_t & a \cdot a \\ 1 & b_x & b_y & b_z & b_t & b \cdot b \\ 1 & c_x & c_y & c_z & c_t & c \cdot c \\ 1 & d_x & d_y & d_z & d_t & d \cdot d \\ 1 & e_x & e_y & e_z & e_t & e \cdot e \\ 1 & p_x & p_y & p_z & p_t & p \cdot p \end{vmatrix} < 0$$

Arithmetic complexity

- InSphere is a dot product with plane tests we must do anyway...
- 6-fold precision required, but only here
- 4-fold precision sufficient everywhere else

$$\text{InSphere}(abcdep) = \begin{vmatrix} 1 & a_x & a_y & a_z & a_t & a \cdot a \\ 1 & b_x & b_y & b_z & b_t & b \cdot b \\ 1 & c_x & c_y & c_z & c_t & c \cdot c \\ 1 & d_x & d_y & d_z & d_t & d \cdot d \\ 1 & e_x & e_y & e_z & e_t & e \cdot e \\ 1 & p_x & p_y & p_z & p_t & p \cdot p \end{vmatrix} < 0$$

$$=(a \cdot a)OD(bcdep) - (b \cdot b)OD(acdep) + (c \cdot c)OD(abdep) - (d \cdot d)OD(abce) + (e \cdot e)OD(abcdp) - (p \cdot p)OD(abcde) < 0,$$

$$\text{where } OD(abcde) = \begin{vmatrix} 1 & a_x & a_y & a_z & a_t \\ 1 & b_x & b_y & b_z & b_t \\ 1 & c_x & c_y & c_z & c_t \\ 1 & d_x & d_y & d_z & d_t \\ 1 & e_x & e_y & e_z & e_t \end{vmatrix} = e \cdot \text{plane}(abcd).$$

Degeneracy handling

- Infinitesimal linear perturbation of points removes all InSphere degeneracies:

$$p \rightarrow (1, p_x + \mathbf{e}_x p_x, p_y + \mathbf{e}_y p_y, p_z + \mathbf{e}_z p_z, p_t + \mathbf{e}_t p_t + \mathbf{e}_{tx} p_x + \mathbf{e}_{ty} p_y + \mathbf{e}_{tz} p_z)$$

- Linear terms drop out of ODs
- Squared terms with different e s appear in last column of InSphere determinant (reuse minors)
- Sign determined by sign of lowest e term
- Points don't simultaneously lie on a sphere and a plane, so some determinant is non-zero.

Incremental Delaunay [ES96]

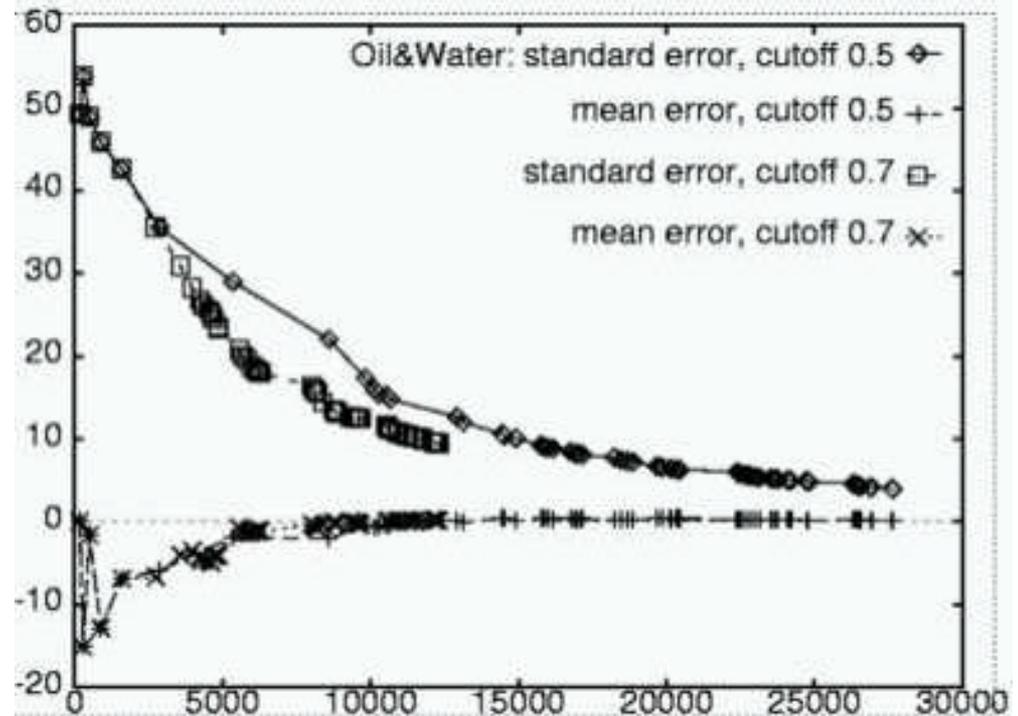
- Start with bounding mesh M (at \mathbb{Y})
- For each point p
 - Locate p in pentatope $p\hat{I} M$
 - Perform 1-5 flip on p, p
 - For each new pentatope using p
 - If opposite has p in Sphere
 - Do “cone test” with pentatope planes
 - Perform indicated flip

Measured computational effort

- Tests on 333Mhz Pentium II laptop
- 10,000 points produced 297,016 pentatopes.
- Computation time of 23.38 secs includes
 - 2,079,850 flips
 - 3,495,754 inSphere tests
 - 13,983,016 plane tests
 - 3,535,307 pentatopes created
 - 2,851,898 plane equations (of 17,676,535).
- Verification time of 4.32 secs includes
 - 1,485,080 inSphere tests
 - 7,425,400 plane tests

Coming:

- Graphs of error as a function of # of points
- Refinement techniques



Thanks to

- Hamish Carr (UBC)
- Ajith Mascarenhas (UNC)
- Lutz Kettner (now MPI Saarbruecken)

- Lawrence Livermore National Labs
- NSF/DARPA CARGO

- <http://www.cs.unc.edu/~compgeom>