The SpiderWeb Algorithm for Surface Construction from Medical Volume Data:

Geometric Properties of its Surface

Daniel Karron[1]
Departments of Applied Science and Physics
New York University, Department of Surgery
New York University Medical Center,
560 First Avenue, New York, NY 10016

voice : 212 263 5210
email : karron@nyu.edu
fax : 212 263 7190

James L. Cox[2]
Department of Computer and Information Science
Brooklyn College, CUNY
Brooklyn, New York, 11210

voice : 718 859 3861
e-mail: cox@spunky.nyu.edu

Bud Mishra[3]
Department of Computer Science
Courant Institute
New York University
251 Mercer Street
New York, New York 10012

voice : 212 998 3464
e-mail: mishra@nyu.edu

July 1992.

[1]Research supported by an equipment loan from Silicon Graphics, Inc.
[2]Research supported by National Science Foundation Grant IRI-9009788 and a PSC-CUNY Grant.
[3]Research Supported by National Science Foundation Grants IRI-9003986 and CCR-9002819.

Abstract — The "SpiderWeb" algorithm is a set of rules to construct a triangular mesh isosurface (simplicial complex) from density data. The algorithm is robust in the presence of noisy data in the sense that the surface grown will be an oriented twomanifold surface. Further, the algorithm constructs the triangulation using only local triangle–spinning rules, thus needing no additional global information. The triangle spinning rules use local geometric symmetries that translate into face–hit rules. These rules are not artifacts of the algorithm or any data assumptions, but rather inherent in the nature of density data, sampled at a given resolution or spatial frequency. SpiderWeb does not require ad hoc rules to deal with inconsistencies caused by so–called nongeometric noise. Thus the algorithm conforms to and strongly exploits previously unrecognized geometric properties of density data, while requiring minimal assumptions on the distribution of such data.

1 Introduction

1.1 What is the "SpiderWeb" algorithm ?

The SpiderWeb algorithm, as discovered and implemented by Karron [5], extracts a consistent [4] and continuous [5] two dimensional sheet as lists of triangles from three dimensional volume data.

A remarkable property of this algorithm is that there are no constraints on the datum value of an individual pixel. Pixel values do not need to be filtered, smoothed, or otherwise pre–processed to generate a satisfactory surface. Once groups of eight adjacent pixels are assembled into a voxel (defined below), a number of geometric constraints emerge by continuity that make triangularization of threshold crossing between voxel vertices ("hits") straightforward. Essentially, we will prove that for any data values hits, a voxel must have 2 or more adjacent face hits (the "Voxel Face Hit" Rule), and any voxel face can have only 2 or 4 hits(the "2-4 Voxel Face Hit" rule). The addition of an interior "articulation point" makes the triangulation of voxel edge hits unique by cyclic traversal of the hits on each voxel face to generate triangles(the "Voxel Face Triangle Spinning Rule"). Further, we will show that by the Voxel Face Hit Rule, triangles constructed in each voxel face will always share triangle edges with other triangles. This produces a continuous surface.

We will show that our isosurfaces are topological polyhedra, and in fact form a simplicial complex. Our surfaces are not constructed from standard CSG (Constructive Solid Geometry) primitives, but from the smaller subset consisting solely of triangles. This permits simplified CSG operators on our isosurfaces, as distinct from arbitrary surfaces embedded in 3D. Other isosurface algorithms produce holes or inconsistent normals in the presence of noise, and thus require complicated CSG operators on the isosurfaces or 3D volumes derived from MRI and Ultrasound imaging systems. There have been various attempts to enforce geometric and topoological integrity or consistency in the geometric primitives constructed from isosurface data. We show that these approaches are unnecessary. We endevour to provide a theory of 3D isosurfaces in the geometric

[4] Oriented: Surface normals are always consistent with respect to adjacent triangles and correct with respect to the local voxel volume gradient.

[5] Closed and Unbounded : The sheet of triangles will always close if allowed to grow to completion (no unresolved triangle edges) and will not have artifactual holes.

and solid modeling context. This theory will provide a next step for research into triangle decimation, to reduce the number of small triangles that do not significantly contribute to the geometry of a surface. Thus SpiderWeb gives a BREP (boundary representation) of the surface of a solid given by density data, avoiding many of the drawbacks associated with contouring or CSG in this problem setting.

Concurrently, we are conducting research into computer aided surgery (CAS), where more traditional applications of CSG (interference, surface joining, solid boolean operations) are performed on the 3D segmented volumes and their associated surfaces. Further, the consistent geometric and normal information obtained from the SpiderWeb surfaces is an important starting point for FEM (finite element methods) and BEM (boundary element methods) analysis of physical properties.

1.1.1 The "SpiderWeb" algorithm : Advantages

The SpiderWeb algorithm is a significant improvement over previous algorithms because:

- The SpiderWeb algorithm produces a closed, consistent, continuous sheet of triangles despite nongeometric noise in the data or weak segmentation threshold gradients. This noise is always present in MRI and Ultrasound data.
- It produces a unique triangulation for any given combination of intersections on voxel edges. It is not "case table driven".
- It is simple to implement. It does not use voxel edge intersection combination case tables and rotations of the case table, or special asymptotic decider auxiliary algorithms. The heart of the algorithm is a simple rule that can be independently applied to each face of the voxel in parallel.
- It can determine triangle edge and vertex adjacent triangles without scanning all adjacent local voxels. The voxel face specifies what is adjacent to each triangle edge and vertex for any sheet configuration.
- It can generate average interpolated normals and color vectors at triangle vertices that can give the tiled surface a smooth, seamless appearance despite geometric noise.
- The constructed sheet is not obscured by isolated noisy voxels. Noisy voxels are isolated clumps of voxels that cross the isovalue threshold but are not part of the sheet geometry that is under construction.

The "SpiderWeb" algorithm solves the problem of constructing a continuous web of triangles inside a voxel and between voxels to represent the passage of a single isovalue sheet from any possible combination of voxel vertex values.

1.1.2 The "SpiderWeb" algorithm : Drawbacks

The drawbacks of the "SpiderWeb" algorithm are :

- The SpiderWeb produces more triangles than "Marching Cubes" [8] or "Alligator" [4]. The modal numbers (most common value, mode) of triangles per voxel with triangle vertex condensation are 1 and 4. The maximum number of triangles per voxel is 24. The average number of triangles per voxel in high resolution MRI studies reconstructed to date is 2 to 3. Some triangles can be extremely small in order to maintain surface continuity. Degenerate triangles (having zero area and co–linear edges) are automatically culled.

- This algorithm has nothing to do with segmentation of data pixels into material or tissue types. The "SpiderWeb" sheet is a purely geometric construct. Segmentation issues are not addressed by this algorithm. The goal of a segmentation algorithm, in this context, is to increase the gradient steepness across the threshold isovalue sheet. A binary segmentation is equivalent to a vertical drop across the sheet threshold. A sheet created by a binary segmentation can be considered to have an almost infinite gradient (0/1).

2 Methods

The properties for the list of triangles we generate are as follows:

- The sheet that emerges from the face of a voxel must be continuous with the sheet that passes into the complementary face of the adjacent voxel. The triangle sheet must be voxel "face–wise" continuous.
- Only one sheet can pass through any one voxel. We do not make any assumptions that the sheet we are constructing can fold back and reenter a single voxel.
- The surface normals must be consistent in the triangles within a voxel and between voxels. No Möbius strips are allowed.
- The "SpiderWeb" algorithm will always construct a triangle sheet in the presence of a detectable gradient [6].

2.1 Voxel Construction

The SpiderWeb algorithm builds a web of triangle edges out of voxel edge threshold intersections ("hits"). A voxel edge vertex that "straddles" the threshold will have a hit. If both voxel edge vertices are above or below the threshold, there will not be a hit on the voxel edge. If two voxel vertices have exactly the same value, and that value is the same as the threshold, there will be exactly one hit contiguous with the far vertex (voxel edge vertex 1). By constructing an additional point from the first moment (center of mass) of the voxel edge hit, which we will call the "articulation point", [7] and using this point as a common vertex for all of the triangles built out to the faces of the voxel, an extremely simple method for constructing a consistent, continuous sheet of triangles is attained.

We will show later that one hit on one voxel edge implies at least two adjacent voxel faces containing two or four hits.

Volume data typically consists of some sampled density data at regular discrete points in three dimensional space. For the purposes of this section we will assume these samplings are at the integral points in three dimensional space, *i.e.,* the data is sampled at the points of the integral

[6] A detectable gradient is at least one count difference in the integer pixel value that crosses the floating point threshold level.

[7] The "articulation point" is the point about which the bending of the sheet between voxel faces occurs.

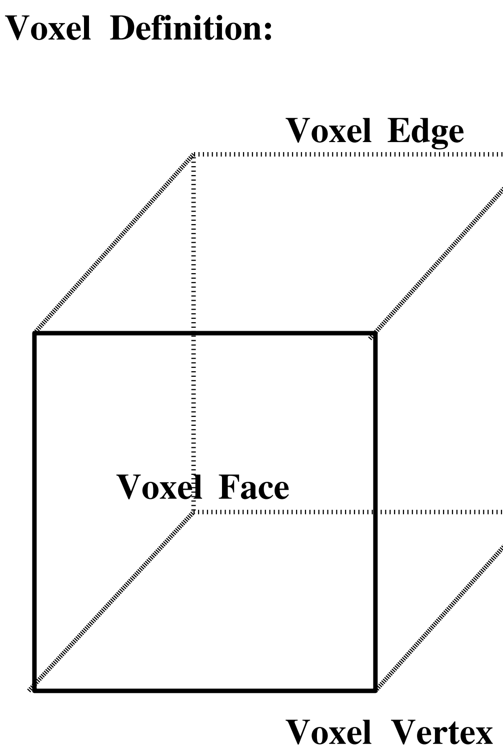

Figure 1: Illustration of voxel terminology.

lattice. A voxel is a unit cubic volume bounded by eight adjacent data sample points from the integer lattice.

In actuality, we are ignoring the distinction between integral data array space and real world coordinates. However, we will show that what we develop for cuboidal, integral voxels generalizes for data sampled in anisotropic non-cuboidal voxel shapes, where the faces can be rhomboidal.

We assume that our volume density data is contained in a 3 dimensional data array, indexed by i, j, k , bounded by $0..I, 0..J, 0..K$. Each slice consists of the subset of the array $0..I, 0..J$ in a particular slice k. Each element of the slice array is a pixel at index i, j . Each pixel is a sample value at an integral coordinate value. A voxel is constructed from 8 adjacent pixels, spanning $i..i+1, j..j+1, k..k+1$ for any particular voxel indexed i, j, k. Thus a voxel refers to the region of space bounded by pixels at the 8 vertices of the voxel. Each voxel consists of 8 voxel vertices, 12 voxel edges, and 6 voxel faces. Each cube edge is a voxel edge. Each lattice point is a voxel vertex.

For a given voxel V, there will be 6 adjacent voxel cubes, which share each of the faces of V. Counting voxels that share either faces, edges, or vertices of V, there will be 26 adjacent voxels. There will be 6 face adjacent voxels, 8 edge adjacent voxels, and 8 vertex adjacent voxels. Each edge e of V will be shared by two of V's faces. Each vertex v of V will be common to three of V's faces.

Voxel edges can cross a threshold value, and by linear interpolation along the voxel edges, the position of a hit can be determined. For a voxel edge Ve, consisting of voxel vertex Vv_0 and Vv_1, with position $Vv_{x0}, Vv_{y0}, Vv_{z0}$ and value Vv_{v0}, and $Vv_{x1}, Vv_{y1}, Vv_{z1}$ and value Vv_{v1}, there exists 4 states with respect to a threshold value τ.

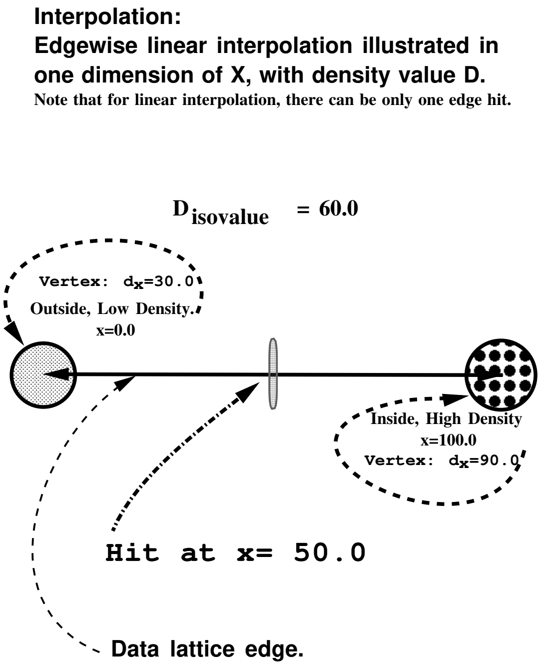

Figure 2: Illustration of voxel edge hits.

$Vv_0 > \tau$ and $Vv_1 > \tau$

There is no hit on VE.

$Vv_0 < \tau$ and $Vv_1 < \tau$

There is no hit on VE.

$Vv_0 < \tau$ and $Vv_1 \geq \tau$ $Vv_0 \geq \tau$ and $Vv_1 < \tau$

There is a hit on VE. The position of hit VE_{h0} is given by:

$$VE_{h0x} = Vv_{x0} + (Vv_{x1} - Vv_{x0}) \frac{\tau - Vv_{v0}}{Vv_{x1} - Vv_{x0}}$$
$$VE_{h0y} = Vv_{y0} + (Vv_{y1} - Vv_{y0}) \frac{\tau - Vv_{v0}}{Vv_{y1} - Vv_{y0}}$$
$$VE_{h0z} = Vv_{z0} + (Vv_{z1} - Vv_{z0}) \frac{\tau - Vv_{v0}}{Vv_{z1} - Vv_{z0}}$$

For example, if our threshold τ is 50.0, and if end 0 of an edge has a voxel vertex value of 100, and the other end 1 has a voxel vertex value of 0, the interpolated position of the hit is halfway between the ends of the voxel edge. Because voxel edges are shared by at least two voxel faces, and because as will be show below, a voxel face can only have 0, 2, or 4 hits, voxel faces containing hits propagate around the voxel.

2.2 Overview of Algorithm

Now we describe how the SpiderWeb algorithm constructs a sheet of triangles from the above voxels and their associated hits.

The simplicial complex surface is grown by a depth first surface tracking method, which recursively extends a portion of the triangle sheet within each adjacent voxel that contains a bit of this surface. The same procedure is applied to each voxel V' that shares a hit edge with V. For a voxel V with hits triangles are spun around each voxel face to a common interior articulation point, described below.

The SpiderWeb algorithm only builds surfaces adjacent to voxel faces that already have some surface triangles contained therein by recursing if a voxel face does indeed contain some triangles. In this way, the algorithm never steps into an empty voxel.

```
void MagicGrow(Voxel *V)
{
register i;

for(i=0;i<nFACES;i++)
  {
  if(V->Face[i].Tri==Spin(V->Face+i))
    Grow(V->Face[i].AdjVoxelPointer);
  }

BuildTriangleAdjacencyLists(V);
CullTriangles(V);
}
```

The above is a simplification of the actual implemented algorithm. First, the actual voxel may not be cubes. Secondly, the voxels with hits and their adjacencies have to be explicitly represented, as they are not actual geometric objects, but are constructed from subsets of the sample data. Thirdly, SpiderWeb, takes advantage of sparsity by not building voxels that do not contain any surface. In fact, once the triangles are completely built, (geometry, adjacency, and normals), the voxels can be discared. The SpiderWeb algorithm constructs these conceptual voxels from density data that is stored in an array. In particular, while the density values may not be sampled at integral lattice points, this view of the data is valid as long as the real world locations of the sampling points are monotonically increasing in each coordinate, since it is only the topological properties of the cuboid lattice we exploit.

In the next section we describe in detail the conceptual construction of triangle geometry, triangle adjacency, and triangle normal resolution.

2.3 Triangle Spinning

The voxel lattice contains pointers assembled in the voxel building stage that are the frame to which we attach the triangle webbing. The connectivity of pointers that is established at voxel build time before the triangle building step simplifies and speeds building and rebuilding of triangles at different surface threshold values.

For any arbitrary threshold value, each of the 8 vertices can be classified as being **In** or **Out**. **In** is defined as having a higher density than the threshold. **Out** is defined as having a lower density value than the threshold. At an infinite distance, we assume a zero density, and an **Out** orientation. After the construction of a voxel, with its associated tables of pointers, the generation of triangles is quite fast and simple.

2.3.1 The Articulation Point

The first moment (center of mass) of all of the hits is calculated as the mean of each hit point. The voxel hit centroid point, or **"articulation point"** is used as the voxel interior triangle vertex for all of the subsequently constructed triangles. For a collection of voxel edge hits h_n, $n = 0, 1..m$, the position of the articulation point $AP_{x,y,z}$ is

$$AP_x = \sum_{n=0}^{m} h_x n/m$$
$$AP_y = \sum_{n=0}^{m} h_y n/m$$
$$AP_z = \sum_{n=0}^{m} h_z n/m$$

The webbing of the triangle edge weaving around a central point give the voxels an appearance of a small spider web, or cob web inside the body of the voxels, hence the name for the algorithm.

There are two phases to spinning triangles.

2.3.2 Triangle Spinning : Phase 1, Scanning for Hits.

The first phase of triangle spinning is scanning for hits (threshold crossings along voxel edges) and then calculating the linear interpolated x, y, z along a voxel edge (a hit). Voxel edges that contain hits to be interpolated can be rapidly determined by a simple difference test along the 12 edges between the start and end vertices of each edge. Where one edge is greater than the threshold, and the other end is less than the threshold, the linear interpolation along the voxel edge is calculated.

2.3.3 Triangle Spinning : Phase 2, Voxel Face Triangle Spinning Rule.

The second phase of triangle spinning is the triangle assembly. This phase can be done in up to 6 parallel threads in shared memory, one for each voxel face. For each relevant face, a list of face edge hits is built. The voxel face edges are always traversed in clockwise order to keep the surface normals consistent. We observe that because we use monotonic linear interpolation along voxel edges, there cannot be more than one hit along an edge (see figure 3). Also, we show below that there can only be 2 or 4 hits on a voxel face (see figure 4).

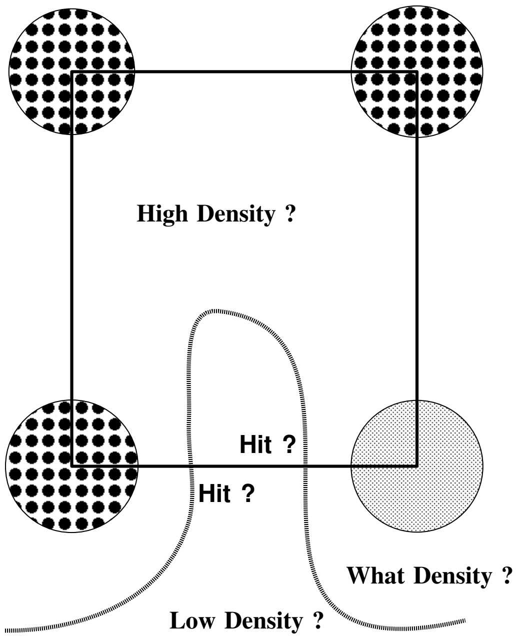

Figure 3: Density data from this surface crossing implies no hits.

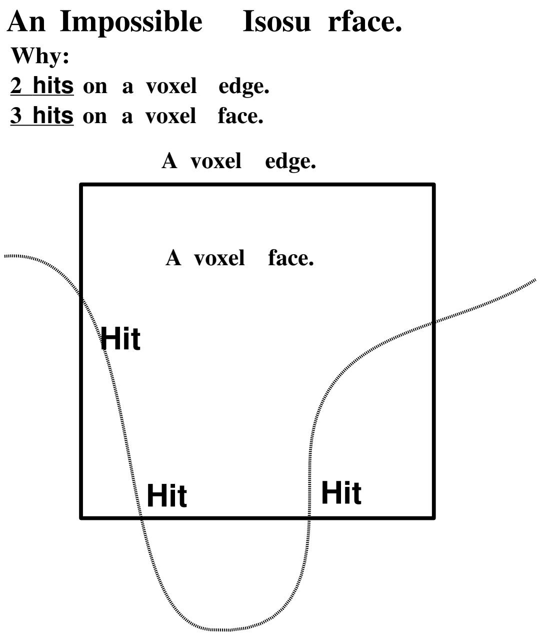

Figure 4: Illustration of impossibility of three face hits.

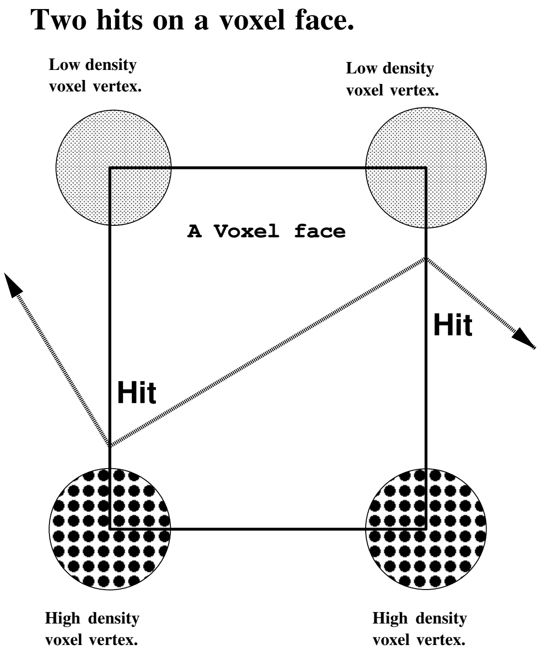

Figure 5: Illustration of a regular 2 hit voxel face.

If there is "0 hit", return as there is no geometry to construct.

There can be niether exactly 1 hit nor exactly 3 hits on a voxel face (see figure 3.3.1 in section 3). See Lemma 2, for a proof of this fact.

The next case is that of 2 voxel face edge hits on a voxel face (see figures 5 and 6).

In this instance, we spin a single triangle. The first voxel edge hit coordinate triple is copied to the triangle 0 vertex. The next triangle vertex is the articulation point inside the voxel. The last triangle vertex 2 is copied from the last voxel face edge hit. All triangle vertex 1's are interior to the voxel, a property we will exploit later.

By spinning around each voxel face in a clockwise manner and knitting triangles from the cyclic permuting of voxel face hits and the articulation point, we consistently spin triangles. The triangles are contiguous by parallel application of this rule on each face of the voxel.

- **Edge 0** plunges into the body of the voxel to the articulation point. **Vertex 0** is on the voxel face edge.
- **Edge 1** emerges from the body of the voxel to the voxel face edge hit. **Vertex 1** is congruent and shares the voxel "articulation point".
- **Edge 2**, the last edge, crosses the voxel face. **Vertex 2** is on the higher number voxel face edge.

A 4–hit is now examined in detail (see figures 7 and 8).

In the instance of 4 voxel face edge hits, we traverse around (always clockwise by convention) the face edges with the voxel vertices alternating from **In** to **Out**. We weave triangles by taking

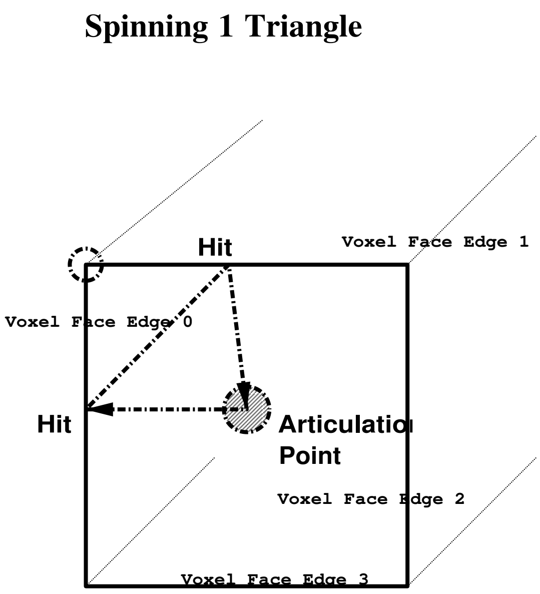

Figure 6: One triangle spun to 2 hit face.

cyclic pairs of hits around the face edges and knit triangles with the **articulation point** (AP) in the body of the voxel. A cyclic pair of hits is for 4 face edges 0,1,2,3, we build 4 triangles with voxel face vertices from $\{0, AP, 1\}, \{1, AP, 2\}, \{2, AP, 3\}, \{3, AP, 0\}$. There are two other possible interior combinations of face hits we could have used. These two triangles are completely interior to the 4 exterior triangles. Of the 6 possible triangles that can be constructed from the 4 hits, we do not need to build the 2 interior triangles. They are never visible. The 4 voxel face hit produce one half bubble, or an open bubble, which is always closed in the complementary face of the face–adjacent voxel. Because a 4 hit on one voxel must be repeated adjacent voxel on their shared face, the bubbles will always immediately close in the adjacent voxel. This is the geometric response of the SpiderWeb to a noisy weak threshold gradient. In the face of nongeometric noise, the "SpiderWeb" builds small bubbles in the surface. In the case of extreme noise, with no clear multivoxel gradient (complete random volume noise), the "SpiderWeb" will create bubbles in each of the faces of a voxel. The constructed triangle sheet geometry will resemble closed cell foam rubber of finite thickness. Normally a 2D surface has no thickness through its face. However, the surface will be continuous at its exterior surfaces, and in the presence of the slightest gradient, it will close around itself. This geometrical property gives the "SpiderWeb" its robustness in the face of noise.

Each face is independent of the other for the purposes of building triangles. A separate processor can be assigned to build triangles in each voxel face independent of each other. The simple cyclic construction of triangles about the face hits with the **articulation point** is the only rule required to implement the "SpiderWeb", once the proper traversal order is set up by pointers to voxel faces, edges, and vertices. No case table is required.

Because each triangle has a parent voxel face, and each triangle edge or vertex is exterior or interior, we can use this information to build out the continuous sheet until it closes, or until some artificial boundary is imposed. An example of an artificial boundary is a "region of interest" bounding box to subset the volume of data for exploration in interactive or reasonable time frames.

The "SpiderWeb" only builds a geometric construct through arbitrary gradient information. A sheet can only exist where there is some difference in density values. In theory, a steep gradient or a shallow gradient can produce the same geometry. Consider a constant gradient field in some

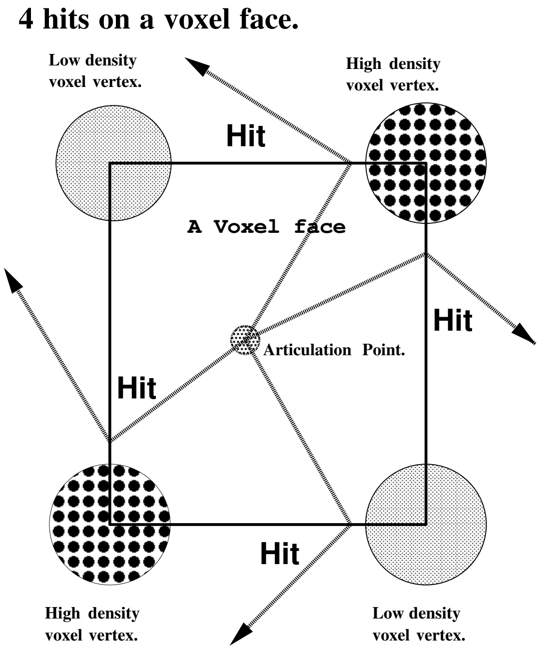

Figure 7: Illustration of 4 hits on a voxel face: A bubble voxel face.

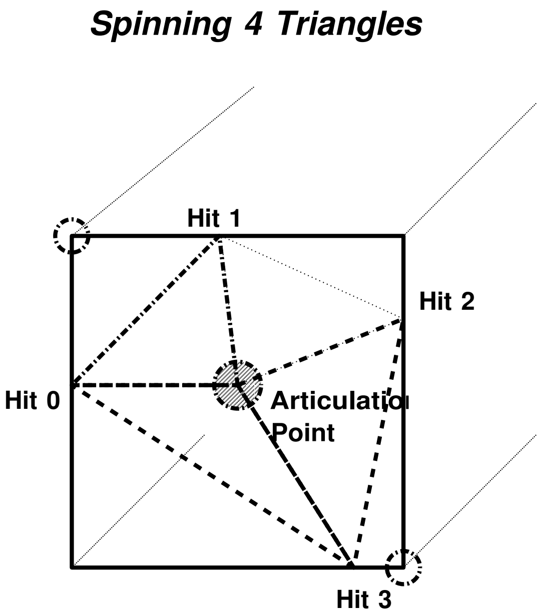

Figure 8: Bubble from 4 hit face.

direction. Any isovalue threshold will produce a flat surface. On one side of the half–space created by the threshold, all voxel vertices will be **In**, and on the other side, all voxel vertices will be **Out**. The difference in the vertices within each voxel split into two parts can be large or small. If we reduce the gradient, but not change the threshold, there will be no change in the geometry (assuming that the change in gradient is centered about the isovalue sheet). As the gradient gets smaller, the difference within the body of a voxel can approach one count. Since medical image information is typically stored in 12 bit shorts, the isovalue sheet can have a minimum of 1 count difference across its surface.

In the case of theoretic data (e.g. potential theory surfaces or laminar Computational Fluid Mechanics), where there is no noise, each voxel that straddles the isovalue sheet is precisely divided by a mostly flat half space. Consider the injection of noise into the voxel vertices. Geometric noise occurs when some vertices have values opposite from what would be given by their position with respect to some local half space dividing plane. The local half space plane is "torn into two" by the presence of contradictory voxel vertices. By enforcing facewise continuity, these tears are immediately closed in the adjacent voxel. A strong, uniform, gradient will pass cleanly through the corner of a voxel (a 3 hit) or through the body of a voxel (a 4–hit). Clean data with a large minimal radius of curvature with respect to the voxel size will never produce 4 hits on a voxel face. The quality of a data set can be evaluated by a histogram of the voxel hit counts. A weak, low contrast gradient of indistinct, noisy objects will produce bubbles. However, it is still possible without any pre–filtering to build a geometry from this data. Since all real medical image data has fuzzy or noisy boundaries, it is difficult to completely eliminate the noise by filtering. We have just shown that it is now unnecessary to prefilter the data to extract a usable geometry, and that filtering can improve the segmentation of a surface geometry.

Geometric noise can be defined for each triangle in a voxel. Each triangle in a voxel divides the voxel into two half spaces. The plane of the half space is defined by the affine hull of the triangle, extended out through the faces of the voxel. One side of each nonbubble triangle should have in its half space all of the **In** voxel vertices, and the other side of that triangle's half space should include all of the **Out** vertices. Therefore, there is no ambiguity in assigning the orientation of the outward pointing triangle normal vector. The triangle normal vector should point **Out** with respect to the half plane of the triangle. Assume that one pair of voxel vertices from the **Out** side and the **In** side swap values. A nongeometric condition will exist where the half space plane of a particular triangle no longer divides the voxel vertices into two groups. This is the operational definition of geometric noise used in the "SpiderWeb" algorithm. The failure of each triangle half space plane to simply segment all of the voxel vertices as **Out** and **In** is a property of bubble triangles.

The splitting of the normally manifold surface into 2 or more bubble surfaces does not invalidate the sheet. The particular instance of a bubble can always be detected by the number of triangles spun to the parent voxel face. Any number > 1 is diagnostic of a bubble triangle. Weiler [10] makes explicit use of nonmanifold surfaces and adjacencies in his study of topological structures for CAD–CAM geometric modeling.

In very noisy boundary gradients, it is possible to have completely submerged bubbles and a very large number of bubble triangles that are not part of any exposed surface. These triangles can be detected by the property that all three edges of these triangles are adjacent to only bubble triangles. Bubble triangles are tagged during their creation and are triangles produced by 4–hits on voxel faces. Therefore, any triangle that has all of its edges adjacent to other bubble triangles can be eliminated with no degradation of geometry. In this way, multi–voxel scale "foam" can be reduced to a single void volume.

2.4 Triangle Face Normal Resolution.

Each triangle face defines a tangent plane and a normal vector. The direction of this normal may or may not be consistent with the overall orientation of the voxel with respect to the volume gradient of the sheet due to nongeometric noise. After triangle construction, the normals need to be checked and reversed, if necessary to be consistent with the overall half–space plane orientation of the surface and the volume gradient. However, the volume gradient is not used to define triangle normals. It is only used to resolve difficult triangle face normals by reversing the direction of the triangle face normal if necessary.

The triangle face normal is constructed from the normalized cross product of the two triangle legs that share the articulation point triangle vertex. An additional test vector is constructed from the centroid of the triangle to each of the **Out** voxel vertices. The test vector is normalized. The average of the normalized **Out** vectors is dotted with the triangle face normal. If the triangle normal dotted with the **In** average vertex is positive, then that vertex is oriented in the correct direction, and it does not need to be reversed. If the dot product is negative, then the triangle normal is effectively reversed by multiplying it by -1.

Bubble triangle normal resolution is taken care of automatically by the above procedure. Consider a bubble in an otherwise flat sheet. Each bubble consists of 4 triangles from one voxel, and another 4 triangles from the complementary face of the adjacent voxel. The top two triangles of the half bubble will have a normal consistent with the adjacent sheet, while on the opposite surface, the bottom two triangles of the bubble will have normals that will be consistent with the reverse side of the sheet. The interior surface of the bubble will have contradictory normals. However, since there is no possible way to walk or see the interior surface of a bubble in normal sheet traversal, the contradictory normals inside a bubble surface are not a practical problem.

2.5 Triangle Edge and Vertex Adjacency List construction.

"Walking" or integrating regions on the surface of our constructed sheet requires knowing what triangles are adjacent at every triangle vertex and every triangle edge. Calculation of the vertex normals and vertex colors also depends on knowing the vertex adjacency in order to average the adjacent values to create a smooth continuous junction between each triangle.

After the triangles are created and the surface normals are resolved, the next step is listing triangle vertex and edge adjacency. Each triangle vertex and edge has a list of those triangle edges and vertices that are adjacent to it. Values at the face of each triangle can be smoothly interpolated across its edges and vertices by taking the mean of the adjacent face values of all adjacent triangles.

There are two ways to calculate the adjacency of a triangle. The first is to scan all of the adjacent triangles and voxels that contain triangles and test each vertex and edge for congruency. A congruent vertex has the same x, y, z values within some tolerance. For single precision floating point numbers, we have found that directly testing for equality is sufficient. A congruent triangle edge is tested for vertex congruence in both directions, *i.e.,* if one vertex of an edge matches one or the other potentially adjacent vertex edge, and the other vertex of this edge matches the other adjacent triangle edge vertex, then we can assume that this edge is adjacent. Each adjacent triangle is added to a growable array of triangle pointers at each triangle edge and vertex. An n–adjacent vertex has n triangles adjacent to it, excluding itself. An n–adjacent edge has n–triangles adjacent to itself. Usually, in a simple, generally flat sheet, the triangle vertices are 7 adjacent and the triangle edges are 1 adjacent. Triangles that constitute bubbles are typically $n > 1$ adjacent.

It is not necessary to scan all local voxel triangles and each adjacent voxel to build triangle edge and vertex adjacency lists. We can use the "SpiderWeb" voxel frame to eliminate unnecessary scanning. The above brute force adjacency scanner is useful to debug and check optimized adjacency scanning. For each triangle vertex we scan for adjacency, we know *a priori* it is triangle vertex 1, it is the "articulation point" vertex, and is adjacent with all of the triangles in the local voxel. If it is triangle edge 2, that vertex lies on the face of the voxel, and we only need scan the voxel face adjacent voxel complementary face triangle edge 2 for adjacency. Our scan of the adjacent voxels for adjacency can limit the search scope to adjacent triangles in the complementary face. Further research will determine the minimal search path required to produce a full adjacency list for all possible triangle geometries.

2.6 "MagicGrow" Depth First Surface Tracking.

Each voxel face that contains triangles points in the direction that the sheet continues. By building out from a seed voxel along each face that contains triangles, we can generate an isovalue sheet even in the presence of noncontiguous isovalues. This aspect of the SpiderWeb algorithm is called "MagicGrow". The isovalue of the cortical surface is also presence on the inner surface of the skull, and also on the outer surface of the scalp. If we built a sheet from all voxels in the head, we would obscure the cortical surface with a double sheet around the skull. However, by seeding our sheet building recursive algorithm with a location on the cortex, and assuming that there is no leakage of isovalues between cortex and skull, we can separate the two surfaces without prior editing or filtering. The recursive build out of an isovalue sheet eliminates "dirty voxels" or noise from obscuring the sheet we are building. We could hand edit the skull, however in the 120 slices typical of a high resolution MRI, this option becomes quite a bit of tedium. Construction of the geometry of many internal anatomic features, such as the brain ventricles, is easy because we can build triangle sheet geometry that is contiguous to the local structure we are building. This technique was originally described by Cline [2]. The main advantage of the "SpiderWeb" algorithm in recursively building continuous sheets is that because each voxel face owns its triangles, any voxel face that contains triangles is adjacent to a voxel that that sheets must extend into. If that voxel is not built, that voxel is built in the next recursive step. It is not necessary to scan all adjacent voxels. The SpiderWeb make it possible to know from the local voxel which way the sheet extends without testing all adjacent voxels.

"MagicGrow" builds a sheet with the hand selection of a seed voxel that contains a bit of the surface we wish to continue. The first call spins triangles inside that voxel. Each face that contains triangles also has a list of indices to the voxel which is face adjacent to the local voxel. The subroutine calls itself with the i, j, k values to build the adjacent voxel and its embedded sheet. The code leaves a marker in the voxels that it builds to indicate that the recursion has passed this way previously. This limits recursion to one pass through any one voxel. Each voxel face with triangles generates the face adjacent voxel until there are no unvisited voxels to grow out into. All of the triangle edge are resolved, *i.e.* all triangle edges are 1 or more adjacent.

In "MagicGrow" mode, a voxel is never built with 0 triangles. Triangle adjacency is calculated on return from the MagicGrow subroutine recursion in each invocation. At this point in the code, we know that the sheet is built out and locally complete. We can then wire up the adjacency pointers and calculate the triangle vertex normals and colors. We can also free up the voxel data structure and save only the triangles at this point because we do not need the voxel data structure as scaffolding for further calculations.

A problem with "MagicGrow" is that as the sheet flows over any and every contiguous surface with zero viscosity. Any leakage from the cortex surface to the skull surface can start the algorithm outlining the entire skull. This defeats the purpose of the surface tracking. A single break in the barrier voxel vertex values can cause the sheet to leak onto unwanted regions. There are two ways to prevent this. One is to limit the region of interest (ROI) to small area, and recurse through them separately. This technique of "iterative recursion" was suggested by Lorensen [7]. When a small region of leakage is found, it is easier to look through a small area to find the gap and edit it manually in this area. The other option is to increase the contrast by prefiltering about the threshold to close the leaky voxels.

2.7 Triangle Culling.

The main drawback of the "SpiderWeb" algorithm is that it builds many co–planar triangles. After all of the adjacent voxel triangles in a sheet are spun, and the sheet is locally complete (we have built all of the local adjacent triangles), we can condense triangles within a voxel.

The simplest and most common combination of triangles in a voxel is produced by a 3–hit voxel. This generates three triangles in three faces (two hits in each face, as one voxel edge hit is counted twice in each voxel face). These triangles are generally co–planar and the interior vertices are 2–adjacent. Each triangle articulation point vertex is adjacent to 2 other triangles. We can, with no degradation of the geometry, remove the articulation point vertex and condense three triangles into one. However, we break up the symmetrical construction pattern of the triangles in the voxel strip and can prevent further tmesh condensation operations.

Any group of three 2–vertex adjacent triangles, can be condensed by removing the common interior vertex. In general, any set of n–vertex adjacent sufficiently co–planar triangles can be combined into a flat polygon, if the standard deviation of the triangle face normals is smaller than some acceptable tolerance level. This means that the face normals point in roughly the same direction as the average of the face normals.

There are drawbacks to excessive triangle culling. For the purposes of physical property modeling, a regular triangulation, even where there is little geometrical change, is important. There may be important magnetic or electric field inflections that might be missed by undersampling. Another problem with arbitrary vertex plucking is that for the Silicon Graphics geometry engines, there is no saving in increasing the value of n of an arbitrary closed n–gon. This is because the SGI geometry engines internally triangulate closed polygons into triangles [1]. The only local operation that produces a savings in draw time is condensing three 2 adjacent triangles into 1 triangle. Recent work on triangle "Decimation" by Lorensen [7] shows promise in reducing the number of triangles by two orders of magnitude required to describe a geometry while retaining fine and essential features of the geometry.

2.8 Segmentation

In medical imaging and 3D reconstruction, segmentation is the classification of each pixel into two or more material groups. A material group is some tissue classification, such as bone, air, blood, fat, skin, etc. A surface is the interface between two different materials. An example of a biologically important surface would have muscle protein consistently on one side and fat on the other.

Assigning each image pixel to a material class can lead directly to a 3D geometry. If we assume a more standard definition of a voxel as one and only one pixel, then a voxel is a box shaped volume region about the center of the pixel. The surface representing the transition from one material to another is along either a voxel face, edge or vertex. This leads to a boxy rendition of a surface as tiled by voxel faces. This is the current state of the art in surface reconstruction: "Segment pixel data first, then construct geometry from the boundaries." Various segmentation algorithms are reviewed in Kalvin91 [4].

Because "SpiderWeb" can construct geometry in any volume data, segmentation can be redefined:

- The segmentation problem is to increase the gradient steepness across some biologically significant threshold surface so that the surface has a strong gradient with minimal nongeometric noise. Further one can map a range of density values to a single density value so that the surface in question is truly an isosurface; i.e. is represented by a single density value.

Current medical instrumentation only provides one dimension of value information at each pixel. This is due in part to the lack of algorithms for interpreting the underlying multidimensional information that MRI and CT can provide. MRI can provide many dimensions of information about the material represented by a pixel. Water and fat are the two most common segmentations desired. Hummel [?] promulgated a class of relaxation labeling algorithms to assign material classifier probabilities to reconstruct material information from grey scale CT. In this way a probability map of air, water, or bone is mapped in an overlapping fashion on a CT slice. Depending on the MRI pulse sequence, MRI can emphasize and increase the contrast between water, fat and protein. By setting the SpiderWeb threshold, approximate segmentation of these materials can be produced. However, a particular threshold does not produce segmentation.

2.9 Overview of algorithm : Macroscopic and Microscopic properties

First let us look at the properties of the "SpiderWeb" algorithm in the vicinity of 1 and 2 voxels. Then we will look at the global properties of a triangle sheet. These are properties of an entire sheet without any unresolved edges that is allowed to grow to completion. The remarkable feature of the SpiderWeb algorithm is that the global properties derive as a logical consequence of the microscopic triangles spun in each individual voxel.

2.9.1 Overview : Microscopic.

The microscopic properties of sheets of triangles produced by the SpiderWeb algorithm are that:

- The sheet is voxel facewise continuous.
- Voxel face 4–hits produces closed bubbles on the size scale of one voxel.

2.9.2 Overview : Macroscopic.

The macroscopic properties of sheets of triangles are: If the sheet is allowed to grow to completion, *i.e.,* no unresolved triangle edges, the sheet will always close in a closed volume. Various artificial boundaries can limit sheet extant for visibility of underlying structures.

3 Proof of correctness

3.1 Introduction

The purpose of this section is to prove that the "SpiderWeb" algorithm constructs a two dimensional simplicial complex that is both a manifold and is coherently orientable, with the exception of a finite number of points. Further, if desired, these exceptional points can be handled so that the complex can be smoothed (interpolated vertex normals) to yield a compact oriented 2–dimensional manifold embedded in Euclidean 3–space. Remarkably the SpiderWeb algorithm produces a globally consistent surface using only local properties of the density function. It is thus able to handle nongeometric noise in a way that is more robust on dense data than other interpolative and case driven surface construction techniques.

In actuality the Spiderweb algorithm may produce several manifold isosurfaces, which are resolved into "bubbles". Here we examine the execution of one thread of the SpiderWeb algorithm run from a seed voxel containing "regular" faces (defined below). The surface is grown by this execution thread via a depth–first surface tracking construction from this single voxel. A subsurface is then identified during normal resolution. If run simultaneously from various voxels, the SpiderWeb surface tracking can produce several manifold surfaces. The bubbles can be popped.

The only assumptions made about the density information are that the region of nonzero density is bounded and that a single isosurface cannot fold back in a region less than two voxels wide. The correctness of the algorithm will follow from this assumption and the purely geometric and topological properties and symmetries of the integral voxel lattice. We will show that bifurcations of the surface due to nongeometric noise can produce nonmanifold "bubbles", but that these bubbles will be eliminated during the normal resolution phase; further, the resulting normals will reflect a consistently oriented surface. The surface will be oriented in the sense that the normals will point in the outward direction, that is toward the region of lower density.

3.2 Preliminaries

For the purposes of this proof we will be using the Euclidean metric[8]. and standard point set topology on E^3 [9].

We recall some basic topological and differential geometric notions:

- A **surface** is an arc–wise connected space that is topologically two dimensional in nature. This means that any two points on the surface are connected by a simple path of finite length.
- A surface is **bounded** if it is contained in an open ball (the interior of some sphere).
- A surface is **closed** if it is bounded and has no boundary, *i.e.,* it closes on itself and has no surface holes.
- A surface is **compact** if it is closed and bounded.

All our surfaces will be subsets of E^3, and thus by definition are embedded in E^3. This will simplify definitions.

[8] Standard simple distance function.
[9] Uniform Cartesian coordinates.

Definition 1 *A* **two manifold surface** *is a topologically two dimensional connected surface where each point on the surface has a neighborhood homeomorphic to an open disk; that is, it is locally Euclidean.*

Definition 2 **A compact two manifold** *is a compact surface which is a two manifold surface.*

Thus a surface is a two manifold if it does not bifurcate at any point into two subsurfaces. At a point of bifurcation there is no way to continuously deform the surrounding region into a flat surface, i.e., the region surrounding and including the bifurcation point is not homeomorphic to plane.

Definition 3 *A surface in E^3 is* **smooth** *if the tangent plane to the surface is well defined at each point, and varies continuously over the surface, i.e., it has a regular parametric representation of class C^2. See for example "Differential Geometry" by Lipschutz [6].*

Thus the definition of an orientable two manifold in E^3 excludes surfaces found in Mobius strips and Klein bottles. The surface is two sided, consisting of an **Inside** and an **Outside**.

The following notions are taken from Geometric and Solid Modeling[3], [9].

Definition 4 *A two dimensional* **simplicial complex** *T is a finite set of triangles satisfying the following restriction: If t and t' are in T then either their intersection is empty or t and t' share an edge or a vertex. A two dimensional simplicial complex is* **without boundary** *if each triangle edge is shared by at least two triangles.*

An immediate consequence of the above definition is given in the following lemma.

Lemma 1 *If a two dimensional simplicial complex without boundary is contained in a bounded region of E^3 then it is a closed surface.*

It is well known that a two dimensional simplicial complex is a manifold if it satisfies the following two properties: Each triangle edge is shared by exactly two triangles, and the link of every triangle vertex is a triangulation of the circle, i.e. the region surrounding the vertex is homeomorphic to the plane.

We will show that after normal resolution each triangle edge will be shared by at most two triangles. Further the region surrounding each vertex will be a triangulation of the circle except for a finite number of cases. Further in these cases, the voxel triangulation can be replaced by an alternate triangulation that preserves overall connectivity and structure and makes the complex a manifold. However in practice this will not be necessary as the smoothed image will replace these exceptions by a portion of a manifold surface in a consistent manner.

Definition 5 *A triangle is oriented by choosing a cyclic ordering of the vertices. Two triangles in a complex are coherently oriented if their orientations are such the orientations they induce on their shared edge (if any) are opposite. A two dimensional simplicial complex* **T** *is* **coherently orientable** *if all pairs of triangles are coherently oriented.*

It is well known that a simplicial complex in E^3 is coherently oriented if and only if the normal directions implied by the triangle orientations represent a consistent orientation in the topological sense. We can show that our complex is coherently oriented by choosing triangle normals and we are also able to choose triangle vertex normals consistently with the corresponding triangle normals. Thus if the complex is coherently oriented we have chosen the vertex normal vectors so that **T** is indeed two sided. It is well known that any coherently oriented two manifold simplicial complex embedded in E^3 is homeomorphic to an oriented two manifold surface (and visa-versa). This surface is constructed by SpiderWeb via a linear interpolation of triangle vertex normals.

3.3 Correctness of SpiderWeb algorithm

3.3.1 Properties of Voxel Hits.

First we review the properties of voxel hits.

Definition 6 *A voxel* **hit** *on a voxel edge occurs when its endpoint voxel vertex values, a and b, straddle the threshold isovalue τ, i.e., $a < \tau \leq b$. The higher density voxel vertex (with value b) is said to be* **IN** *while the lower density vertex (value a) is* **OUT**. *Note that in the degenerate case, where $\tau = b$, a hit occurs that is coincident with a voxel vertex. These hits are counted by multiplicity; once for each adjacent voxel vertex with a value $a < \tau$.*

Now the SpiderWeb algorithm will perturb any vertex value that is identically τ by some small positive constant ϵ. It is easy to see that this perturbation still preserves the number of hits, when counted by multiplicity as in the above definition. The perturbation merely distributes the hits on the perturbed vertex onto the incident voxel edges.

Henceforth we shall assume that no voxel vertex density value is exactly τ. We call this the **perturbation assumption**.

Definition 7 *A voxel face is said to be* **empty** *if it contains no voxel edge hits, is* **regular** *if it contains exactly 2 voxel edge hits and* **irregular** *if contains exactly four hits.*

The reason for this terminology is that irregular faces represent the results of noise in the data. Four hits on a face corresponds to a situation that does not represent an isosurface that cleanly divides space locally into separate regions of high and low density, that is the IN and OUT regions.

Definition 8 *A voxel is said to be a* **tunnel voxel** *if it has exactly six hits in the following manner: Three of the hits are on voxel edges that share a common corner C and the other three hits are on three edges that share the corner diagonally opposite C.*

The reason for this terminology is that the triangles spun in this tunnel voxel will form a degenerate structure that appears like a closed tunnel, crimped at the articulation point. It is easy to see, however, that this structure can be replaced by an alternate triangulation that produces a structure homeomorphic to a tube (genus 1) by removing the articulation point and "opening" the tunnel (see figure 9). Examination of the possible patterns of hits on cube edges reveals that this is the only way that such a crimped point can occur. A crimped point cannot be the center

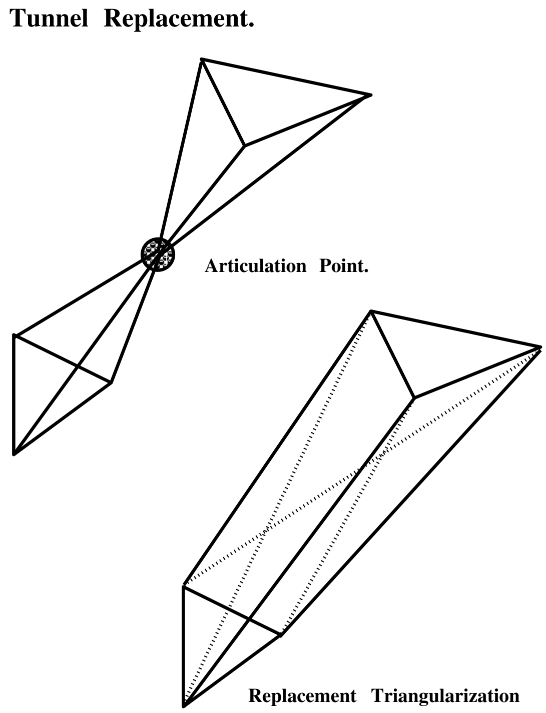

Figure 9: Resolution of Tunnel Voxels to remove nonmanifold point.

of more than two of these pyramidal structures, as this would imply more than one hit on a voxel edge.

Thus even though the region surrounding the articulation point in a tunnel voxel is not homeomorphic to a disk, we can replace this portion of the complex by a set of triangles that still share the same triangle face edges with neighboring voxels, that is, without disturbing the connectivity of our simplicial complex (see figure 9).

Lemma 2 *There can be neither exactly 1 nor exactly 3 hits on a voxel face.*

> **Proof:** Examine all possible assignments of high and low density values (resp. IN and OUT) to the four corners of a square. No such assignment can produce an odd number of face threshold crossings. □

Notice that this is one of the key observations that enables one to construct a surface via the SpiderWeb algorithm. There is no possible way to assign density values (values greater than or less than τ) to the four vertices of a square so as to produce an odd number of edge hits on that face. This is because as we traverse the edges of the square in clockwise fashion we see that for each threshold crossing, from say IN to OUT, there must be a corresponding crossing from OUT back to IN.

To further illustrate, note that in figure 3.3.1 there is a single hit on the bottom edge of the face. The upper right most corner has not been assigned a density value. If we assign this corner a high density, then there will be a hit on the right most edge. If we assign it a low density value, there will be a hit on the top edge. Thus, there is no way we can have a single hit on a voxel face. Similarly, there cannot be three hits on a voxel face.

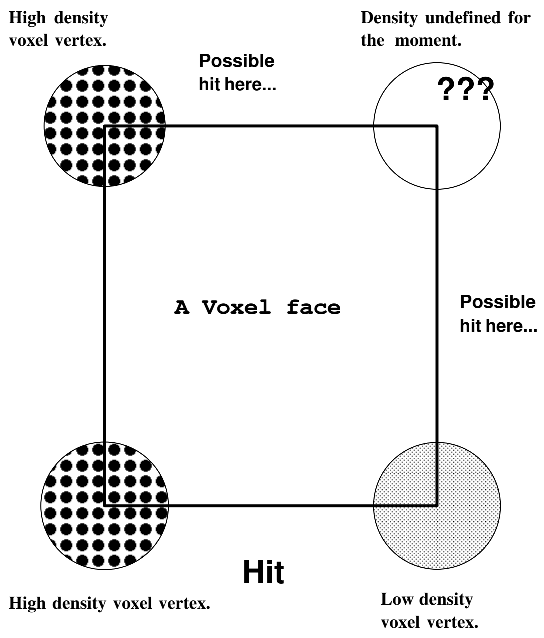

Figure 10: Impossibility of an odd number of hits.

3.3.2 Properties of Triangles

We will now prove that the depth–first surface tracking of the SpiderWeb algorithm run from a regular voxel face produces a simplicial complex that is **closed**, **compact**, and **oriented**.

Lemma 3 *SpiderWeb produces a simplicial complex without boundary.*

Proof: To show that the complex is without boundary it is enough to show that each triangle edge produced by the SpiderWeb algorithm is shared by at least one other triangle.

Each triangle constructed by the algorithm consists of three vertices: the articulation point of the voxel, and two edge hits on a single face. The corresponding triangle edges consist of two edges that connect the articulation point with the respective hit points, and an edge connecting the two hit points. Note that there is a triangle edge from the articulation point of a voxel to each voxel hit point.

In the first case let e be the edge of the triangle t that connects two distinct hit points h_0 and h_1 on face F of voxel V. Consider the adjacent voxel V' that shares face F. If F is a regular face then the segment e uniquely connects the hits in face F. Thus Spiderweb will construct a triangle t' to the articulation point of V' that shares triangle edge e. If F is an irregular face (contains four edge hits), SpiderWeb insures that the four triangle edges in F that connect these hits are chosen so that each edge connects a pair of hits on adjacent edges of F, that is, there will be a triangle edge from h_0 to h_1 if and only if the voxel edges containing the pair of hits share a common voxel vertex in F. Thus SpiderWeb will construct a triangle t' to the articulation point of V' that shares edge e.

In the second case triangle edge e connects the articulation point of V to a hit point h on face F. This hit is also shared on an adjacent face F' of voxel V. By Lemma 2, F' must have exactly 2 or exactly 4 hits. Thus h will be paired with a hit h' on F' and

e will be shared by the triangle t' from the articulation point of V to this pair of hits on F'.

The third case is the degenerate case where both voxel edge hits are coincident on a single voxel vertex, with density value equal to τ. But this case is excluded by the perturbation assumption. □

3.3.3 Zero Value Assumption

Zero Value Assumption:

We assume that the density value of space at infinity is 0. In addition the region of nonzero density is bounded.

This assumption has powerful implications. One of the important implications is that the data cannot be completely "noisy", in the sense that there must exist some regular faces. Thus in the case of extremely noisy data, we are guaranteed to be able to grow at least one coherent isosurface that surrounds the noisy high density region.

Theorem 1 *SpiderWeb produces a closed surface.*

Proof: By the zero value assumption the region of the voxel lattice containing nonempty voxel faces is bounded. Thus from Lemmas 1 and 3 the complex is closed. □

Definition 9 *An edge shared by more than 2 triangles is a* **bifurcation edge** *if the* **entire** *edge is an edge of at least three triangles. Thus we do* **not** *count an edge as shared between two triangles if they only share an endpoint of e, that is, a shared triangle vertex.*

We now show that bifurcations in the complex can occur only in a particular way. First observe that if the complex bifurcates, then by edge connectivity of the complex the bifurcation must occur at a bifurcation edge.

Lemma 4 *If an edge e is a bifurcation edge in voxel V then e is a segment from the articulation point a of V to a hit point h.*

Proof: The lemma is proved by observing that an edge e between two hit points on a face F can only be shared by two triangles: the triangles t and t' to the articulation points of the voxels V and V' that share F. □

Theorem 2 *Voxel V contains a bifurcation triangle edge e if and only if V has an irregular face F; further, e is incident upon a hit point of this irregular face F.*

Proof: Suppose e is a bifurcation edge shared by triangles t_1, t_2, and t_3. By the preceding lemma e is a triangle edge from a to hit point h_0 on face F. Since e is shared by three triangles, h_0 is a common hit point of these three triangles. Let h_1, h_2, and h_3 be the additional hit points of t_1, t_2, and t_3, respectively. Since by the perturbation assumption h_0 is not a voxel vertex, h_0 is common to at most two faces of V, F and F'. Thus by the pigeon–hole principle one of the faces F or F' must contain two of the additional hits, say h_1 and h_2. Without loss of generality, assume this to be F. But since F also shares h_0 it contains three hits. But by Lemma 2 this implies that F has four hits and hence V has an irregular face.

For the other direction assume that F is an irregular face of voxel V. SpiderWeb will construct 4 triangles $t_1, \ldots, t_4$, from the articulation point a of V to the irregular face F, with shared triangle edges (emerging from a) $e_1, \ldots, e_4$, where each e_i $i = 1, \ldots, 3$ is shared between t_i and t_{i+1}, and e_4 is common to t_4 and t_1.

Consider, without loss of generality, triangle t_1, where t_1 contains edge e_1 from a to hit point h. An adjacent face F' will share hit point h with F. By Lemma 2, there exists another hit point h' on F', so that the triangle t' with vertices a, h, and h' will share the edge e_1. Thus in this case e_1 is shared by the 3 triangles t_1, t_2 and t' and is therefor a bifurcation edge. □

Definition 10 *A* **bubble** *triangle is a triangle with an edge e so that e is the segment between two hits on an irregular face. A connected region of a voxel consisting of bubble triangles is called a* **bubble**. *A connected region of the complex consisting of bubbles is* **foam**. *A triangle lying completely inside foam is said to be* **submerged**. *A bubble triangle which is not submerged is called a* **foam surface** *triangle.*

Lemma 5 *A triangle is submerged if and only if each edge is adjacent only to bubble triangles and both triangle normals, when of sufficiently small length, are contained in a region enclosed by a bubble.*

Proof: If a triangle t is submerged then it is completely surrounded by bubbles and hence has no adjacent nonbubble triangle. Further each normal points into a bubble. Conversely if a triangle has an adjacent nonbubble triangle then by theorem 1 it is adjacent to a nonbifurcating portion of the complex and hence cannot be submerged. If a sufficiently short segment extended from a point in the interior of t in one of its normal directions lies completely outside of a bubble, then t is on the surface of the foam. □

In addition to the surface tracking phase of SpiderWeb, there is a normal resolution phase. The two phases are intertwined by the recursive nature of the algorithm. However we may view the phases as distinct.

Recall that the normal resolution process begins at a nonbubble triangle t, one that is adjacent to a regular face. It resolves normals by computing the tangent plane at each triangle vertex v of t (via an interpolation of incident triangles) and then choosing the perpendicular normal to v in the outward direction. Normals are then resolved for the vertices of the triangles adjacent to t, and so on.

Theorem 3 *The normal resolution process on complex T produces, upon appropriate replacement of tunnel voxels, an oriented two manifold complex T'.*

Proof: A triangle has been resolved when its normal has been assigned. The theorem will be demonstrated by showing that the following invariant is maintained during the normal resolution process: the set of resolved triangles forms a coherently oriented complex, and for each edge on a resolved triangle, exactly one other triangle that shares that edge will be resolved. Then the complex can be made a manifold by replacing the triangulation in a tunnel voxel by the appropriate structure (see figure 9).

The normal resolution process begins with a triangle t adjacent to a regular voxel face F. It computes each of the two possible normal directions. The normal is then selected to be the direction perpendicular to the triangle plane and consistent with the OUT voxel vertices of F. Note that since F is regular there are either one, two, or three adjacent OUT voxel vertices of F, and if there are two they lie in the same half-plane induced by t. Specifically the unit normals reside in the same t half–space (defined by the plane of t) as the OUT voxel vertices of F. This can always be done in the case of a regular face.

Once the normal direction to a triangle t has been resolved, then we resolve each edge adjacent nonbubble triangle t', We argue that this can always be done. Since t' shares a hit point with t, on a face F', F' will also share one of the out voxel vertices with F'. But again since F' is regular, its OUT voxel vertices will lie in the same t'-plane, and thus the normal direction for t' can be chosen coherently with t. Thus the two triangles have been coherently oriented (where the cyclic ordering of a triangle is the one induced by the right hand rule applied to the vertex normal of any one of its vertices). The invariant is thus maintained for all nonfoam regions.

It remains to demonstrate that the normals are resolved correctly for foam, and no bifurcation is included in the final resolved surface. Once the surface surrounding a foam region has been resolved, we propagate the resolution process along a surface of the foam. Thus no submerged bubble triangles will be included, as these can be recognized locally, from Lemma 5.

Let t be a resolved triangle with a neighbor triangle along edge e. If e is a bifurcation edge, SpiderWeb will choose one of the at most two adjacent bubble surface triangles (that share e). SpiderWeb chooses the unresolved triangle t' that may be assigned a normal consistent with the normals of t, i.e. whose OUT direction lies in the same t half–space as t's vertex normals. This process insures that only one side of a bifurcation is resolved and hence selected for inclusion in the normalized surface. Hence the invariant is maintained.

Once each tunnel voxel is replaced by a tunnel, the complex now satisfies the properties of a manifold. Further it is coherently oriented, since the normal to each included triangle points in a unique OUT direction. □

4 Concluding Remarks

In this section we describe some of the properties of volume density data that SpiderWeb exploits. First we discuss resolvability of surface features and curvature smaller than the voxel size. Secondly we discuss how, under minimal assumptions on the nature of the density distribution that is being sampled, isosurfaces cannot bifurcate in arbitrary ways. The voxel analogue of this important observation implies the face hit rules. We suggest then that SpiderWeb exploits and conforms to these properties, and is thus in no way ad hoc.

If the minimal radius of curvature of the sampled isosurface within any plane is at least twice the voxel length in each direction, then this curvature and associated features can be detected. This assumes that the density data is sampled regularly, though perhaps an–isotropically. Such an-isotropic sampling is common for MRI data, where the in plane resolution is 1 mm by 1 mm, and the between plane (or the Z axis) resolution is 3 mm. This gives us an ellipsoid of safely resolvalble curvature.

The pattern of density readings created by the two curves in figure 11 are identical. Thus one cannot distinguish a double surface crossing (or any even number of crossings) between two adjacent sampling points from no crossing at all.

The key point here is that there is no reason to interpolate an additional surface crossing on the bottom edge in each of the two faces shown in figure 11. The surface in face A is indistinguishable from the surface in face B. In fact the principal of Occam's Razor suggests that the linear interpolation used by SpiderWeb (edge hit rule) is perhaps the proper one. In analogy to the Nyquist Sampling Theorem for frequency sampled data, in order to detect such a curvature within a voxel unambiguously, one would have to sample the density at a larger number of points, i.e. increase the spatial frequency of the sampling.

We can see in figure 12 that the minimum resolvable "U–turn" is such that the U-turn must straddle a voxel vertex (face A) while a U-turn completely contained within a voxel is unresolvable (face B). Thus if the minimal radius of curvature is greater than the width of a voxel, we will insure that the associated feature never lies entirely within a voxel.

For the purposes of this discussion let us assume that the actual density has the following properties. First assume that at points where the function attains isovalues, the function does not achieve a local extremum. Secondly we assume that the set of isovalued points forms a two dimensional surface within the space. A simpler, but stronger, assumption is to require that the function be continuous in space.

From these two assumptions we find that any open ball containing a surface portion must also contain points of both higher and lower density than the isovalue. This means that the volume surrounding the surface contains both interior and exterior regions.

Consider a single plane that properly slices the isosurface, so that there are no two dimensional regions of isovalues, as in figure 13. Note that following the above assumptions and the Jordan Curve theorem, the surface cannot trifurcate; in fact, it cannot split into any odd number of subsurfaces at any point in this plane. This would violate the division of this plane into interior and exterior regions, i.e., it would violate one of our assumptions about the density data. Thus the corresponding face hit rules that SpiderWeb exploits are not just regularities in lattice data, but are expressions of symmetries that exist in the underlying continuum density distribution. The density assumptions made above are common to many applications, such as isosurface construction from bone and brain MRI data.

5 Bibliography

References

[1] Kurt Akeley. The silicon graphics 4d / 240gtx superworkstations. *IEEE Computer Graphics and Applications*, 9(4):71 – 83, July 1989.

[2] Harvey E. Cline, H.R. Hart Jr. C.L. Dumoulin, and William E. Lorensen. 3d reconstruction of the brain from magnetic renonance images using a connectivity algorithm. *Magnetic Resonance Imaging*, 5:345 – 352, July 1987.

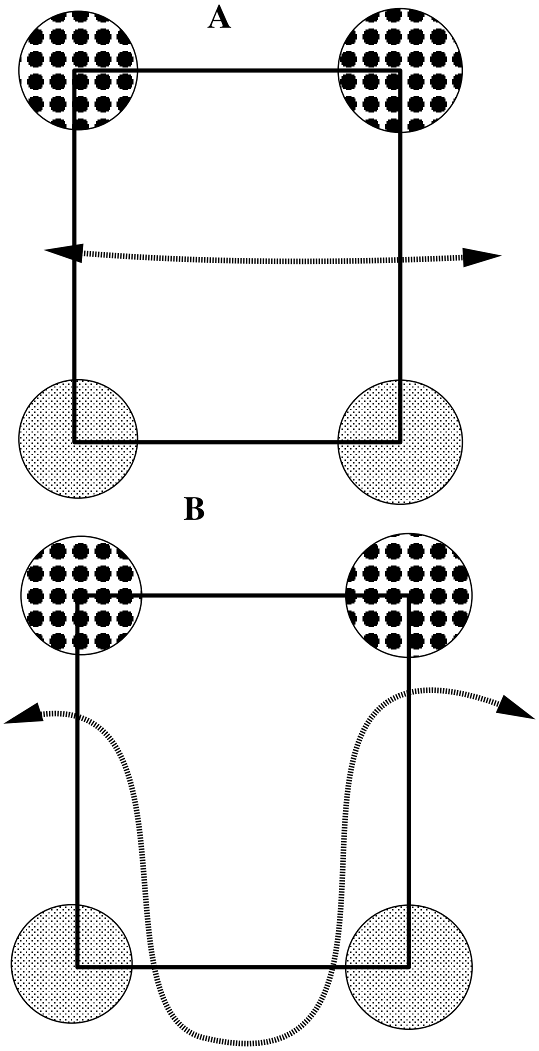

Figure 11: Indistinguishable isosurfaces.

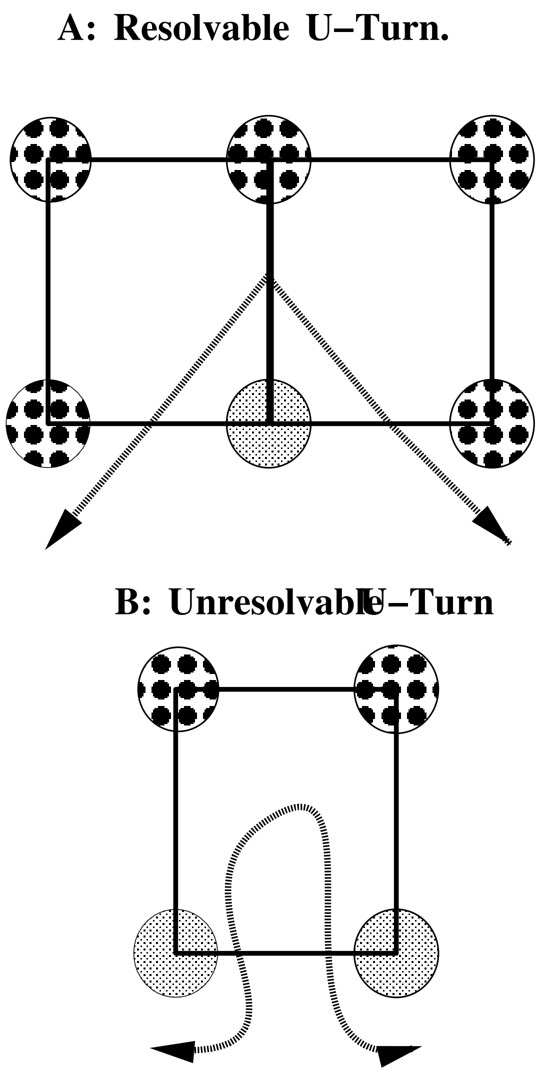

Figure 12: Resolvable and unresolvable surface curvature.

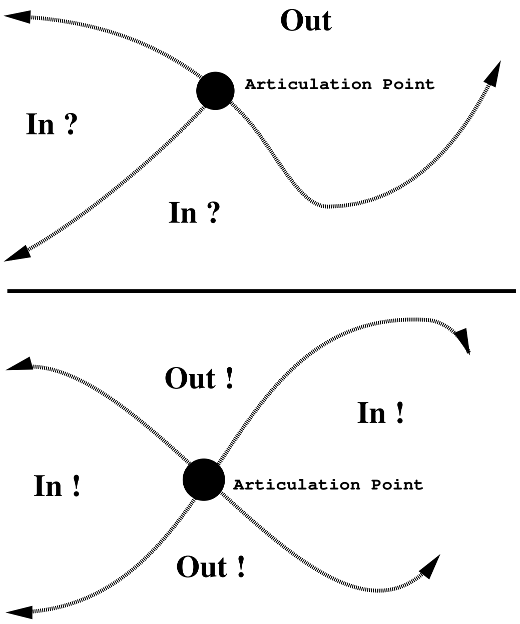

Figure 13: Impossibility of trifurcation.

[3] Christoph M Hoffmann. *Geometric and Solid Modeling: An Introduction.* Morgan Kaufmann Publishers, Inc., San Mateo, California, 94403, 1989.

[4] Robert A. Hummel and S.W. Zuker. On the foundations of relaxation labeling processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(1):267 – 283, 1983.

[5] Alan D. Kalvin. *Segmentation and Surface-Based Modeling of Objects in Three-Dimensional Biomedical Images.* PhD dissertation, New York University, Computer Science Department, 1991.

[6] Daniel B. Karron. The "spiderweb" algorithm for surface construction in noisy volume data. *In Preparation*, 1992.

[7] Martin M. Lipschutz. *Differential Geometry.* Schaum's Outline Series. McGraw-Hill Book Company, New York, New York, 1969.

[8] William E. Lorensen. Triangle decimation. Submitted to SIGGRAPH 92, personal communication, March 1992.

[9] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM Computer Graphics*, 21(4):163–169, July 1987.

[10] Bud Mishra. Notes on solid modeling. Lecture notes, New York University, Computer Science Department, 1991.

[11] Kevin J. Weiler. *Topological Structures for Geometric Modeling.* PhD dissertation, Rensselaer Polytechnic Institute, Computer Science, August 1986.