Semi-Regular Mesh Extraction from Volumes

Zoë J. Wood, Caltech
Mathieu Desbrun, USC
Peter Schröder, Caltech
David Breen, Caltech

Abstract

We present a novel method to extract iso-surfaces from distance volumes. It generates high quality semi-regular multiresolution meshes of arbitrary topology. Our technique proceeds in two stages. First, a very coarse mesh with guaranteed topology is extracted. Subsequently an iterative multi-scale force-based solver refines the initial mesh into a semi-regular mesh with geometrically adaptive sampling rate and good aspect ratio triangles. The coarse mesh extraction is performed using a new approach we call *surface wavefront propagation*. A set of discrete iso-distance ribbons are rapidly built and connected while respecting the topology of the iso-surface implied by the data. Subsequent multi-scale refinement is driven by a simple force-based solver designed to combine good iso-surface fit and high quality sampling through reparameterization. In contrast to the Marching Cubes technique our output meshes adapt gracefully to the iso-surface geometry, have a natural multiresolution structure and good aspect ratio triangles, as demonstrated with a number of examples.

Keywords: Semi-regular meshes, subdivision, volumes, surface extraction, implicit functions, level set methods

1 Introduction

Iso-surface extraction is a fundamental technique of scientific visualization and one of the most useful tools for visualizing volume data. The predominant algorithm for iso-surface extraction, Marching Cubes (MC) [36], computes a local triangulation within each voxel of the volume containing the surface, resulting in a uniform resolution mesh. Often much smaller meshes adequately describe the surface since MC meshes tend to oversample the iso-surface, encumbering downstream applications, e.g., rendering, denoising, finite element simulations, and network transmission. These challenges can be addressed through multiresolution mesh representations.

We present a method for the *direct* extraction of an adaptively sampled multiresolution iso-surface mesh with good aspect ratio triangles. The multiresolution structure is based on adaptive *semi-regular* meshes, well known from the subdivision setting [54]. A semi-regular mesh consists of a coarsest level triangle mesh which is recursively refined through quadrisection. The resulting meshes have regular (valence 6) vertices almost everywhere. Adaptivity is achieved through terminating the recursion appropriately and enforcing a restriction criterion (triangles sharing an edge must be off by no more than one level of refinement). Conforming edges are used to prevent T-vertices (see Fig. 1). Because of their special structure such meshes enjoy many benefits including efficient compression [25] and editing [55] (among many others). Since the mesh hierarchy is represented through a forest of quad-trees, implementation is simple, elegant, and efficient. Figure 1 shows an example of a multiresolution semi-regular mesh extracted from a distance volume with our algorithm.

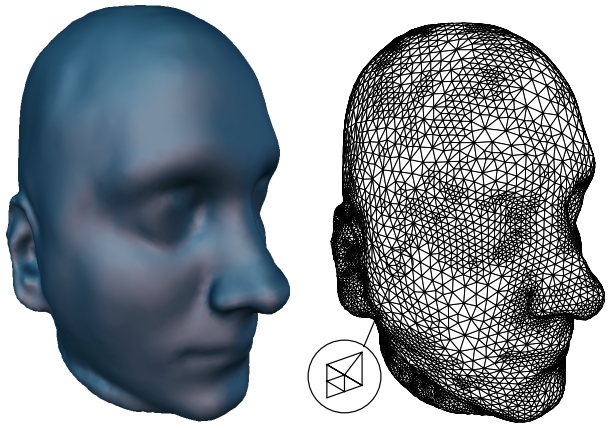

Figure 1: *Example extractions of adaptive semi-regular meshes from volumes using our algorithm.*

1.1 Contributions

We propose an algorithm for the extraction of semi-regular meshes directly from volume data. In a first step a coarse, irregular connectivity mesh with the same global topology as the iso-surface is extracted (Fig. 2, left). This stage works for arbitrary scalar volumes with well defined iso-surfaces and has a small memory footprint. In a second step the mesh is refined and its geometry optimized (Fig. 2, right). Here we require a distance volume for the desired iso-surface. During refinement, aspect ratios and sizes of triangles are controlled through adaptive quadrisection and *reparameterization forces*. Since our algorithm proceeds from coarser to finer resolutions, simple multi-scale methods are easily used. In particular we solve successively for the best fitting mesh at increasing resolutions using an upsampling of a coarser solution as the starting guess for the next finer level. In summary, novel aspects of our algorithm include:

- direct extraction of semi-regular meshes from volume data;
- a new and fast method to extract a topologically accurate coarse mesh with low memory requirements, suitable for large datasets;
- an improved force-based approach to quickly converge to a refined mesh that adaptively fits the data with good aspect ratio triangles.

1.2 Related Work

Traditional Methods and Multiresolution proceed by first constructing an MC mesh and then improving it through simplification [20] and/or remeshing [11, 29, 33, 28, 19]. Common mesh simplification algorithms have large memory footprints [21, 15] and are impractical for decimating meshes with millions of samples (see [35, 34] to address this issue). In addition, simplification algorithms create irregular connectivity meshes with non-smooth parameterizations. These cannot be compressed as efficiently as semi-regular meshes [25] leading to the need for remeshing. In

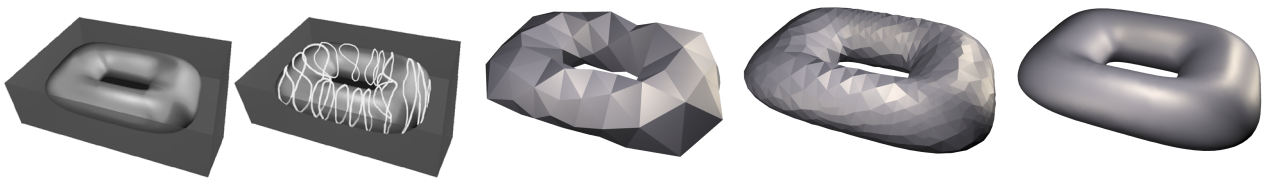

Figure 2: *Overview of our algorithm (left to right). Given a volume and a particular iso-value of interest a set of topologically faithful ribbons is constructed. Stitching them gives the coarsest level mesh for the solver. Adaptive refinement constructs a better and better fit with a semi-regular mesh.*

contrast we wish to directly extract multiresolution meshes with a smooth parameterization.

Alternatively multiresolution can be applied to the volume followed by subsequent MC extractions [50, 2]. Unfortunately, it is difficult to guarantee the topology of the mesh extracted from the simplified volume, e.g., small handles will disappear at various stages of the smoothing step, causing a change in the topology of the extracted mesh (see [16] for a new solution). In contrast our approach constructs a topologically accurate semi-regular mesh at every stage of the algorithm.

Deformable Model Approaches define the surface as the minimum (thin-plate) energy solution induced by a suitable potential function [40, 23, 38, 43, 28]. The second stage of our algorithm proceeds similarly with the important distinction that we exert specific control over the connectivity of the mesh to achieve a semi-regular structure and we use a balloon [5] approach coupled with a novel *reparameterization* force. Similar to previous approaches the initial mesh for our finite element solver must have the correct topology, however almost all previous approaches rely on user input to determine the appropriate global topology for the initial mesh [40, 43, 28, 38]. The largest advantage of our algorithm is our ability to extract a surface of arbitrary topology without any input from the user. Solvers which accommodate topological modifications are possible, but rather delicate [31, 39]. Instead we opt for a robust algorithm which *automatically* extracts a surface with the correct global topology from the volume data *without recourse to MC*.

Topological Graphs can be constructed to encode the topology of a surface. Our algorithm uses the adjacency relationships of the voxels in the volume to traverse the surface and record its connectivity in a graph that is topologically equivalent to the MC mesh for the same volume. This traversal and graph construction is related to work done by Lachaud [30] on topologically defined iso-surfaces. However, unlike Lachaud we do not triangulate the entire graph. Instead, our algorithm extracts a coarse mesh by eliminating redundant regions of the graph where the topology does not change.

Morse Theory and Reeb Graphs are also concerned with coding the topology of a surface [47, 45, 46]. However, neither method uniquely identifies the embedding of the surface in space, potentially leading to ambiguities in the topology coding. Work done on surface coding and Reeb graph construction by Shinagawa, using contours defined by a height function, resolves these ambiguities through requiring apriori knowledge [45, 46] of the number of handles. In contrast the topological graph we construct from the contours of the wavefront propagation *uniquely* determines the topology of the surface with *no* apriori information (for more details and a proof see [53]).

Distance Iso-contours are critical in our approach. We use ideas from level set methods on manifolds [26, 44] and discrete distance computations [32, 49]. Note that we compute these distances on implicitly defined (through the volume) surfaces, not on meshes. Specifically, we use the connectivity relationship of voxels in the volume to build a graph representing the surface. Distances are then propagated on this graph, creating a discrete distance graph. Iso-distance contours in this graph are used to correctly encode the topology of the surface without ever constructing an explicit mesh as in the MC algorithm.

Signed Distance Volumes are required by our solver, though the initial topology discovery stage runs on any volume with well-defined iso-contours. A signed distance volume stores the shortest signed distance to the surface at each voxel which is useful in a variety of applications [7, 6, 17, 42, 51]. Distance volumes are constructed by computing the shortest Euclidean distance within a narrow band around the desired iso-contour and then sweeping it out to the remaining voxels using a Fast Marching Method [44]. Distance volumes can easily be generated for a variety of input data. For example, distance volumes for MRI and CT data are computed by fitting a level set model to the desired iso-surface, creating a smooth segmentation of the input data [37, 52].

2 Coarse Mesh Extraction

In order to construct a topologically accurate coarse representation of a given iso-surface we slice the surface along contours that capture the overall topology. This concept is similar to representing a surface with a Reeb graph, which uses contours defined by a height function. The latter leads to ambiguities which we avoid by using contours of a distance function defined *on* the iso-surface. Examining the way these geometric contours are connected, we can always uniquely encode a topological graph of the iso-surface. This is achieved by discarding topologically redundant cross-sections, i.e., those where surface topology can not change.

Background Before we explain the details of this approach, recall some important theorems and definitions from Geometric Topology [41]. First, the topology of a 2-manifold M (closed polyhedral surface) is completely determined by its genus:

$$\chi(M) = V - E + F = 2(1 - g)$$

where χ is the Euler characteristic, V the number of vertices, E the number of edges, F the number of faces and g the genus. We use this fact and two related theorems:

- the Euler characteristic of an entire polyhedron can be decomposed into the sum of the Euler characteristics of smaller regions whose disjoint union is the polyhedron;
- the Euler characteristic of any given 2-manifold, or subset of a 2-manifold is invariant, *regardless of how the surface is triangulated*.

Given these facts, it is easy to see that topology can be captured accurately by selecting contours where the Euler characteristic of the associated region will change the genus of the surface. This selection is based on decomposing the surface into a combination of a few simple primitives:

1-sphere: A 1-sphere J is a set homeomorphic to a unit circle with $\chi(J) = 0$.

2-cell: A 2-cell D is a set homeomorphic to a disk with $\chi(D) = 1$.

For example, we can decompose a sphere into two 1-spheres (contours), two 2-cells (disks), and the triangulation between the two contours (which we call a *ribbon*) that respects the orientation of the original surface (see Fig. 3). Consider the combined Euler characteristic of these regions. As stated in the definitions, the Euler

characteristic of each of the two disks equals 1 while the Euler characteristic of the contours equals 0. Given this, and since the genus g of the sphere is 0, we deduce that the Euler characteristic χ of this ribbon is 0. This type of decomposition gives a general way

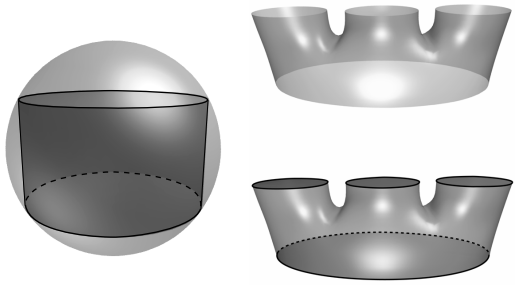

Figure 3: *On the left is a sphere decomposed into a ribbon and two disks. On the right (top) is a n-to-1 ribbon. On the right (bottom) is the closed ribbon, making it homeomorphic to a sphere*

to compute the Euler characteristic and thus the genus of a surface: separate the surface into regions that either are redundant or important with respect to the topology based on the Euler characteristic of those regions. It is important to note that we do not compute the Euler characteristic on a triangulated mesh and instead we rely on the implicit representation of the surface in the volume data.

Volume Setting Specifically, consider an implied surface intersected by a Cartesian grid. This intersection and the entire grid can be represented by tuples $(i, F(i))$, where i is a point in 3D space and $F(i)$ is the scalar value of the distance volume at that point in space. Without loss of generality we assume that the surface is the zero iso-contour of the volume. The surface will be pierced by the edges and faces of the Cartesian grid, creating a collection of patches each of which we denote as a *Surfel*, for surface element (Fig. 4, left). The edges of the grid which pierce the surface are denoted *active edges*. Their endpoints lie on opposite sides of the surface. Edge endpoints are considered either outside the surface if $F(i) \geq 0$, or inside the surface if $F(i) < 0$, thus edge endpoint cannot degenerately lie *on* the surface. The active edges intersect the surface at points called *nodes*. For the case of an iso-surface embedded in volume data, the resulting Surfel graph will be regular in the sense that all nodes are valence four. This Surfel graph is never triangulated, only its connectivity information is used to build the topological graph of the surface.

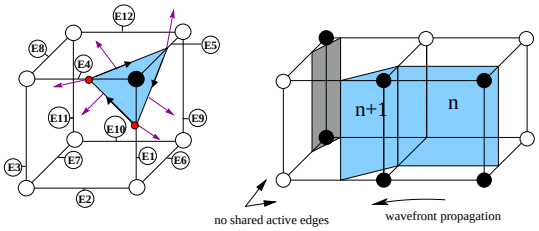

Figure 4: *Dark grey arrows indicate how to follow active edges from a given Surfel (left). On the right, the Surfel with distance n will propagate across its active edges the distance $n + 1$ to connected Surfels. Note that the other Surfel in this voxel will only receive a distance when the wavefront reaches it.*

Given this setting we return to the original goal of generating slices to subsample the surface while retaining the original topology. In order to code the Euler characteristic we traverse the Surfel graph and establish connectivity relationships between all the regions of the surface. Connectivity information is already implicitly represented by voxel adjacency in the volume. The construction of this graph has two parts. First we construct a distance tree, similar to propagating a wavefront across a surface in the geodesic setting. The frontier of the wavefront at any given distance will be a contour that geometrically fits the surface. Next we augment the distance tree by establishing connectivity between Surfels of the same distance, similar to constructing iso-contours for geodesics on the underlying iso-surface.

2.1 Wavefront Propagation and Distance Tree

The first step in our approach is to construct a topological distance tree by enumerating the Surfels through a wavefront-like propagation of Surfel distance. First consider the following graph representation of the surface: G is a graph, such that each vertex $s \in G$ is a Surfel and $n \in G$ is *1-node adjacent* to s if n shares a node with s. The edges of G are defined as the connections between each $s \in G$ and its 1-node adjacent neighbors. The distance tree D is induced by running Dijkstra's algorithm on G starting from any source Surfel s, with edge weights all equal to one. This propagates a distance [1] to all Surfels and constructs a tree such that:

- Each Surfel is 1-node adjacent to its parent in the tree;
- The shortest distance from a Surfel to the root is the depth of the Surfel in the tree hierarchy.

Surface Wavefront Propagation Any voxel that the surface passes through can serve as the root Surfel of our distance tree. From there, we construct the tree by enumerating the Surfels using Dijkstra's algorithm (Fig. 5, left). This propagation between adjacent Surfels can be done efficiently using active edges of the initial Cartesian grid to determine Surfel neighbors. The distance tree requires only a compact data structure and is represented by storing an additional integer and pointer per Surfel for each voxel as indicated by Figure 5(left). Each voxel typically has a single Surfel but up to four Surfels may be associated with a single voxel. This is of no consequence to the algorithm since we propagate the wavefront only across active edges (Fig. 4). Ambiguities can arise when using only the eight corners of a voxel to determine an ordering of the active edges but are easily avoided by selecting one consistent solution [3].

2.2 On-the-fly Construction of Topological Graph

The next step in the algorithm constructs a topological graph by augmenting the distance tree. This is done by collecting Surfels of the same distance into continuous ribbons, representing strips of the surface topology. The process of linking ribbons requires that we start with a given Surfel of distance n and traverse pairs of active edges—*faces* of the voxel bounding the given Surfel—in an ordered manner until we find another adjacent Surfel of the same distance n. As the ribbon is traversed, we enumerate an in-ribbon ordering for all the Surfels to assist in triangulation of the coarse mesh (see Fig. 4).

Constructing Ribbons To construct a consistent ordering within the ribbons, we use an idea very similar to work done on encoding a digital region boundary [13] and digital surface tracking [18]. Since the edges of each Surfel are ordered (see Fig. 4), a consistent traversal ordering can be established. For example, as shown in figure 4, this Surfel could be identified as: {E1, E4, E5}. During ribbon construction for the distance n, if this Surfel is reached by crossing the active edge pair {E1, E4}, first the next active edge pair {E4, E5} would be checked to see if the neighboring Surfel incident on this edge pair is the same distance. If it was not, the next pair would then be checked. One of these neighboring Surfels must be the same distance by definition of our wavefront propagation. The predecessor of the present Surfel must have at least one other successor which is 1-node adjacent to the present Surfel. This process of linking neighboring Surfels is continued

[1] When we refer to Surfel distance, we mean the path distance associated with the edges of G, i.e. each Surfel is distance 1 from its 1-node adjacent neighbors. This is a discrete, Surfel based distance.

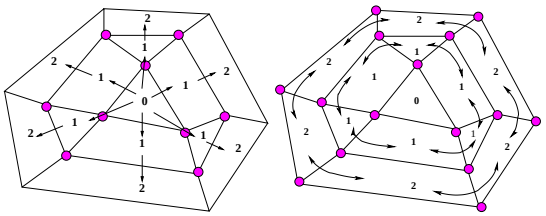

Figure 5: *Small portion of the distance tree overlayed on some Surfels (left). The Surfel labeled 0 is 1-node adjacent to all the Surfels labeled 1 since it shares at least one grey node with each of them. On the right is an example of 2-node adjacency between Surfels of the same distance as required in ribbon construction.*

until the initial Surfel of distance n is found, creating a continuous contour of the surface.

For a given distance n, after a single ribbon is constructed, we check to make sure that all the valid Surfels of distance n are part of a ribbon. If not, the ribbon construction is restarted with one of the unused Surfels at level n. This process continues until all Surfels are incorporated in the topological graph structure. Each distinct ribbon of the same distance is assigned a distinct branch name. Consequently, if there are multiple ribbons at level n, they will have unique branch names, either derived from their parent or assigned uniquely for completely new branches.

Cleanup of Ribbons If distance is propagated naïvely, ribbons could have tails (Fig. 7). Tails are large or small dead-ends of the wavefront. A dead-end of a wave front occurs when the wavefront runs into itself. Tails do not provide additional topological information [53] and are removed by pruning them from the distance tree during distance propagation: if a voxel cannot propagate its distance forward because all of its neighbors have already been visited, it is pruned from the distance tree.

The Topological Graph This construction guarantees that the topological graph has particular properties. Specifically, our topological graph is a representation of all the Surfels such that:

- All of the properties of a distance tree hold;
- Every Surfel has 2-node adjacency with exactly two other Surfels of the graph that are of the same distance and the same branch number — i.e. they share an edge (see Fig. 5, right).

These criteria establish that our topological graph is essentially composed of a collection of continuous contours of the surface. The dual of these contours are homeomorphic to a 1-sphere and combined with the root Surfel and leaf ribbons (homeomorphic to 2-cells), can be used to completely code the topology of the surface.

2.3 Coarse Mesh Construction

The topological graph provides everything needed to build the coarse mesh. In order to have a good coarse sampling of the surface, we only include the smallest number of ribbons necessary: Ribbons essential for coding topology are those inducing topological *events*. A ribbon represents a topological event only if it contributes to a change in the Euler characteristic of that region of the surface.

Ribbon Classification Consider the Euler characteristic of the three types of ribbon adjacencies:

Endcaps: A root Surfel or a leaf ribbon: these are 2-cells with $\chi = 1$.

1-to-1 ribbon : The most common case for a ribbon comprised of two connected 1-spheres with $\chi = 0$ (by the same argument used in section 2).

1-to-n ribbon (and vice-versa) : The regions of the surface that represent a possible change in the topology. For these branchings the Euler characteristic can be computed similar to the 1-to-1 ribbon case: close the different branches by endcaps to get a topological sphere. Hence for 1-to-n ribbons (see Fig. 3) we have $\chi = 1 - n$.

For example, in a torus there would be one 1-to-2 ribbon where the graph traversal first encounters the hole of the torus and one 2-to-1 ribbon where the hole ends. Both of these events need to be captured in order to construct the correct topology of the torus. In contrast, the surface region between these two important events is a sequence of adjacent 1-to-1 ribbons for each branch which can be discarded without changing the topology of the surface.

Since these adjacency relationships are completely determined by ribbon neighbors, ribbon construction *and* event detection can be performed in a sweep algorithm. Once the ribbons at level n are constructed, event detection is performed by walking along the previous ribbons at level $n-1$ to see if an event ribbon was encountered. For example, for each of the Surfels in ribbons at level $n-1$, we check that their descendants have the same branch number. If not, a 1-to-n ribbon has been found. Likewise by keeping track of the branch numbers already seen, a n-to-1 ribbon can be detected when different predecessor ribbons are connected to the same descendant ribbon. Finally, if a ribbon has no valid descendant ribbons, it is saved as an endcap.

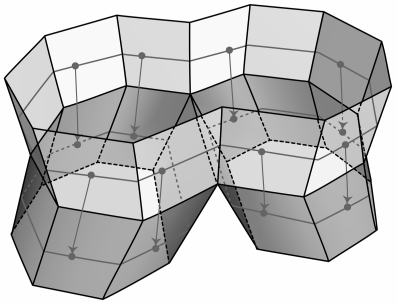

Figure 6: *1-to-n ribbon detection (n-to-1 ribbon detection is similar but inverted).*

The desired coarseness of the mesh can be controlled by adding criteria for ribbon selection. For example, consider a requirement that the initial mesh exhibit good aspect ratio triangles. This can be achieved by selecting ribbons at multiples of some integer distance w and changing the sampling density within the ribbons to also be of average distance w.

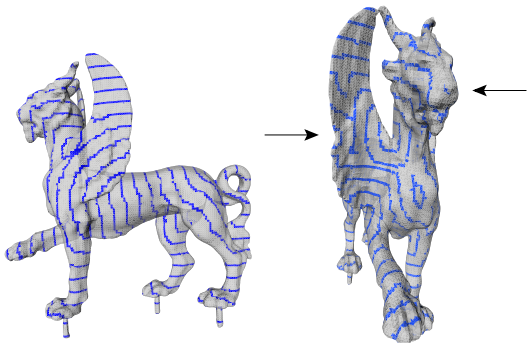

Figure 7: *On the left is the the distance ribbons for the feline dataset. The source Surfel is near the feline's tail. On the right is subsampling of the unmodified distance ribbons. There are two visible tails on the left wing and on the nose.*

Mesh Construction At this point, we have a list of all contours of the surface which are required for tiling a good coarse approximation of the final surface. The final step of our algorithm is related to contour stitching [1, 14, 12]. However, since we work within the

framework of the volume data we do not face the traditional correspondence problems of contour stitching. Specifically, the volume data and the topological graph prevent ambiguities about inter-contour connections.

Ribbon Subsampling and Shortest Distance Projection The general procedure is to subsample each ribbon along its length to convert it into a coarse contour of edges and vertices to be triangulated with adjacent contours. Adjacent contours are connected to one another by projecting ribbon samples to the next saved ribbon (see Figure 6). The projection step may result in samples being too close or too far away from one another due to changes in the geometry of the iso-surface. In this case we can adjust the number of samples to accommodate the density change by snapping close points together, or inserting a midpoint sample. The samples on both contours are enumerated in corresponding order to facilitate triangulation. Endcaps are evenly subsampled and connected to a central point.

Stitching It is easy to tile two contours that have a one-to-one correspondence in their sample enumeration. The general approach of our algorithm is to *break* the ribbons into one-to-one correspondence and then use bridges between adjacent connected ribbons to correctly model the topology of the surface. Thus 1-to-n ribbons and n-to-1 ribbons are conceptually handled by "breaking" them into n pairs of 1-to-1 ribbons with conforming bridges between appropriate segments (Fig. 8). This is done by making a pass around the larger ribbon to find if two neighboring samples have been projected from different predecessor ribbons, in which case they are stored to make the conforming bridge (Fig. 8). The following pseudo code outlines the stitching algorithm:

```
For all saved ribbons
    //process all m ribbons of distance n
    If a ribbon is not sampled
        evenly sample at intervals of w Surfels
    //else the ribbon may already be sampled from previous projection
    For each sample of the current ribbons
        Project down to next saved ribbons
    //check the spacing for the new samples
    For each Surfel of the child ribbons
        If samples too close: snap to one sample
        If samples too distant: insert a midpoint

    allocate sample lists for breaking ribbons into 1-to-1
    top-lists[m], bottom-lists[n] //n is the number of child ribbons
    //put the current and projected samples into the appropriate lists
    Traverse the current ribbon's samples
        If the current ribbon is a 1-to-n ribbon
            branch = child sample's branch number
            Put the current sample in the top-list[branch]
            Put the associated child sample in the bottom-list[branch]
        Else if the current ribbon is a n-to-1 ribbon
        //same procedure but branch = current ribbons branch number
    Triangulate the ordered samples of the corresponding top and bottom lists

    //check for edges to make conforming bridge
    If the current ribbon is a 1-to-n ribbon
    Traverse the current ribbon's samples
        If two neighbor samples have children with different branch numbers
            Store the samples until the corresponding pair is found
        Triangulate the four samples to make the conforming bridge
    Else if the current ribbon is a n-to-1 ribbon
    //same procedure but traverse the child ribbon's samples
```

It is worth noting that there is a case equivalent to a n-to-1 ribbon immediately followed by a 1-to-m ribbon. Due to the discrete nature of the samples this can appear as an n-*to*-m ribbon. This case is easily identifiable and tagged in the event detection: two child ribbons will have more than one parent in common. The previous pseudo-code applies to this special case as well.

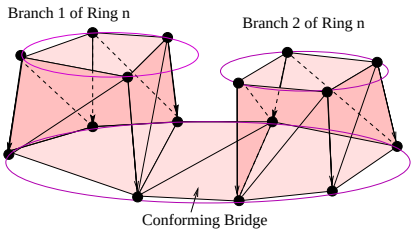

Figure 8: *Stitching example of a n-to-1 ribbon.*

2.4 Discussion

One of the benefits of this approach is the low memory overhead for the topological graph representation. In the case of an $O(n^3)$ volume the storage requirement for the distance tree is on average $O(n^2)$, as it depends on the size of the surface. The only other data that we need to store for generation of the coarse mesh is dependent on the ribbons of the topological graph which is approximately $O(n)$. Memory overhead for ribbons is minimized by keeping only, (i) the ribbons selected to be part of the coarse mesh; (ii) the last ribbon constructed and (iii) the current ribbon, which is being evaluated for possible selection. Although both our algorithm and MC use total storage of $O(n^2)$ on average, our algorithm has a more compact runtime footprint than a typical MC implementation. In particular, a time efficient implementation of the MC algorithm typically keeps information for all the voxels on the surface. This requires storage of three float values associated with each edge intersection (up to 36 floats per voxel) and three integers per face (up to 12 integers per voxel). In contrast, our algorithm does not require such detailed storage and only requires one integer and one pointer per voxel. Furthermore, we have presented the algorithm as if a distance value is permanently stored for each Surfel. This is only true conceptually, as distance values can be stored temporarily and only for voxels on the *frontier* region of the sweep. The frontier region of the sweep is the region of the surface between the last ribbon selected to be a part of the mesh and the current ribbon being evaluated. In addition, assuming that a subsequent simplification is performed on the MC mesh, typical algorithms will use at least an additional copy of the finest mesh and a sorted list of vertices, resulting in an even larger memory footprint than our entire coarse extraction routine.

3 Multi-Scale Force-based Solver

Once a coarse mesh with the correct topology is found, the next step of the algorithm consists of turning this initial mesh into a hierarchical triangulation fitting the data with suitable sampling densities and well shaped triangles. To solve for the iso-surface one may consider the signed distance function of the volume as a potential field and search for the minimum potential solution [24, 23, 22, 43, 38]. Unfortunately, this approach has a significant drawback: the trade-off between closeness to the data and the smoothness of the solution is hard to tune. In essence, smoothness of the solution and faithfulness to the desired goal surface compete with each other. Too much regularization will lead to smooth, unfit surfaces, while not enough regularization will lead to convergence difficulties. In both cases, the overall speed and accuracy is very dependent on fine tuning of parameters. This has been partially addressed by scheduling the regularization as decreasing in time [22]. Such strategies help, but still require careful tuning of parameters on a case by case basis.

The above approaches use the gradient of distance whose computation is notoriously unstable, especially in the presence of noise. For this reason we have chosen to use the distance itself. The current mesh approximation locally inflates or deflates based on the distance to the zero-contour. The direction of (local) motion of the mesh is given by its local normal, while the magnitude (and sign) of motion are determined by the distance function itself, similar to [40]. This approach, inspired by work in image processing [5],

has already been used with success in the context of active implicit surfaces [8, 51]. As a novel element we add a reparameterization technique to control triangle shapes and their variation across the surface. In this way, we obtain adaptive sampling and well shaped triangles without introducing forces which compete with the interpolation constraints. Since the meshes are refined through adaptive quadrisection we have a natural multiresolution structure which we exploit directly for an efficient multiscale solver. Our setup gives rise to a number of different force terms detailed below. External forces minimize the distance between the mesh and the zero-contour of the data. Internal forces arise from the reparameterization terms.

3.1 External Forces

We begin by considering the force acting on a single triangle before giving the actual equations for the net force on a vertex in the mesh. Following the balloon strategy, we define the force acting on a triangle T of our mesh as being along the normal of the triangle, with a sign and a magnitude depending on the surface integral of the distances d between the triangle and the actual zero-contour C:

$$F_T = \mathbf{n}_T / \mathcal{A}_T \int_{x \in T} d(x, C)\ dx$$

where $\mathbf{n}_T$ is the triangle normal and $\mathcal{A}_T$ is the area of T. The integral of the distance across the face can be computed exactly in the volume setting, since we assume that the distance varies linearly across a given voxel. In practice this is overkill and we use a much cheaper sampling criterion. Each triangle face is randomly sampled with a uniform distribution whose area density depends on the total area of the triangle. First, however, we compute the variance of the distance for a small number of uniform samples in order to short circuit unnecessary sampling. This results in quicker force computations, while preserving the quality of the approximation. Note that the minimum bound on the discretization rate is of the order of a voxel size, since everything is assumed to vary linearly within a voxel. Therefore, we use the following simple sampling strategy:

```
Temporarily quadrisect the triangle T into four small triangles t_i
For each t_i
    E[d] += d_i = DistanceAtBarycenter(t_i)
    E[d^2] += (d_i)^2
m_T = 4     //the number of samples
//calculate the variance V_T[d] of these distances
V_T[d] = E[d^2] - (E[d])^2
If V_T[d] ≥ δ
    m_T = A_T / a_vf     //a_vf = area of a voxel face
    For each m_T
    //stochastically sample the triangle with a uniform distribution
    E[d] += DistanceAtRandomSample(T)
```

The variance of a discrete set of distances is computed in the standard way $V_T[d] = E[d^2] - E[d]^2$, where E denotes the mean of its argument. A more sophisticated method, using fully adaptive sampling depending on variance, can be derived, but this simple approach has proved sufficient and has the advantage of being very efficient. The final net force on a triangle is be given by the above mean of the distances

$$\mathbf{F}_T = \mathbf{n}_T E[d].$$

The solver requires forces acting on vertices. To arrive at these we use the above sample points to compute integrals for each vertex by integrating over all incident triangles, weighting each sample point with its respective barycentric coordinate. Every sample point within a triangle contributes to the force integrals associated with its corner points as follows:

$$\begin{aligned} &1/m_T\ \mathbf{n}_T d(x_i, C)\ \phi_j(x_i) \\ &1/m_T\ \mathbf{n}_T d(x_i, C)\ \phi_k(x_i) \\ &1/m_T\ \mathbf{n}_T d(x_i, C)\ \phi_l(x_i) \end{aligned}$$

where $x_i \in T$ is the sample location; (j, k, l) are the corners of T; and the ϕ give the barycentric coordinate of x_i with respect to j, k, and l respectively. Effectively we are using piecewise linear finite elements and stochastic sampling to evaluate the associated integrals. In the implementation we simply iterates over all triangles and accumulates the integrals at each vertex.

With this scheme, faces will tend to move towards the zero-contour. If the mesh is coarser than the small details of the zero-contour, it will settle in an optimal position, smoothing the details. The finer the mesh is, the better the fit will be. As mentioned in [23], we also noticed that vertices tend to align with sharp features on the zero-contour.

3.2 Internal Forces

Internal forces are usually added as a regularizing term, to guide the minimization to a desirable local minimum. In our approach internal forces are mainly used to ensure good aspect ratios for the faces and to keep the sampling across the surface smoothly distributed. Usually, springs of zero rest length and identical stiffness are used to keep sample points from clustering locally and ensure uniform sampling [23]. Instead we define *reparameterization* forces which act similarly, but only along the local parameter plane, not in space.

Decoupling Smoothing and Reparameterization In recent work on mesh smoothing [48, 9], the Laplacian operator has been used extensively to denoise triangulated surfaces, using the approximation:

$$\mathcal{L}(x_i) = \frac{1}{m} \sum_{j \in N_1(i)} x_j - x_i,$$

where x_j are the neighbors of vertex x_i, and $m = \#N_1(i)$ is the number of these neighbors (valence). Note that this definition is equivalent to springs with zero rest length whenever the valence is constant throughout the mesh. This Laplacian of the mesh at a vertex can be broken down into two orthogonal components:

- a component normal to the surface, creating *shape smoothing*
- and a component in the tangent plane, fairing the *parameterization* of the mesh.

The normal vector to the surface can be found easily by normalizing the curvature normal vector $\mathbf{K}$ [9, 10]:

$$\mathbf{K}(x_i) = \frac{1}{2\mathcal{A}} \sum_{j \in N_1(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(x_i - x_j). \qquad (1)$$

For arbitrary connectivity meshes numerical evidence shows that no spurious drifting artifacts appear when the surface is modified only in the direction of $\mathbf{K}$ [9]. This decomposition into normal and tangential components separates motion into one component changing shape and one changing the parameterization. We are only interested in the latter.

Reparameterization as Tangential Laplacian Smoothing In our context shape smoothing would act *against* the external forces trying to fit the initial data. Thus we are only interested in the tangential motion of Laplacian smoothing in order to improve the quality of the discretization. This reparameterization force is defined as

$$\mathbf{T}(x_i) = \mathcal{L}(x_i) - (\mathcal{L}(x_i) \cdot \mathbf{n})\mathbf{n}, \qquad (2)$$

where $\mathbf{n}$ is the normalized $\mathbf{K}$ of Equ. 1. We also use the second Laplacian operator $\mathcal{L}^2$ [27, 9] to ensure a smoother variation of sampling rate over the surface, and suppress the normal component in the same way. By proceeding as described, we keep internal and external forces distinct, thus simplifying parameter choices.

3.3 Refinement Strategy

After an optimal solution has been found for a given mesh, we evaluate a refinement criterion over each triangle. Any triangle failing the criterion is quadrisected. This hierarchy is naturally maintained in a forest of quadtrees, one tree for each original coarsest level triangle. The solver is run anew after refinement.

The two criteria used to determine if a triangle should be refined are curvature and variance of distance. If the variance of the distance samples for a given triangle is too high, the surface underneath this particular triangle must have high curvature, and the triangle requires refinement. Using a user supplied threshold ϵ_V all triangles T with $V_T[d] \geq \epsilon_V$ are refined.

Additionally we also test the curvature of the current mesh to ensure good discretization in highly curved areas. If the three vertices of a triangle have too high a curvature compared to the area of the triangle, our solver refines the triangle to better adapt to the local geometry. For generality, we add a condition to deal with sharp features in the volume data: we invalidate the test on curvature if the variance of sampled distances is too small. Refinement will be avoided if we are already describing the surface adequately. Therefore, our second refinement criterion for a triangle $T = (x_i, x_j, x_k)$ can be written:

$$(|\mathbf{K}(x_i)| + |\mathbf{K}(x_j)| + |\mathbf{K}(x_k)|)\mathcal{A}_T \geq \epsilon_\kappa \text{ and } V_T[d] \geq \frac{\epsilon_V}{10}$$

where ϵ_κ, the maximum discrete curvature, is a user-defined value. The choice of $\epsilon_V/10$ seems reasonable in all our tests, but could be defined by the user if needed, depending on the prevalence of high frequency detail in the iso-surface. It is worth noting that ϵ_V can be viewed as a smoothing factor. For example if the user wants a smoothed version of the surface they can set ϵ_V to a higher number and the system will stop after reaching a solution with fewer triangles to approximate the surface.

3.4 Overall Solver Algorithm

Once forces have been computed for every vertex in the current mesh, vertex positions are updated through an explicit dynamics step:

$$x_i^{(t+\delta t)} = x_i^{(t)} + F_{x_i}\delta t$$

advancing the mesh in time until the approximation error does not decrease further. When advancing the mesh a restriction must be placed on the time step δt to satisfy the Courant condition: the velocity of change must not travel faster than the minimum detail in the system. This condition is simple to compute in our system and as $\delta t = m_e/M_f$, where m_e is the minimum edge length and M_f the maximum force. After a step is taken the refinement criteria are evaluated and quadrisection is performed as needed. Subsequently we solve again until convergence and continue this process until the user supplied error criteria are satisfied.

The behavior of the solver is controlled by the relative weightings of distance and reparameterization forces. We have found a factor of 2 in favor of the distance forces to work reliably for a wide variety of data sets. Similarly time steps of $\delta = 0.1$ and error thresholds of $\epsilon_\kappa = 15$ and $\epsilon_V = 10^{-4}$ have proven to work well without the need for tuning. To make the error criteria scale invariant we consider the object to occupy the unit cube.

4 Results

We have applied our algorithm to a variety of datasets and compared the results with MC reconstructions as "ground truth." Some of these are shown in Figure 9.

The top sequence illustrates the case of a MRI dataset (128^3) which was segmented through a level set method. Construction of the coarsest mesh (186 triangles) took .5 seconds. The intermediate

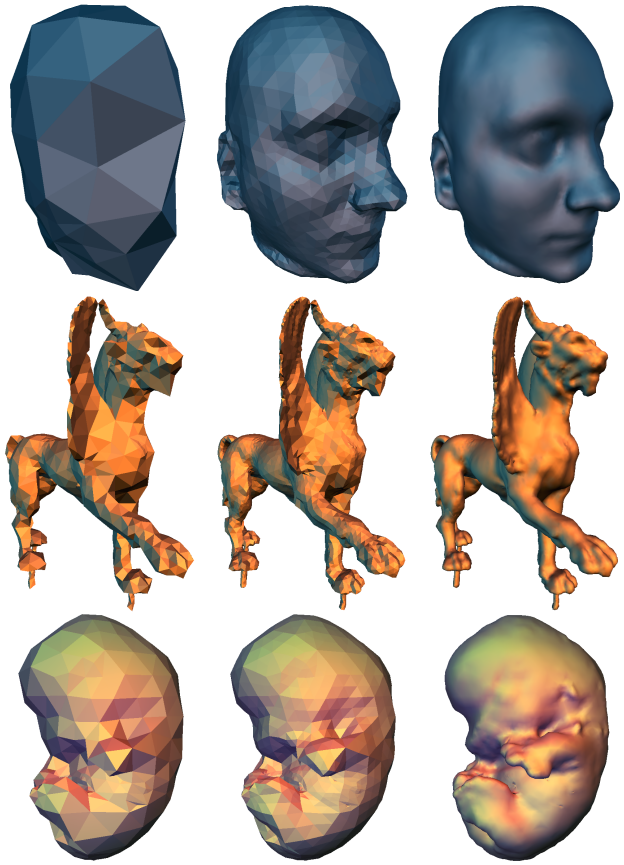

Figure 9: *Reconstructions performed with our algorithm on MRI datasets (top and bottom) and a 3D scanner generated distance function (middle). The coarsest mesh is shown on the left followed by an intermediate adaptive mesh and a final result.*

mesh contains 4810 triangles, while the final mesh has 21360 triangles. Using Metro [4] to compare our reconstruction against the MC mesh (58684 triangles) we find a relative L^2 error of $1.8*10^{-4}$ (Fig. 10). The surface is a topological sphere, but requires fairly fine levels of refinement near the ears, attesting to the performance of our solver in the presence of rapidly changing local geometric complexity.

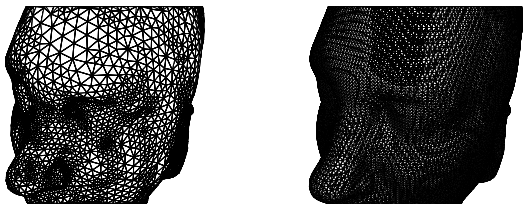

Figure 10: *Comparison between our algorithm output and a MC mesh. The relative L^2 error between these is* $1.8 * 10^{-4}$.

The middle sequence shows an extraction from a 3D scanner generated distance function [7]. The topology of the feline is non-trivial containing numerous handles in the tail region (Fig. 11) and demonstrates the performance of our coarsest level mesh extraction and topology discovery algorithm. It also demonstrates the ability of our solver to resolve fairly fine detail such as the mounting posts on the bottom of the paws. Triangle counts are 3412, 13412 and 46996 respectively (MC: 72685) for an error of $3.3*10^{-4}$. Coarsest mesh extraction time was .34 seconds on a volume of $158*74*166$ voxels.

Finally the bottom row shows another MRI dataset of a mouse

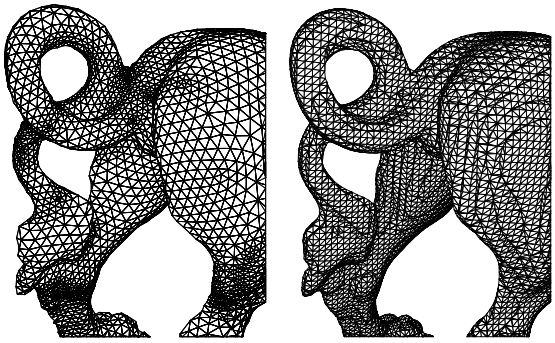

Figure 11: *Tail section of feline showing nontrivial topology. MC extraction on the right, adaptive semi-regular mesh on the left.*

embryo which was segmented with a level set method. The surface has several handles (near both front paws) and numerous concavities. All were resolved successfully. Triangle counts are 1030, 4086, and 26208 respectively (MC: 129670) with an error of $6 * 10^{-4}$. Coarsest level extraction took .78 seconds on a volume of $256 * 128^2$. Typical solver times are on the order of a few seconds for the initial meshes increasing to 4 to 5 minutes for the final reconstructions.

5 Conclusion and Future Work

We have demonstrated a novel algorithm for the capture of iso-surfaces in the form of hierarchical, adaptive semi-regular meshes. It is based on a new approach to construct a coarsest mesh with guaranteed topology approximation of the iso-surface using surface wavefront propagation to discover the topology and ensure that it is represented faithfully. In a subsequent solver step, we use a novel explicit reparameterization force employing tangential components of the first and second Laplacian of the mesh. Thus we do not have to trade off fidelity to the original data and uniqueness of the solution. The resulting meshes have a natural multiresolution structure since they are semi-regular, making them suitable for a variety of powerful digital geometry processing algorithms.

In order to avoid self-intersection problems during the solution process we have so far relied on coarsest meshes which resolve the geometry reasonably well to begin with. It would be desirable to start with the coarsest possible (in the topological sense) initial mesh and counteract any self-intersection problems in the solver itself. Other interesting areas for future work include:

- investigation of the use of multiresolution representations of the volume [16];
- optimization of the solver including adaptive time stepping strategies and automatic selection of the relative weighting for the reparameterization forces;
- application of the topological graph to irregular meshes to code topology.

Acknowledgments This work was supported in part by NSF (ACI-9624957, ACI-9721349, ACI-9982273, DMS-9874082), the NSF STC for Computer Graphics and Scientific Visualization, Alias|Wavefront, and a Packard Fellowship. A very special thanks to Eitan Grinspun, Mark Meyer, and Khrysaundt Koenig for their support and assistance. Thanks to Sean Mauch for an implementation of the Fast Marching Method, Ross Whitaker for the level set segmentation, and Ken Museth for dataset conversion. Thanks to Martin Nguyen for assistance with testing some related ideas. The head MRI data is courtesy the University of Utah's Scientific Computing and Imaging Institute, the MRI dataset of the mouse embryo courtesy the Caltech Biological Imaging Center, and the feline scan courtesy Stanford's Computer Graphics Group.

References

[1] BAJAJ, C., COYLE, E., AND LIN, K. Arbitrary Topology Shape Reconstruction from Planar Cross Sections. *Graphical Models and Image Processing 58*, 6 (1996), 524–543.

[2] BAJAJ, C., AND PASCUCCI, V. Progressive Isocontouring. Tech. Rep. TR 99-36, University of Texas at Austin, 1999.

[3] BLOOMENTHAL, J. An Implicit Surface Polygonizer. In *Graphics Gems IV*, P. S. Heckbert, Ed. Academic Press, 1994, pp. 324–349.

[4] CIGNONI, P., ROCCHINI, C., AND SCOPIGNO, R. Metro: Measuring Error on Simplified Surfaces. *Computer Graphics Forum 17*, 2 (1998), 167–174.

[5] COHEN, L. D., AND COHEN, I. Finite-Element Methods for Active Contour Models and Balloons for 2D and 3D Images. *IEEE Trans. PAMI 15*, 11 (1993), 1131–1147.

[6] COHEN-OR, D., LEVIN, D., AND SOLOMIVICI, A. Three-Dimensional Distance Field Metamorphosis. *ACM Transactions on Graphics 17*, 2 (1998), 116–141.

[7] CURLESS, B., AND LEVOY, M. A Volumetric Method for Building Complex Models from Range Images. *Proceedings of SIGGRAPH 96* (1996), 303–312.

[8] DESBRUN, M., AND CANI-GASCUEL, M.-P. Active Implicit Surface for Computer Animation. In *Graphics Interface (GI'98) Proceedings*, 143–150, 1998.

[9] DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow. In *SIGGRAPH 99 Conference Proceedings*, 317–324, Aug. 1999.

[10] DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. Anisotropic Feature-Preserving Denoising of Height Fields and Bivariate Data. In *Graphics Interface'2000 Proceedings*, May 2000.

[11] ECK, M., DEROSE, T., DUCHAMP, T., HOPPE, H., LOUNSBERY, M., AND STUETZLE, W. Multiresolution Analysis of Arbitrary Meshes. *Proceedings of SIGGRAPH 95* (1995), 173–182.

[12] EKOULE, A. B., PEYRIN, F. C., AND ODET, C. L. A Triangulation Algorithm From Arbitrary Shaped Multiple Planar Contours. *ACM Transactions on Graphics 10*, 2 (1991), 182–199.

[13] FREEMAN, H. Computer Processing of Line-drawing Images. *ACM Computing Surveys 6*, 1 (1974), 57–97.

[14] FUCHS, H., KEDMEN, Z., AND USELTON, S. Optimal Surface Reconstruction from Planar Contours. *Communications of the ACM 20*, 10 (1977), 693–702.

[15] GARLAND, M., AND HECKBERT, P. S. Surface Simplification Using Quadric Error Metrics. *Proceedings of SIGGRAPH 96* (1996), 209–216.

[16] GERSTNER, T., AND PAJAROLA, R. Topology Preserving and Controlled Topology Simplifying Multiresolution Isosurface Extraction. *IEEE Transactions on Visualization and Computer Graphics* (2000).

[17] GIBSON, S. Using Distance Maps for Accurate Surface Representation in Sampled Volumes. In *Proceedings of the 1998 Symposium on Volume Visualization*, 23–30, October 1998.

[18] GORDON, D., AND UDUPA, J. Fast Surface Tracking in Three-dimensional Binary Images. *Computer Vision, Graphics, and Image Processing 45*, 2 (Feb. 1989), 196–241.

[19] GUSKOV, I., VIDIMČE, K., SWELDENS, W., AND SCHRÖDER, P. Normal Meshes. *Proceedings of SIGGRAPH 00* (2000).

[20] HECKBERT, P. S., AND GARLAND, M. Survey of Polygonal Surface Simplification Algorithms. Tech. rep., Carnegie Mellon University, 1997.

[21] HOPPE, H. Progressive Meshes. *Proceedings of SIGGRAPH 97* (1997), 189–198.

[22] HOPPE, H., DEROSE, T., DUCHAMP, T., HALSTEAD, M., JIN, H., MCDONALD, J., SCHWEITZER, J., AND STUETZLE, W. Piecewise Smooth Surface Reconstruction. *Proceedings of SIGGRAPH 94* (1994), 295–302.

[23] HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. Mesh Optimization. *Proceedings of SIGGRAPH 93* (1993), 19–26.

[24] KASS, M., WITKIN, A., AND TERZOPOULOS, D. Snakes: Active Contour Models. In *1st Conference on Computer Vision*, 321,331, June 1988.

[25] KHODAKOVSKY, A., SCHRÖDER, P., AND SWELDENS, W. Progressive Geometry Compression. *Proceedings of SIGGRAPH 00* (2000).

[26] KIMMEL, R., AND SETHIAN, J. Fast Marching Method on Triangulated Domains. In *Proceedings of the National Academy of Science*, vol. 95, 8341–8435, 1998.

[27] KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H.-P. Interactive Multi-Resolution Modeling on Arbitrary Meshes. In *SIGGRAPH 98 Conference Proceedings*, 105–114, July 1998.

[28] KOBBELT, L. P., VORSATZ, J., LABSIK, U., AND SEIDEL, H.-P. A Shrink Wrapping Approach to Remeshing Polygonal Surfaces. *Computer Graphics Forum 18*, 3 (1999), 119–130.

[29] KRISHNAMURTHY, V., AND LEVOY, M. Fitting Smooth Surfaces to Dense Polygon Meshes. *Proceedings of SIGGRAPH 96* (1996), 313–324.

[30] LACHAUD, J.-O. Topologically Defined Iso-surfaces. Research Report 96-20, Laboratoire de l'Informatique du Parallélisme, ENS Lyon, France, 1996.

[31] LACHAUD, J.-O., AND MONTANVERT, A. Deformable Meshes with Automated Topology Changes for Coarse-to-fine 3D Surface Extraction. *Medical Image Analysis 3*, 2 (1999), 187–207.

[32] LAZARUS, F., AND VERROUST, A. Level Set Diagrams of Polyhedral Objects. In *Proceedings of the Fifth Symposium on Solid Modeling and Applications*, 130–140, June 1999.

[33] LEE, A. W. F., SWELDENS, W., SCHRÖDER, P., COWSAR, L., AND DOBKIN, D. MAPS: Multiresolution Adaptive Parameterization of Surfaces. *Proceedings of SIGGRAPH 98* (1998), 95–104.

[34] LINDSTROM, P. Out-of-Core Simplification of Large Polygonal Models. In *Computer Graphics (SIGGRAPH '00 Proceedings)*, July 2000.

[35] LINDSTROM, P., AND TURK, G. Evaluation of Memoryless Simplification. *IEEE Transactions on Visualization and Computer Graphics 5*, 2 (1999), 98–115.

[36] LORENSEN, W., AND CLINE, H. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics (Proceedings of Siggraph '87 21*, 4 (1987), 163–169.

[37] MALLADI, R., SETHIAN, J., AND VEMURI, B. Shape Modeling with Front Propagation: A Level Set Approach. *IEEE Trans. PAMI 17*, 2 (1995), 158–175.

[38] MCINERNEY, T., AND TERZOPOULOS, D. Deformable Models in Medical Image Analysis: a Survey. *Medical Image Analysis 1*, 2 (1996), 91–108.

[39] MCINERNEY, T., AND TERZOPOULOS, D. Topology Adaptive Deformable Surfaces for Medical Image Volume Segmentation. *IEEE Transactions on Medical Imaging 18*, 10 (1999), 840–850.

[40] MILLER, J. V., BREEN, D. E., LORENSEN, W. E., O'BARA, R. M., AND WOZNY, M. J. Geometrically Deformed Models: A Method for Extracting Closed Geometric Models from Volume Data. *Computer Graphics (Proceedings of SIGGRAPH 91) 25*, 4 (1991), 217–226.

[41] MOISE, E. E. *Geometric Topology in Dimensions 2 and 3*. Springer-Verlag, New York, USA, 1977.

[42] PAYNE, B., AND TOGA, A. Distance Field Manipulation of Surface Models. *IEEE Computer Graphics and Applications 12*, 1 (1992), 65–71.

[43] QIN, H., MANDAL, C., AND VEMURI, B. C. Dynamic Catmull-Clark Subdivision Surfaces. *IEEE Transactions on Visualization and Computer Graphics 4*, 3 (1998), 215–229.

[44] SETHIAN, J. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, UK, 1999.

[45] SHINAGAWA, Y., KERGOSIEN, Y. L., AND KUNII, T. L. Surface Coding Based on Morse Theory. *IEEE Computer Graphics and Applications 11*, 5 (1991), 66–78.

[46] SHINAGAWA, Y., AND KUNII, T. L. Constructing a Reeb Graph Automatically from Cross Sections. *IEEE Computer Graphics and Applications 11*, 6 (1991), 44–51.

[47] STANDER, B. T., AND HART, J. C. Guaranteeing the Topology of an Implicit Surface Polygonization for Interactive Modeling. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, 279–286, August 1997.

[48] TAUBIN, G. A Signal Processing Approach to Fair Surface Design. In *SIGGRAPH 95 Conference Proceedings*, 351–358, Aug. 1995.

[49] VERROUST, A., AND LAZARUS, F. Extracting Skeletal Curves from 3D Scattered Data. *The Visual Computer 16*, 1 (2000), 15–25.

[50] WESTERMANN, R., KOBBELT, L., AND ERTL, T. Real-time Exploration of Regular Volume Data by Adaptive Reconstruction of Isosurfaces. *The Visual Computer 15*, 2 (1999), 100–111.

[51] WHITAKER, R., AND BREEN, D. Level-Set Models for the Deformation of Solid Objects. In *Proceedings of the Third International Workshop on Implicit Surfaces*, 19–35, June 1998.

[52] WHITAKER, R. T., AND CHEN, D. T. Embedded Active Surfaces for Volume Visualization. In *SPIE Medical Imaging VIII*, 340–352, 1994.

[53] WOOD, Z. J. Semi-Regular Mesh Extraction from Volumes. Master's thesis, Caltech, Pasadena, California, 2000.

[54] ZORIN, D., AND SCHRÖDER, P., Eds. *Subdivision for Modeling and Animation*. Course Notes. ACM SIGGRAPH, 1999.

[55] ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. Interactive Multiresolution Mesh Editing. *Proceedings of SIGGRAPH 97* (1997), 259–268.