Meshing interfaces of multi-label data with Delaunay refinement*

Tamal K. Dey Firdaus Janoos Joshua A. Levine†

Abstract

In medical imaging, the generation of surface representations of anatomical objects, obtained by labeling images from various modalities, is a critical component for visualization. The interfaces between labeled regions can meet at arbitrary angles and with complex topologies, causing most meshing algorithms to fail. We apply a recent Delaunay refinement algorithm to generate high quality triangular meshes which approximate the interface surfaces. This algorithm has proven guarantees for meshing piecewise-smooth shapes and its implementation overhead is low. We need only provide an oracle for generating intersection points between the surfaces and line segments and a routine for generating the curve network where the interface surfaces meet. Consequently, the approach is applicable to labeled datasets generated from binary segmentations as well as from probabilistic segmentation algorithms. We show the effectiveness of this technique on data from a variety of medical fields and discuss the its ability to control the quality and size of the output meshes. With little additional effort, the same algorithm can be used to generate tetrahedral meshes of the segmentation space.

*Research supported by NSF, USA (CCF-0430735 and CCF-0635008).

†Department of Computer Science and Engineering, The Ohio State University, Columbus, Ohio, USA. Email: {tamaldey|janoos|levinej}@cse.ohio-state.edu

1 Introduction

A commonly used source of information for the medical fields is obtained in the form of 3D images scanned from real-life entities. For example, microscopy, PET-CT, and MRI all produce volumetric images scanned from anatomical objects; primarily for the purposes of aiding in the diagnosis and treatment by a skilled professional. These devices capture data in the form of 3D grids where at each point in the grid a scalar value is stored. To further understand the object, a classification algorithm is then applied to label each grid point using the scalar data values and hints of its known structure. These labels subdivide the volume into regions representing structural elements of the scanned object such as different materials (e.g. bone, muscle, organs, etc.) or different components (e.g. spine, jaw bones, uncus, etc.).

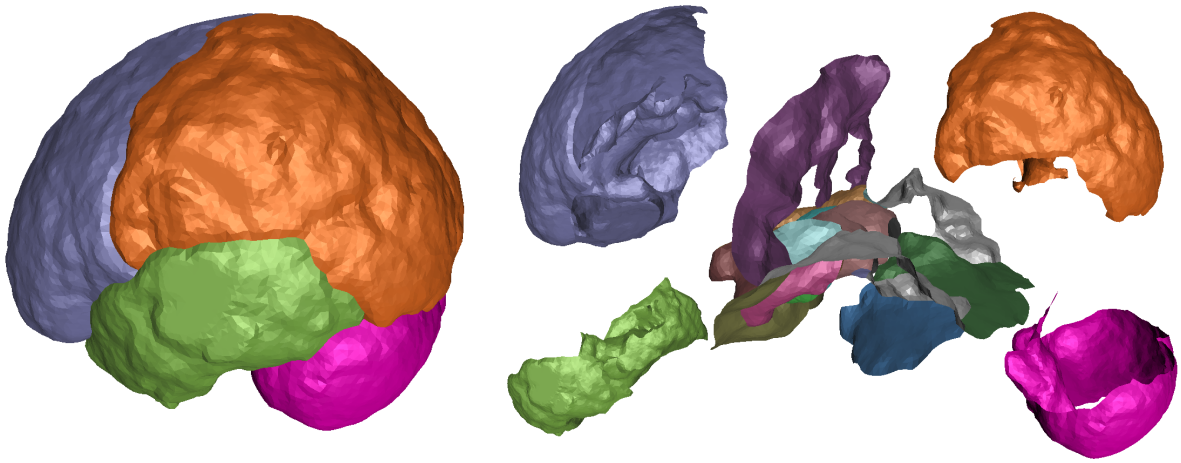

Figure 1: MNI structural atlas of the brain, with seven anatomical regions shown. We generate consistent meshes for the surfaces where these regions interface. The exploded view on the right shows the interior surfaces.

Given a labeled volume dataset, there is a need for the reconstruction and visualization of the surfaces where different labeled regions meet. We call such surfaces *interface surfaces*, as they lie precisely where two or more labeled regions intersect. More formally, interface surfaces are a subclass of shapes known as *piecewise-smooth complexes* or PSCs (see Section 2.2). The class of shapes that are PSCs is broad: any collection of smooth surface patches meeting at curves non-smoothly or in non-manifold configurations qualifies.

Interface surfaces are often a key component in the structure of the data. For example, in images obtained for medical applications it is essential to extract and visualize the surfaces of the different anatomical features for diagnosis, surgery simulation and planning, and therapy evaluation [17].

Standard labeling approaches use "hard" segmentations that produce a binary mask for each label. However, there is usually a substantial overlap between these segments, resulting in multiple labels at voxels in the overlap region. Alternatively, in "soft" segmentation methods a real-valued probability for anatomical feature is associated at each voxel of the image, resulting in a fuzzy boundary between the segments [38]. For a more in-depth review of medical image segmentation, the reader is referred to [20].

Similarly, multi-label data is generated in domains such as finite element simulations and multi-fluid hydrodynamic calculations [33]. Storing per-voxel material labels eases the simulation of the fluid flows; however, the reconstruction of material interface surfaces is essential for computing the advection of the fluids [30]. In [15], the authors present an example of extracting the crystal grain boundaries of poly-

crystalline materials, where they use a Potts model to identify the exact location of the boundary. Another application is representation of the configuration space in motion planning problems by a regular grid, in which the obstacles are partitioned into convex neighborhoods. Here, the boundary surface represents the region that can be efficiently searched for reachability [26].

In general, the identification of the interface surface for multi-label data is not an easy task, confounded by overlaps of the components, fuzzy boundaries, inherent non-manifold configurations, and arbitrarily sharp angles at the intersections of surface patches. Thus there is need for robustly generating a mesh for interface surfaces which preserves their non-manifold topology and captures their geometric features.

1.1 Contributions

Our primary contribution is the application of a recent Delaunay meshing algorithm for piecewise-smooth complexes [14] to automatically generate meshes which approximate the interface surfaces defined by multi-label datasets. This algorithm is an ideal choice for meshing this domain. Its theoretical pedigree [8] guarantees the ability to generate meshes that are topologically equivalent to the non-manifold topology that occurs frequently in interface surfaces. Its novel protection scheme allows it to generate meshes for interface surfaces that meet at any angle.

A second major stength of the algorithm is its simplicity. It only requires two numeric primitives that can be implemented robustly. The first computes intersection points between arbitary line segments and the interface surface. Second, the algorithm needs a routine for computing the curve network where three or more labeled regions meet. As a result, high quality meshes can be generated with less computational overhead.

Moreover, the algorithm can be applied to datasets from a broad range of data sources. Our experiments demonstrate the algorithm's effectiveness at producing meshes of interface surfaces for datasets from a variety domains and modalities. These meshes may be used for surface visualization and computation, and the same algorithm can produce volume meshes which are suitable inputs for finite element simulations. The quality of output meshes can be controlled by various user parameters, allowing the generation of both low density meshes suitable for rapid prototyping needs or high density meshes when accuracy is a crucial factor.

2 Background

2.1 Interface Surface Generation

The generation of interface surfaces for multi-label data has been studied in many communities, including visualization, medical imaging, and computational fluid dynamics.

For dealing with multi-labeled data on regular rectilinear grids, the concept of marching cubes [23, 41] has been extended to automatically create and apply lookup tables for interface surfaces between two or more materials [3, 4, 40, 42]. These methods can suffer from problems of ambiguities in the interface surface, topological defects, inconsistent meshes (where the boundaries of the triangle meshes do not consistently fit together), and an algebraic increase in the number of cases with respect to the number of materials. Solutions to these problems have been variously proposed, including repairing ambiguous voxels before extracting the meshes [5], domain sub-division [32], assigning probabilities at the vertices and interpolating them to extract topologically correct surface elements [18]. Algorithms based on marching tetrahedra [27] are also commonly used [15, 26]. Dual contouring methods that preserve sharp edges [19] have also been proposed for tetrahedral and hexahedral meshes [44].

In general though, grid based methods do not guarantee topological correctness, can generate poor quality triangles, and the resolution of the output mesh depends upon the grid resolution itself. The resulting

meshes often require additional post-processing to make them suitable for later applications.

Alternate approaches for extracting interface surfaces use methods from computational solid geometry [5, 22]. Level-set methods [21, 35] or particle-based methods [10, 25] are also commonly used for determining interface surfaces and meshing them. While these methods produce high quality and topologically accurate meshes, they can also be computationally expensive.

In computational fluid dynamics, the extraction of the interface between multiple materials from volumetric simulations has been extensively studied. The simple line interface calculation (SLIC) algorithm [28] partitions cells with axis-aligned planes, such that the total material volume in each cell is correct but results in discontinuous interfaces and is only first-order accurate. Alternative methods including the piecewise linear interface calculation (PLIC) algorithm [43] and its extensions [30, 34] provide second order accuracy for these data. Other approaches involve iso-surfacing over the dual grid [7], grid sub-division [2], and extensions to generalize polyhedral meshes [1]. A complete review of this literature of out the scope of the current work. The main difference of these methods is that they make assumptions of continuity, low curvature, and smoothness that are acceptable for fluid-flow data but not necessarily for the kinds of problems addressed here.

2.2 Meshing PSCs with Delaunay Refinement

To handle the complex topology common to interface surfaces, we model the interface surface as a piecewise-smooth complex (PSC). A PSC is a collection of k-dimensional *faces* called vertices (0-faces), curves (1-faces), and patches (2-faces) (see Figure 2). Each k-face in the PSC is a subset of the volume where $4-k$ labeled regions intersect.

For a PSC $\mathcal{D}$, we use $\mathcal{D}_i$ to denote the set of i-faces in a PSC and $\mathcal{D}_{\leq i}$ to represent the union $\mathcal{D}_0 \cup \mathcal{D}_1 \cup \ldots \cup \mathcal{D}_i$. If $\mathcal{D}$ is a PSC, it satisfies the usual conditions for being a complex: if any two $(i+1)$-faces $\sigma, \sigma' \in \mathcal{D}_{i+1}$ intersect, their intersection is a union of (lower dimensional) faces in $\mathcal{D}_{\leq i}$ on the boundary of both σ and σ'.

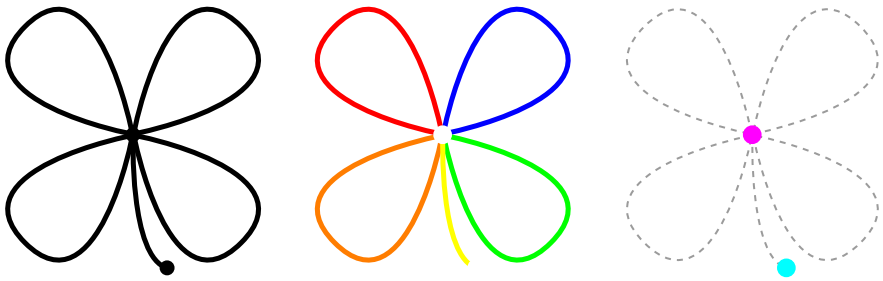

Figure 2: A clover-shaped 1-dimensional PSC. There are five elements in $\mathcal{D}_1$ (middle), highlighted in red, orange, yellow, green, and blue and two elements in $\mathcal{D}_0$ (right), highlighted in cyan and magenta. Note there is no limitation on the number of elements which share adjacent boundaries.

The well known Delaunay triangulation has become a de facto standard for representing various domains. One regularly used strategy for constructing a Delaunay triangulation uses an iterative refinement approach. *Delaunay refinement* relies on the idea of Chew's furthest point strategy [9] to incrementally build a Delaunay mesh for an input domain. Points from the domain in question are repeatedly inserted until the mesh satisfies a set of conditions. This set of conditions is chosen to trigger insertions that ensure desired properties in the output mesh. The burden is then to show that the types of points inserted eventually satisfy all conditions—implying the algorithm terminates. A packing argument is often applied by showing that

these conditions are be satisfied when the point set reaches some density. A scheme to choose points that are "furthest" (locally or globally) from the current point set completes the termination proof.

The applicability of Delaunay-based approaches for interface surfaces is slowly gaining momentum. An early approach extends the labeling to the tetrahedra within Delaunay triangulation to construct the labeled regions [31]. Particle sampling can be combined with a Delaunay reconstruction to generate interface surfaces as well [25].

However, a recent algorithm proposed a unique approach for meshing PSCs [8] based on Delaunay refinement. One novelty of this technique is that all non-smooth and non-manifold features (the faces in $\mathcal{D}_{\leq 1}$) are protected throughout the meshing algorithm by sampling them with balls. These protecting balls are turned into weighted points and Delaunay refinement is performed using a weighted Delaunay triangulation [16]. This algorithm was the first certified algorithm that could in theory successfully handle PSC inputs; however, since it employs some expensive numeric computations, a more practical version of the algorithm was necessary [14]. This version computes protecting balls in an "on-the-fly" manner and reduces the refinement triggers to a single "topological disk" test that is combinatorial in nature.

3 Algorithm

Our experiments involve extending the algorithm in [14] to mesh the interface surfaces of multi-labeled datasets using Delaunay refinement. We first give a brief review of the algorithm and discuss the components implemented specifically for this type of data.

3.1 Input and Output

The input to our algorithm is a set of N smooth *indicator functions* [29], $f_i : \mathbb{R}^3 \to \mathbb{R}, 1 \leq i \leq N$. Each f_i corresponds to the presence of label i within our input dataset. For practical purposes, our datasets come in the form of a 3D grid sampled over a space $V \subset \mathbb{R}^3$ with a list of N scalar values at each grid point which corresponding the values of each of the indicator functions at that point. We use trilinear interpolation to compute the values of the indicator functions at any point within the grid.

For any $x \in V$, the indicator functions dictate the labeling at x. Since the values are scalar, if $f_i(x) \geq f_j(x), \forall j \neq i$, then x is given the label i. The set $R_i = \{x \in V \mid f_i(x) \geq f_j(x)\ \forall j \neq i\}$ is the region in space given the label i. Note that this definition implies that a point x may have multiple labels, or alternatively, the sets R_i are not disjoint. The set $R_i \cap R_j$ where $i \neq j$ is the interface (surface) between labels i and j, which is either empty or a set of smooth two-dimensional patches for every i, j pair. Similarly, the set $R_i \cap R_j \cap R_k$, where $i \neq j \neq k$, is the interface (curve) between labels i, j, and k, which is either empty or a set of smooth one-dimensional curves for every i, j, k triple.

We can define a piecewise-smooth complex $\mathcal{D}$ using the sets R_i. The set of 2-faces, $\mathcal{D}_2$, is defined by the union of all pairwise interfaces between the labeled regions, that is $\mathcal{D}_2 = \bigcup_{i \neq j} R_i \cap R_j$. Likewise, the set of 1-faces is defined by every triple of labeled regions, $\mathcal{D}_1 = \bigcup_{i \neq j \neq k} R_i \cap R_j \cap R_k$. Similarly, $\mathcal{D}_0$ is the set of intersections of four labeled regions. Together, these three sets completely define the PSC $\mathcal{D}$. The Delaunay refinement algorithm we use outputs a mesh that approximates this PSC.

3.1.1 Obtaining Indicator Functions

To generate indicator functions we took several segmented datasets and converted them to scalar fields. The output of typical hard segmentation algorithms is a set of binary masks over the space V. Each binary mask corresponds to one f_i, but using the binary values directly is not feasible as an input form. We apply imaging filters (see Section 4.1) that convert the representation of this data to a scalar field over V. Soft segmentation algorithms output a probability that a grid point is of each label i, and hence those can be used

more readily to represent indicator functions. Our strategy for generating interface surfaces leaves the input form as flexible as possible to allow a domain expert control over labeling.

3.2 Structures

In this section we review some of the basic geometric structures which our algorithm uses. Refer to [13, 16] for additional details.

Let S be a point set sampled from the underlying space of $\mathcal{D}$. We associate each point $p \in S$ with a real-valued weight $r \geq 0$; alternatively, the weighted point p can be represented as a ball centered at p of radius r. The weight r skews the distance metric locally with respect to p. The *squared weighted distance* of any point $x \in \mathbb{R}^3$ from p is given by $\mathrm{d}(x,p) = \|x-p\|^2 - r^2$. Under this new distance metric, we define the Voronoi diagram, $\mathrm{Vor}\, S$, and Delaunay triangulation, $\mathrm{Del}\, S$ for weighted point sets. These structures follow a similar rule as their unweighted counterparts with distance between points computed using weighted distance [16].

To generate a Delaunay mesh for $\mathcal{D}$ we take special subcomplexes of $\mathrm{Vor}\, S$ and $\mathrm{Del}\, S$. In particular, we are interested in elements of $\mathrm{Vor}\, S$ and $\mathrm{Del}\, S$ that provide an approximation of $\mathcal{D}$. For any space $\mathbb{X}$, the *restricted Voronoi diagram* is the complex formed by the collection of intersections between Voronoi elements and $\mathbb{X}$. Its Delaunay counterpart, the *restricted Delaunay triangulation* of S with respect to $\mathbb{X}$, is denoted $\mathrm{Del}\, S|_{\mathbb{X}}$. A simplex $\tau \in \mathrm{Del}\, S$ is in $\mathrm{Del}\, S|_{\mathbb{X}}$ if its dual Voronoi face V_τ has a nonempty intersection with $\mathbb{X}$. For any $\sigma \in \mathcal{D}$, $\mathrm{Del}\, S|_\sigma$ denotes the Delaunay subcomplex restricted to σ. Figure 3 illustrates an example restricted Delaunay triangulation in two-dimensions.

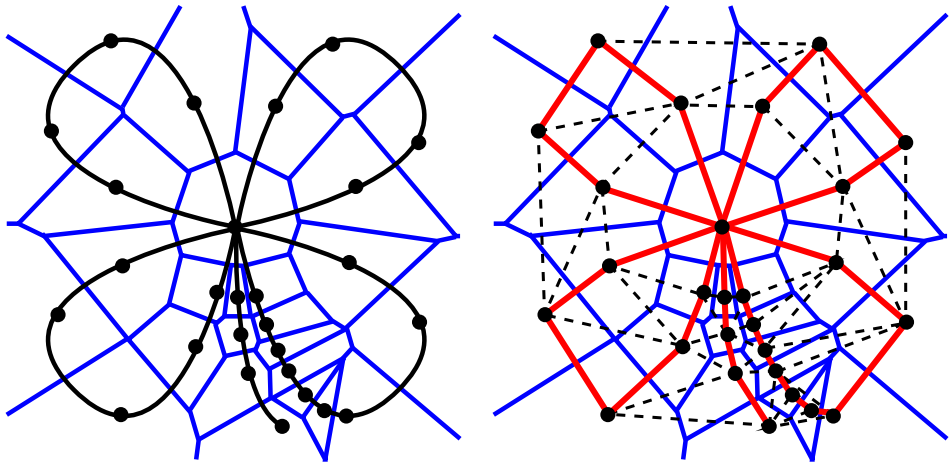

Figure 3: Restricted Delaunay triangulations. Left: a sample of our clover PSC with its corresponding Voronoi diagram (in blue). Right: Within the dual Delaunay triangulation (dashed and red segments), only a subset of its edges are restricted to the clover (red segments). When a sample is chosen carefully, restricted Delaunay triangulations can approximate this shape.

During Delaunay refinement, restricted Delaunay triangulations act as a mesh of a working approximation of the input. Since in three dimensions $\mathrm{Del}\, S$ contains simplices spanning the convex hull of S, $\mathrm{Del}\, S|_{\mathcal{D}}$ acts as a filter to compute the subset that approximates the input $\mathcal{D}$. As a subset of the Delaunay triangulation, elements in the restricted Delaunay triangulation inherit the Delaunay property as well. This fact is important for meshes with shape quality concerns, a byproduct of satisfying Delaunay constraints is that elements have a tendency toward improved shape over elements used in arbitrary triangulations over the same point set.

3.3 Refinement for Interface Surfaces

The refinement algorithm in [14] proceeds in two distinct phases. The first phase protects input features, the set $\mathcal{D}_{\leq 1}$, by covering them with a set of balls (see left two images in Figure 4). These balls act as a protected zone—it is guaranteed that no point will be inserted within the balls during refinement. Computing these balls could require a number of expensive computations involving feature size, so the algorithm from [14] begins by using a set which may have radii that are too large. The progression of the algorithm itself controls the need to shrink balls that are too large.

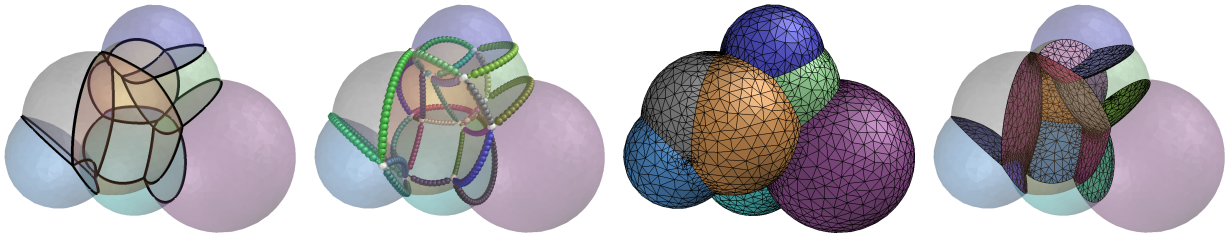

Figure 4: Refinement algorithm on the spheres dataset. Left: a set of curves are extracted where three or more labeled regions intersect. Middle left: these curves are protected with an initial set of balls that may later get refined by the algorithm. Middle right: these balls are converted to weighted points and refinement fills in samples for the interface surfaces. Right: The output mesh captures all interfaces.

The balls are then turned into weighted points and inserted into a Delaunay triangulation. The restricted Delaunay triangulation is computed, and each vertex in it is checked if it satisfies a *disk condition*. This condition requires that vertices on elements of $\mathcal{D}_2$ have a neighborhood which is a topological disk, while vertices lying on elements of $\mathcal{D}_{\leq 1}$ are allowed to satisfy the condition with a set of half disks. If any vertex fails the disk condition, a new vertex is inserted which lies on the intersection between $\mathcal{D}$ and a Voronoi edge dual to some a restricted triangle. Finally, once all vertices satisfy the disk condition, the restricted Delaunay triangulation is refined further to satisfy size and quality requirements.

The computation of intersection points between Voronoi edges and $\mathcal{D}$ requires a primitive that computes a point of intersection between an arbitrary line segment and the interface surface defined by the indicator functions. We use a binary-decision walking oracle, similar to the one discussed in [6]. This oracle uses binary search to find all points on the segment where the labeling transitions. It proceeds as follows. First, the labels at each endpoint of the line segment, l_a and l_b, are computed as well as the label at the midpoint, l_m. If $l_a = l_b$, but l_m is different, then each half of the segment is recursively checked. If $l_a \neq l_b$ each the half between a and b is checked provided that the label at that half's endpoint differs from l_m. Once the distance between a and b becomes smaller than a user specified tolerance, an intersection point (the midpoint) is returned instead of recursively checking the halves. This algorithm is simple to implement and our experiments indicate more robust than solving the trilinear system defined by the indicator functions.

3.4 Extracting Curves

The final ingredient required to implement the algorithm is a method to extract the curve network $\mathcal{D}_{\leq 1}$ from the input dataset. An example network for the spheres dataset is shown in left image of Figure 4. We use an algorithm that walks through each cube cell in the grid that contains three or more labels. Our goal is to generate a set of segments whose union is the set $\mathcal{D}_{\leq 1}$.

We first use bilinear interpolation on each face of the cell to compute the intersection points with a curve (defined by three indicator functions) and the face. If a cell has only two such points, we connect them with a segment, forming a piece of some element of $\mathcal{D}_1$. Otherwise, if a cell has more than two (indicating that

more than one curve passes through the cell), we subdivide the cell at its midpoint into eight pieces. Each of the 8 subcells are checked again, and we allow them to be subdivided up to three levels deep. If there are still more than two points at the finest level of subdivision, we connect all intersection points to their centroid, forming an element of $\mathcal{D}_0$.

This process again saves us from solving the roots of a cubic equation, instead favoring more robust subdivision. One caveat is that occasionally multiple elements of $\mathcal{D}_0$ may be computed in close proximity to each other. Our curve extraction algorithm performs a final merge step which combines elements of $\mathcal{D}_0$ that are nearby within a user defined tolerance. After this merging process, we group segments to form each individual curve in $\mathcal{D}_1$ by growing curves from each element of $\mathcal{D}_0$.

4 Implementation Details

4.1 Data Preprocessing

Each of our datasets requires some preprocessing to convert them to indicator functions suitable for meshing. Some datasets require first separating the data into individual binary masks for each label. At this point, binary morphology operators such as dilation or median filtering [36] can be applied to clean the data. All datasets from hard segmentations then are converted from binary maskes to a signed scalar field. For this task we apply a signed Danielsson distance filter [11] to each mask.

After producing the raw indicator functions, additional preprocessing steps were applied to control the size of features in the data. We use a mix of Gaussian smoothing [12] as well as curvature anisotropic diffusion (CAD) smoothing [39] at this step. CAD is a variation of classical non-linear diffusion, where the speed of the diffusion is proportional to the local curvature of the level-sets of the image. Areas of high curvature diffuse faster than areas of low curvature, thereby smoothing out noise while preserving large scale image features and edges. While this method does not exhibit the edge enhancing properties of classic anisotropic diffusion, it is more stable and preserves finer structures in the images.

4.2 Input Parameters

Our implementation accepts a parameter to scale the radii of initial protecting balls as well as parameters to control triangle shape. Our parameters include a maximum bound for triangle circumradii (thus uniformly sizing all triangles) and shape quality is controlled with a bound for the circumradius to shortest edge ratio. Mesh conformity to the interface surface is controlled by allowing the user to specify a maximum distance from the circumcenter of the triangle to target shape. Triangles are refined to satisfy shape criteria provided their insertions do not get too close to a protecting ball.

For volume meshing, a similar set of parameters control tetrahedra shape. We allow the user to specify a uniform size control via a maximum bound on tetrahedra circumradii. Tetrahedra aspect ratios are controlled similar to triangles by specifying tetrahedra aspect ratio in terms of a circumradius to shortest edge threshold. This quality criterion generates tetrahedra suitable for finite element methods, but can still allow some sliver tetrahedra in the output mesh. These slivers can be eliminated as a postprocessing step or using the circumradius to inscribing radius ratio for refinement.

5 Experimental Results

Our experimental datasets cover a broad range of inputs from various sources. We used synthetic data and segmented datasets from both CT and MRI. Our datasets include segmentations done manually by experts as well as by using various both hard and soft classification algorithms. Table 1 gives an overview of the datasets, their size, number of labels, and modalities.

Table 1: Dataset Summary. We list the dimensions, source, and number of labels for each data used.

Dataset	# of Labels	Source	Dimensions
Spheres	8	synthetic	$128 \times 128 \times 128$
Orange	4	MRI	$256 \times 256 \times 64$
Temporal Bone	5	CT	$256 \times 256 \times 256$
Tomato	4	MRI	$256 \times 256 \times 64$
Brain Atlas	7	MRI	$182 \times 218 \times 182$
Torso	4	MRI	$260 \times 121 \times 169$
Bulldog Skull	4	CT	$196 \times 176 \times 268$
Soft Brain	2	MRI	$181 \times 217 \times 181$

*Note, there is an additional "background" label present in all datasets.

Our experimental PC setup consisted of an Intel Pentium4 CPU at 2.8Ghz with 2GB of memory. We used the same machine for both preprocessing and mesh generation. Preprocessing required no more than a few minutes for each label, once the appropriate filters were selected. Mesh generation times are reported in Table 2.

Table 2: Meshing summary. We list the number of output vertices and the number of curves protected. Output times (in minute:seconds) are broken down by protection time as well and time to generate the mesh. File I/O times are excluded.

Dataset	# of Curves	# of Vertices	Protection Time	Meshing Time
Spheres	36	4492	0:16	0:50
Orange	24	24543	0:14	8:46
Temporal Bone	16	24898	0:11	8:32
Tomato	35	25701	1:01	9:07
Brain Atlas	64	35890	1:04	17:50
Torso	57	39826	1:37	19:27
Bulldog Skull	17	51474	0:10	32:48
Soft Brain	100	211664	1:57	244:01

These times are quite competitive for meshing interface surfaces in a high quality manner. In particular, the major bottleneck of the algorithm is computing the weighted Delaunay triangulation with dynamic insertions. If we specify a dense output mesh (as we did for the soft segmentation of the brain), we see the computation time grow nonlinearly. However, since our meshing density does not rely on the grid resolution, we can save ourselves computation by specifying more relaxed refinement criteria. At lower resolutions we can still capture the shapes quite well. Thus, one strength of our algorithm is we can compute meshes which capture the topology using significantly fewer vertices.

A visualization of the mesh we generated for the MNI brain atlas is shown in figure 1. This brain image was obtained from the MNI-space atlases provided along with FSL [37]. A single subject's structural image was hand segmented, and the labels were then propagated to more than 50 subjects' structural images using nonlinear registration. Each resulting labeled brain was then transformed into MNI152 space using affine registration, before averaging segmentations across subjects to produce the final probability images.

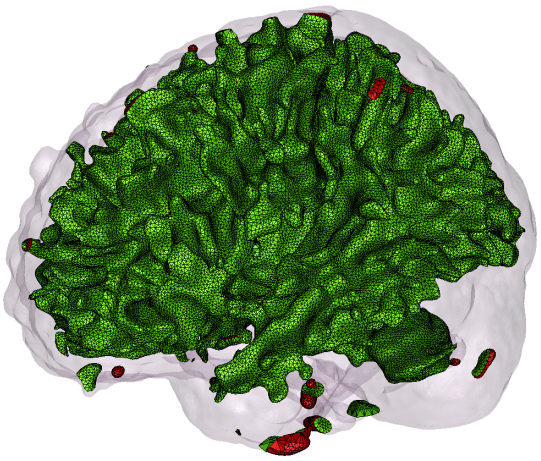

Figure 5: Soft segmentation of the brain. Grey matter is draw transparently and white matter is drawn in green. Red triangles show where the white matter protrudes through the grey.

Figure 5 shows a mesh from a dataset generated by probabilistic segmentation of the brain's grey and white matter. This dataset was obtained by first normalizing a T1-weighted MR image to the MNI152 reference space, followed by a probabilistic segmentation [38] using a combination of in-house and standard tools such as FSL [37]. The prior probabilities for the tissue distributions were obtained from the MNI-ICBM atlas (NIH P-20 project) [24], specified in the same reference space. The resulting probability images contain values in the range of zero to one, representing the posterior probability of a voxel being either grey matter or white matter. It is interesting to note that our mesh reveals that in this segmentation the white matter pierces through the outer grey matter in several locations.

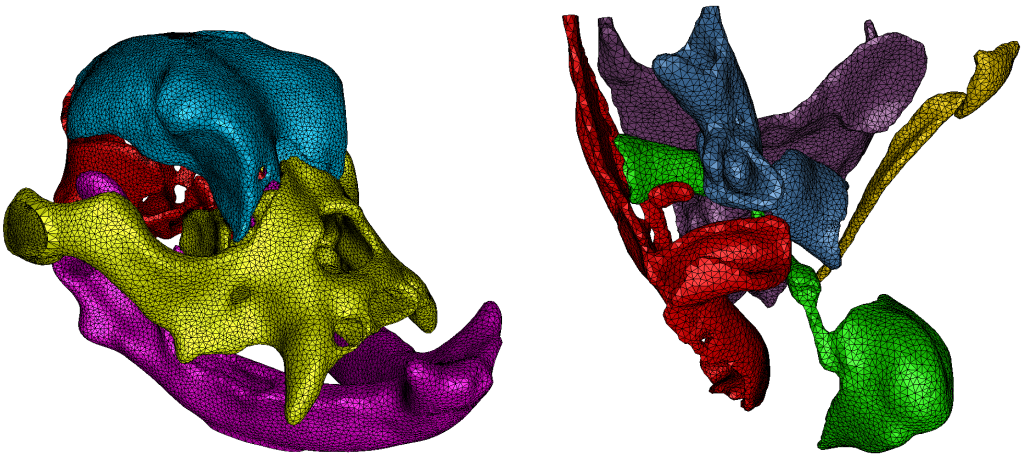

Figure 6: CT datasets. We grouped labels of the english bulldog (top) and temporal bone (bottom) datasets to form 4 and 5 labels, respectively.

Two datasets generated from CT sources are shown in figure 6. Both datasets were hand-segmented by experts into more than 50 structural elements. We group them in a smaller number of labels to make them more manageable for our software to mesh. As a result, both of these datasets had far fewer curve elements

Table 3: Quality and shape statistics. We list the average and maximum aspect ratio (radius/shortest edge length) for each of the datasets. We also computed the minimum dihedral angle (in degrees) between all pairs of faces circling each non-manifold edge. This measure shows how sharply the interface surfaces are meeting at curve elements.

Dataset	# of triangles	mean aspect ratio	max aspect ratio	min angle
Spheres	9527	1.00	1.05	42.35
Orange	49017	1.00	1.03	20.71
Temporal Bone	49911	1.00	1.03	3.74
Tomato	52729	1.00	1.03	18.39
Brain Atlas	72962	1.00	1.03	13.30
Torso	82104	1.00	1.05	14.69
Bulldog Skull	103036	1.00	1.02	12.10
Soft Brain	424690	1.00	1.02	8.72

than the MRI datasets.

Table 3 shows a summary of quality statistics for the triangles produced by the meshing algorithm. We control the aspect ratios of triangles by refining to enforce the condition that all triangles have aspect ratio below a minimum threshold. This table also shows the minimum dihedral angle between any pair of triangles circling an edge along a curve element. This value indicates how sharply the interface surfaces are meeting in the output mesh. Our implementation can tolerate surfaces meeting at arbitrary angles, as the table indicates.

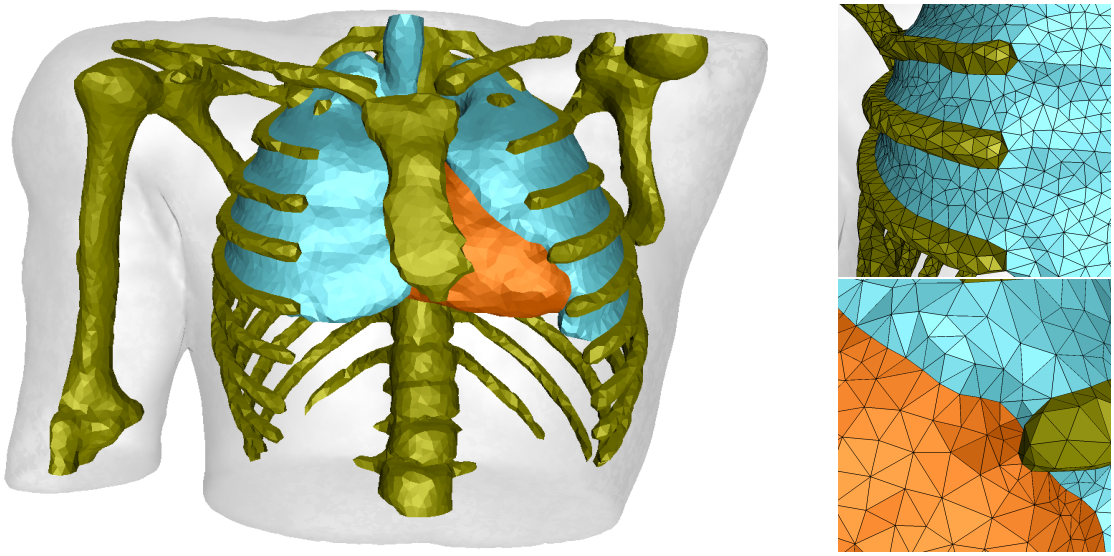

Figure 7: Torso dataset with close ups of protected curves. Labels are four the skin (transparent), skeleton (yellow), lungs (cyan), and heart (orange).

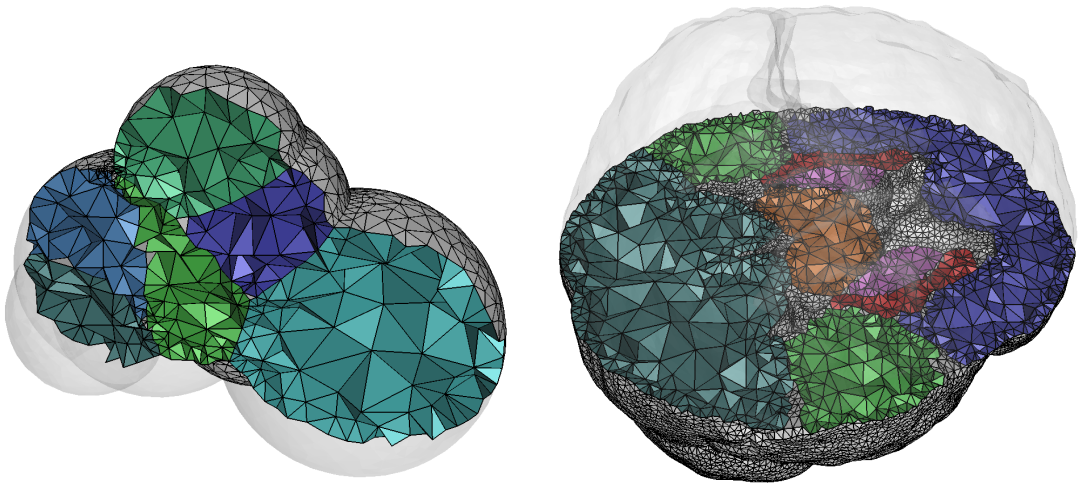

Figure 8: Volume meshes for the sphere and brain atlas. Our algorithm generates Delaunay tetrahedral meshes which can be used for finite element simulations.

Three additional datasets come from MRI sources. Figure 7 shows a torso dataset with four of the major structures labeled. This dataset has a large network of curves to protect where the skin, lungs, and skeleton meet, shown in the closeup shots. Figure 9 shows both the orange and tomato datasets, where we peel away the outer layers to see the internal structures. These two datasets labels are generally contained within each other, thus they have few curves to protect.

After satisfying the constraints for surface refinement, we can generate volume meshes as well. Figure 8 shows some example volume meshes for the spheres and brain atlas datasets. Since the 3D Delaunay triangulation already generates a tetrahedralization of the volume contained within the shape, as soon as the interface surface is captured we automatically have a volume mesh. We can refine these tetrahedra by using the standard approach of inserting the circumcenters of tetrahedra with bad radius-edge ratios. We perform this insertion for any tetrahedra provided that the circumcenter does not cause the deletion of one of the triangles on the interface surface. This strategy lets us refine within the volume, but ensures the interface surface triangles remain in the output mesh.

6 Conclusion and Future Work

By using a Delaunay refinement approach, the challenge of generating quality meshes for interface surfaces is significantly diminished. By applying an algorithm that extends a proven theoretical result, we can mesh interface surfaces with surface elements that meet at arbitrarily sharp angles. Our experiments show that this technique lends itself to generating both low resolution meshes fast and higher quality approximations in reasonable time. While our times for meshing are competitive, its performance bottleneck is the asymptotic speed to compute the weighted Delaunay triangulation. A faster implementation of this data structure could greatly improve the run time.

Since we do not explicitly compute a measure of feature size for the surfaces, our implementation favors using a uniform size requirement to control the mesh density. It is often desirable to have meshes which adapt to the input's features. However, for the case of interface surfaces, a true estimation of this measure may not be possible to compute without a prohibitive amount of overhead. Future research is necessary to explore how should an adaptive mesh be generated for the case of piecewise smooth shapes.

Our implementation allows for labeled inputs generated from various imaging and segmentation sources.

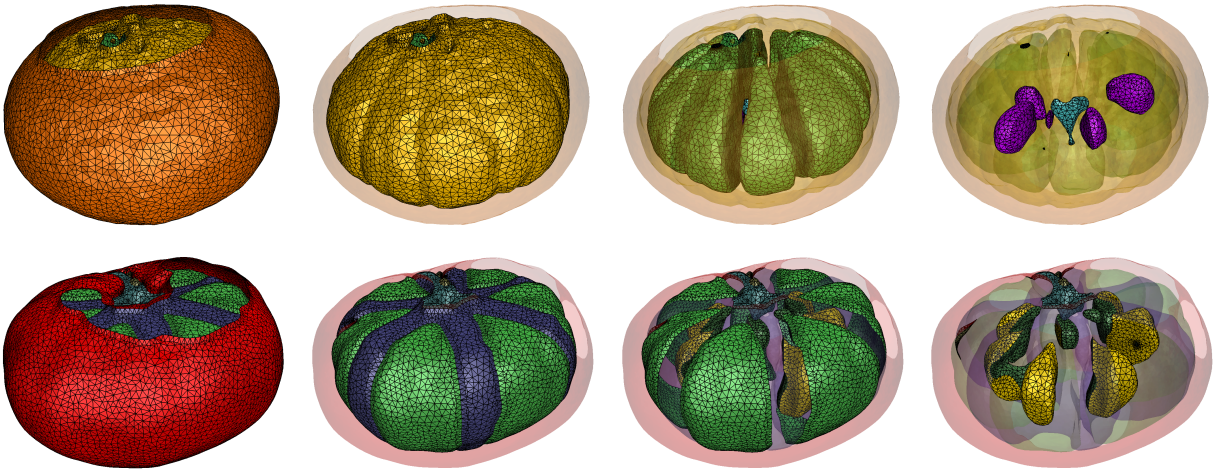

Figure 9: Multi-label fruit datasets. The top row shows the orange dataset with outer skin, membrane, pulp, and seeds. The bottom row has the tomato with labels endocarp, locule, placenta, and core.

We use only basic filters on these datasets to minimize the amount of preprocessing time. Separating this processing step from the algorithm itself allows a domain expert control over how much the data is altered when converted to a system of indicator functions. Since our algorithm uses the indicator functions as input, it is interesting to note that we can process probabilistic segmentations in a more direct manner. Common usage thresholds and converts these soft segmentations to binary masks before processing—creating a potential loss of information.

Acknowledgements

The authors wish to thank Thomas Kerwin, Miriah Meyer, and Ross Whitaker for aiding in obtaining datasets. The temporal bone and dog skull dataset are provided courtesy of Don Stredney at the Ohio Supercomputer Center. The tomato and orange datasets were produced at Lawrence Berkeley Laboratory by Bill Johnston and Wing Nip of the Information and Computing Sciences Division. The torso dataset is courtesy of John Triedman and Matthew Jolley at Boston Children's Hospital.

References

[1] H. T. Ahn and M. Shashkov. Multi-material interface reconstruction on generalized polyhedral meshes. *Journal of Computational Physics*, 226(2):2096–2132, 2007.

[2] J. C. Anderson, C. Garth, M. A. Duchaineau, and K. I. Joy. Discrete multi-material interface reconstruction for volume fraction data. *Computer Graphics Forum*, 27(3):1015–1022, 2008.

[3] D. C. Banks and S. A. Linton. Counting cases in marching cubes: Toward a generic algorithm for producing substitopes. In *IEEE Visualization*, pages 51–58, 2003.

[4] M. Bertram, G. Reis, R. H. van Lengen, S. Köhn, and H. Hagen. Non-manifold mesh extraction from time-varying segmented volumes used for modeling a human heart. In *EuroVis05: Joint Eurographics - IEEE VGTC Symposium on Visualization*, pages 199–206, 2005.

[5] J. Bloomenthal and K. Ferguson. Polygonization of non-manifold implicit surfaces. In *SIGGRAPH*, pages 309–316, 1995.

[6] J.-D. Boissonnat and S. Oudot. Provably good sampling and meshing of surfaces. *Graphical Models*, 67(5):405–451, 2005.

[7] K. S. Bonnell, M. A. Duchaineau, D. Schikore, B. Hamann, and K. I. Joy. Material interface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 9(4):500–511, 2003.

[8] S.-W. Cheng, T. K. Dey, and E. A. Ramos. Delaunay refinement for piecewise smooth complexes. In *SODA*, pages 1096–1105, 2007.

[9] L. P. Chew. Guaranteed-quality mesh generation for curved surfaces. In *Symposium on Computational Geometry*, pages 274–280, 1993.

[10] P. Crossno and E. Angel. Isosurface extraction using particle systems. In *IEEE Visualization*, pages 495–498, 1997.

[11] P.-E. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14:227–248, 1980.

[12] R. Deriche. Fast algorithms for low-level vision. *IEEE Transactions on Pattern Analysis and Machince Intelligence*, 12(1):78–87, 1990.

[13] T. K. Dey. *Curve and surface reconstruction: algorithsm with mathematical analysis*. Cambridge University Press, New York, 2007.

[14] T. K. Dey and J. A. Levine. Delaunay meshing of piecewise smooth complexes without expensive predicates. Technical Report OSU-CISRC-7108-TR40, Department of CSE, The Ohio State University, July 2008.

[15] S. E. Dillard, J. Bingert, D. Thoma, and B. Hamann. Construction of simplified boundary surfaces from serial-sectioned metal micrographs. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1528–1535, 2007.

[16] H. Edelsbrunner. *Geometry and Topology for Mesh Generation*. Cambridge University Press, England, 2001.

[17] G. Gerig, W. Kuoni, R. Kikinis, and O. Kübler. Medical imaging and computer vision: An integrated approach for diagnosis and planning. In *Mustererkennung 1989, 11. DAGM-Symposium, Hamburg, 2.-4. Oktober 1989, Proceedings*, pages 425–432, 1989.

[18] H.-C. Hege, M. Seebaß, D. Stalling, and M. Zöckler. A generalized marching cubes algorithm based on non-binary classifications. Technical Report SC-97-05, Zuse-Institute Berline (ZIB), 1997.

[19] T. Ju, F. Losasso, S. Schaefer, and J. D. Warren. Dual contouring of hermite data. *ACM Transactions on Graphics*, 21(3):339–346, 2002.

[20] T. Kapur. *Model based three dimensional Medical Image Segmentation*. PhD thesis, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, May 1999.

[21] T. Karkanis and A. J. Stewart. Curvature-dependent triangulation of implicit surfaces. *IEEE Computer Graphics and Applications*, 21(2):60–69, 2001.

[22] D. H. Kim, U. Döring, and B. Brüderlin. Polygonization of non-manifolds with the aid of interval operators. pages 145–151, 2000.

[23] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th ACM SIGGRAPH*, pages 163–169. ACM, 1987.

[24] J. Mazziotta, A. Toga, A. Evans, P. Fox, J. Lancaster, K. Zilles, R. Woods, T. Paus, G. Simpson, B. Pike, C. Holmes, L. Collins, P. Thompson, D. MacDonald, M. Iacoboni, T. Schormann, K. Amunts, N. Palomero-Gallagher, S. Geyer, L. Parsons, K. Narr, N. Kabani, G. L. Goualher, D. Boomsma, T. Cannon, R. Kawashima, and B. Mazoyer. A probabilistic atlas and reference system for the human brain: International consortium for brain mapping (icbm). *Philos Trans R Soc Lond B Biol Sci*, 356(1412):1293–1322, Aug 2001.

[25] M. D. Meyer, R. T. Whitaker, R. M. Kirby, C. Ledergerber, and H. Pfister. Particle-based sampling and meshing of surfaces in multimaterial volumes. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1539–1546, 2008.

[26] H. Müller. Boundary extraction for rasterized motion planning. In *Modelling and Planning for Sensor Based Intelligent Robot Systems*, pages 41–50. World Scientific, 1994.

[27] G. M. Nielson and R. Franke. Computing the separating surface for segmented data. In *IEEE Visualization*, pages 229–233, 1997.

[28] W. Noh and P. Woodward. SLIC (simple line interface calculation). *Lecture Notes in Physics*, 59:330–340, 1976.

[29] A. A. Pasko, V. Adzhiev, A. Sourin, and V. V. Savchenko. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer*, 11(8):429–446, 1995.

[30] J. E. Pilliod and E. G. Puckett. Second-order accurate volume-of-fluid algorithms for tracking material interfaces. *Journal of Computational Physics*, 199(2):465 – 502, 2004.

[31] J.-P. Pons, F. Ségonne, J.-D. Boissonnat, L. Rineau, M. Yvinec, and R. Keriven. High-quality consistent meshing of multi-label datasets. In *20th International Conference on Information Processing in Medical Imaging*, volume 4584, pages 198–210. Springer, 2007.

[32] B. Reitinger, A. Bornik, and R. Beichel. Constructing smooth non-manifold meshes of multi-labeled volumetric datasets. In *WSCG (Full Papers)*, pages 227–234, 2005.

[33] W. J. Rider and D. B. Kothe. Reconstructing volume tracking. *Journal of Computational Physics*, 141(2):112–152, 1998.

[34] S. P. Schofield, R. V. Garimella, M. M. Francois, and R. Loubère. A second-order accurate material-order-independent interface reconstruction technique for multi-material flow simulations. *Journal of Computational Physics*, 228(3):731–745, 2009.

[35] J. M. Schreiner, C. E. Scheidegger, and C. T. Silva. High-quality extraction of isosurfaces from regular and irregular grids. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1205–1212, 2006.

[36] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, Inc., Orlando, FL, USA, 1983.

[37] S. Smith, M. Jenkinson, M. Woolrich, C. Beckmann, T. Behrens, H. Johansen-Berg, P. Bannister, M. D. Luca, I. Drobnjak, D. Flitney, R. Niazy, J. Saunders, J. Vickers, Y. Zhang, N. D. Stefano, J. Brady, , and P. Matthews. Advances in functional and structural MR image analysis and implementation as FSL. *NeuroImage*, 23(S1):208–219, 2004.

[38] W. M. Wells III, W. E. L. Grimson, R. Kikinis, and F. A. Jolesz. Adaptive segmentation of mri data. *IEEE Transactions on Medical Imaging*, 15(4):429–442, Aug 1996.

[39] R. T. Whitaker and S. M. Pizer. A multi-scale approach to nonuniform diffusion. *CVGIP: Image Understanding*, 57(1):99–110, 1993.

[40] Z. Wu and J. M. Sullivan. Multiple material marching cubes algorithm. *International Journal for Numerical Methods in Engineering*, 58(2):189–207, 2003.

[41] G. Wyvill, C. McPheeters, and B. Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4):227–234, February 1986.

[42] S. Yamazaki, K. Kase, and K. Ikeuchi. Non-manifold implicit surfaces based on discontinuous implicitization and polygonization. In *Geometric Modeling and Processing*, pages 138–148. IEEE Computer Society, 2002.

[43] D. Youngs. Time-dependent multi-material flow with large fluid distortion. In *Numerical Methods for Fluid Dynamics*, pages 273–285. Academic Press, 1982.

[44] Y. Zhang, T. Hughes, and C. L. Bajaj. Automatic 3d mesh generation for a domain with multiple materials. In *Proceedings of the 16th International Meshing Roundtable*, pages 367–386. Springer, 2007.