To appear in ACM TOG 31(4).

Object-Space Multiphase Implicit Functions

Zhan Yuan
The University of Hong Kong

Yizhou Yu
The University of Hong Kong
University of Illinois at Urbana-Champaign

Wenping Wang
The University of Hong Kong

Abstract

Implicit functions have a wide range of applications in entertainment, engineering and medical imaging. A standard two-phase implicit function only represents the interior and exterior of a single object. To facilitate solid modeling of heterogeneous objects with multiple internal regions, object-space multiphase implicit functions are much desired. Multiphase implicit functions have much potential in modeling natural organisms, heterogeneous mechanical parts and anatomical atlases. In this paper, we introduce a novel class of object-space multiphase implicit functions that are capable of accurately and compactly representing objects with multiple internal regions. Our proposed multiphase implicit functions facilitate true object-space geometric modeling of heterogeneous objects with non-manifold features. We present multiple methods to create object-space multiphase implicit functions from existing data, including meshes and segmented medical images. Our algorithms are inspired by machine learning algorithms for training multicategory max-margin classifiers. Comparisons demonstrate that our method achieves an error rate one order of magnitude smaller than alternative techniques.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Boundary representations

Keywords: Implicit Surfaces, Non-Manifold Surfaces, Support Vector Machines, Linear Programming, Piecewise Polynomial Surfaces

Links: DL PDF

1 Introduction

Implicit functions have been frequently used for solid modeling. In addition, they have a wide range of other applications in entertainment, engineering and medical imaging. A standard two-phase implicit function only returns a signed scalar number, which classifies points into two regions representing the interior and exterior of a single object. To facilitate solid modeling of heterogeneous objects with multiple internal regions, object-space multiphase implicit functions are much desired.

Nevertheless, developing a true multiphase implicit representation imposes many challenges. First, a multiphase implicit function should be able to perform multiway region membership tests. Traditional two-phase implicits use a sign to indicate two-way region membership. How can we generalize this two-way test to a multiway test? Second, an object with multiple internal regions gives rise to singularities (non-manifold features), where three or more regions are simultaneously adjacent. How can a multiphase implicit function accurately and compactly represent these non-manifold features as well as their incident surfaces? Third, many operations, such as blending and deformation, not only need the region membership of a point, but also require some estimation of distance between the point and the implicit surface. How can we enable such distance estimation in a multiphase implicit function?

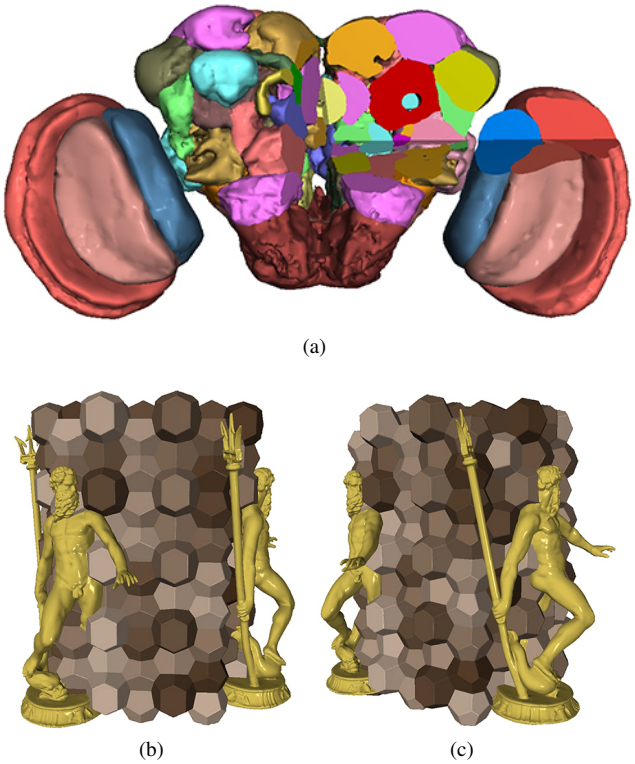


Figure 1: *Two examples of object-space multiphase implicit functions. (a) A MIF for a segmented fruit fly brain (original data courtesy of Howard Hughes Medical Institute), (b)-(c) two views of a MIF converted from multiple manifold meshes, including the* NEPTUNE *(provided courtesy of INRIA by AIM@SHAPE-VISIONAIR Shape Repository) and cells from the Weaire-Phelan structure.*

Albeit being challenging to develop, a multiphase implicit representation opens the door to many practical applications (Fig. 2). Many natural organisms and materials, such as the human body, fruits, and rocks, have complex volumetric compartments and structures. A multiphase implicit representation offers a systematic way to model the internal geometry of such natural objects and materials. Another usage is modeling heterogeneous solids made of different constituent materials, such as different types of metal. Such heterogeneous solids are ubiquitous, including all sorts of mechanical parts and man-made machinery. In biomedical research and education, an anatomical atlas typically consists of multiple spatial regions, where annotations and other region-specific attributes are recorded. Modeling an atlas with a multiphase implicit offers multiple advantages, including faster region queries and better support for topological variations.

In this paper, we introduce a novel class of object-space multiphase implicit functions that are capable of accurately and compactly representing objects with multiple internal regions. Our proposed multiphase implicit functions facilitate true object-space geometric modeling of heterogeneous objects with non-manifold fea-

Figure 2: *Examples of heterogeneous volumes with complex structures. From left to right: human head, tomato, combustion chamber engine, which is made of different constituent materials.*

tures. Since there exist multiple scalar components in a vector-valued multiphase implicit function, a multiway region membership test can be conducted through comparisons among the scalar components. Every pair of adjacent regions are separated by a surface blended from multiple polynomial pieces. Each polynomial piece corresponds to a hyperplane in a higher-dimensional space. To facilitate distance computation, we further make our multiphase implicit function represent a signed distance to the bounding hyperplanes of a region.

We present multiple methods to create object-space multiphase implicit functions from existing data. In the first method, we rely on an efficient linear programming formulation to compute a multiphase implicit function from labeled data, such as a segmented medical image. Our formulation is inspired by machine learning algorithms for training multicategory support vector machines. In the second method, we develop algorithms for creating multiphase implicit functions from existing manifold or non-manifold meshes.

We have successfully applied our algorithms and created multiphase implicit functions from multiple data sources, including artificial data, images and meshes representing biomedical contents as well as meshes representing heterogeneous mechanical parts. Comparisons demonstrate that our method achieves an error rate one order of magnitude smaller than alternative techniques.

2 Related Work

An implicit function F can be defined in various forms. It can have a closed analytical form. For example, the zero isosurface of $F(x, y, z) = x^2 + y^2 + z^2 - 1$ defines a unit sphere. Note that it is hard for a single analytical function to describe sharp features, such as corners and creases. The second choice is a regular discrete grid, and an interpolation scheme, such as trilinear and tricubic, is needed to evaluate the function value at an arbitrary point. The Marching Cubes algorithm [Lorensen and Cline 1987] can be used to extract the zero isosurface as a triangle mesh. A powerful approach defines F through the blending of a set of basic implicit shapes [Blinn 1982; Wyvill et al. 1986], called blobs or meta balls. To increase the complexity of implicit models, Ricci [1973] organizes multiple implicit surfaces into a CSG tree to construct complex solid geometries from simpler primitives. Wyvill *et al.* [1999] proposes blob trees that further incorporate blending and warping operations. To perform collision resolution, Gascuel [1993] introduced an effective method for modeling multiple contacting implicit surfaces. This method ensures there is no interpenetration between any pair of contacting surfaces. However, it does not allow multiple contact points, which give rise to non-manifold features that can be represented by our method in this paper.

An implicit function can also be defined through data-driven methods. One option is for F to interpolate a set of irregularly distributed points. There could be more point samples at regions with fine details. Frequently used interpolation schemes include various forms of basis functions [Carr et al. 2001; Turk and O'Brien 2002], such as thin-plate splines and Gaussian radial basis functions (RBFs). Alternatively, F can be defined in a piecewise manner using an adaptive space subdivision scheme. For example, if an octree is adopted to partition the space into a set of nodes, a simple closed-form implicit function, such as a second-order polynomial, can be used to describe the local shape of the surface confined within every octree node [Frisken et al. 2000; Ohtake et al. 2003]. Since the local shapes are defined independently, a blending scheme is also needed to produce a final smooth implicit surface across different nodes. All the aforementioned methods are designed for representing objects with a single interior whose boundary forms a manifold. They are unable to represent volume objects with internal non-manifold features.

Support vector machines (SVMs) have previously been applied to implicit surface modeling and deformation field representation [Schölkopf et al. 2005; Steinke et al. 2005]. However, these methods use global kernel SVMs while our method in this paper relies on adaptive space subdivision and locally linear/polynomial SVMs that are fast to train. More importantly, these methods use two-class SVMs for modeling traditional two-phase implicit surfaces while our method is inspired by multicategory SVMs and models the geometry of heterogeneous volumes with non-manifold features.

In the context of multiphase level set methods [Zhao et al. 1996; Losasso et al. 2006], researchers have been using vector-valued implicit functions to define multiphase level set functions. These vector-valued implicit functions are not defined for the purpose of geometric modeling, but for simulations solved by a multiphase level set method. Therefore, they are all defined on a regular discrete grid used by the multiphase level set method. Non-manifold implicit surfaces in [Yamazaki et al. 2002] and the multi-material volumes in [Feng et al. 2010] are also defined similarly. The most significant difference between our multiphase implicit function and these vector-valued implicit functions is that our multiphase implicit function has a compact piecewise polynomial form and is defined using an adaptively subdivided octree. Therefore, our multiphase implicit function is more suitable for object-space resolution-independent geometric modeling. Note that the method in [Zhang et al. 2010] can be used for extracting tetrahedral and hexahedral meshes from a composite domain made up of multiple materials.

Volume fraction data [Takayama et al. 2010; Anderson et al. 2010] is another representation of multi-material objects. It stores volume fractions of different materials in a grid of cells. However, the boundary of each material cannot be uniquely inferred from the given volume fractions. Thus, it cannot directly support region membership tests. In contrast, our MIF clearly defines region boundaries and can efficiently support region queries.

3 Multiphase Implicit Functions

Multiphase implicit functions, *MIF* for short, is a type of vector-valued implicit functions for describing a partition of the 3D space into nonoverlapping regions so that the region membership of any point can be conveniently obtained. A MIF has multiple scalar components, the number of which is the same as the number of regions. Each scalar component is a continuous function of 3D coordinates. The scalar components in a MIF are not mutually independent. Within every region, the scalar component corresponding to that region takes the largest values among all components [Stalling et al. 1998; Feng et al. 2010].

Let $(\Omega_1, \Omega_2, \ldots, \Omega_m)$ be a decomposition of $\mathcal{R}^3$ such that $\cup_{i=1}^{m} \Omega_i = \mathcal{R}^3$ and $\text{Vol}(\Omega_i \cap \Omega_j) = 0, i \neq j$, which means any two regions do not overlap except at their shared boundary, which has

zero volume. Let $\mathcal{F}(\mathbf{x}) = [F_1(\mathbf{x})F_2(\mathbf{x})\dots F_m(\mathbf{x})]^T, x \in \mathcal{R}^3$, be a vector-valued continuous function defined on $\mathcal{R}^3$. $\mathcal{F}$ is a MIF if the following conditions hold: i) $F_k(\mathbf{x}) = \max_i F_i(\mathbf{x})$ iff $\mathbf{x} \in \Omega_k$; ii) if $\mathbf{x}$ belongs to the interior of Ω_k, there does not exist $l(\neq k)$ such that $F_l(\mathbf{x}) = F_k(\mathbf{x})$. The second condition emphasizes that the scalar components must have a unique maximum at the interior of a region. Figure 3 visualizes a multiphase implicit function defined for four regions. Note that the coupling among scalar components give rise to high-quality consistent region boundaries.

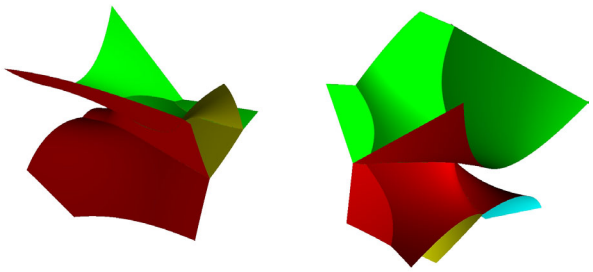


Figure 3: *Two views of a multiphase implicit function for four regions. Note that region boundaries are cleanly defined without any penetrations and gaps.*

We can further verify the following facts. Along a boundary surface shared by two adjacent regions, the two scalar components corresponding to these two regions take equal values that are strictly larger than the rest of the components. Likewise, along a curve or point shared by three or more simultaneously adjacent regions, the scalar components corresponding to these adjacent regions take equal values that are strictly larger than the rest of the components.

3.1 Polynomial Multiphase Implicits from Labeled Data

Suppose a set of labeled data points are available for a decomposition of the 3D space into multiple regions. Each data point is associated with 3D coordinates and a label indicating its region membership. In this section, we focus on computing an analytical MIF from such labeled data points, and adopt multivariate polynomials as the class of analytical implicit functions. Advantages of such implicit functions include compactness and resolution independence. Examples of labeled data include segmented and labeled 3D biomedical images as well as labeled sample points around a closed non-manifold mesh, whose compartments define multiple regions.

Note that for a two-phase implicit function, the sign of the function is more important than the magnitude of the function value when we determine whether a point belongs to the interior of an object. Likewise, for a MIF, the relative magnitude of its scalar components is more important than their standalone values. This argument essentially means that the region label associated with a data point is more important than any prescribed function values at that point. Thus our goal should be seeking an implicit function that maximizes the accuracy of predicted region labels rather than one that minimizes an implicit surface fitting error. This goal is actually the same as classifier training in machine learning. More interestingly, support vector machines (SVMs) [Vapnik 1998] are a type of classifiers that use an implicit function in a multiple dimensional space to predict the class label of any testing data point. Inspired by multicategory support vector machines [Bredensteiner and Bennett 1999; Oladunni and Singhal 2009], we formulate the optimization of a MIF according to the training of a multicategory SVM where region labels in our problem are treated as class labels used by the SVM.

Let $\{(\mathbf{x}_i, l_i)\}_{i=1}^n$ be a set of labeled training data, where $\mathbf{x}_i \in \mathcal{R}_3$ denotes the 3D position of a data point and l_i denotes its associated region label. Note that $1 \le l_i \le m$, where m is the number of regions. In the optimal MIF we seek, we first consider the case where each scalar component is a linear function. That is, $F_j(\mathbf{x}) = \mathbf{w}_j^T\mathbf{x} + b_j,\ j = 1, ..., m$. Since region boundaries have zero volume, without loss of generality, we assume all training data points fall inside the regions, not on their boundaries. Thus, the definition of a MIF imposes the following inequality constraints,

$$(\mathbf{w}_{l_i} - \mathbf{w}_j)^T\mathbf{x}_i + (b_{l_i} - b_j) > 0 \qquad (1)$$
$$i = 1, \dots, n,\ j = 1, \dots, m,\ j \neq l_i$$

Note that if we assume the training data points are linearly separable, there must exist two parallel planes between every pair of regions such that there are no points from these regions between the two planes (Figure 4(a)). The existence of such parallel planes further indicates that there exists a positive lower bound γ so that the left hand side of the inequalities in (1) are all greater than or equal to γ. By scaling $\mathbf{w}_j$'s and b_j's by $1/\gamma$, and reusing the same notations for such scaled coefficients, the following constraints must hold

$$(\mathbf{w}_{l_i} - \mathbf{w}_j)^T\mathbf{x}_i + (b_{l_i} - b_j) \ge 1 \qquad (2)$$
$$i = 1, \dots, n,\ j = 1, \dots, m,\ j \neq l_i$$

where the margin (distance) between the pair of parallel planes for regions Ω_k and Ω_j becomes $2/(\|\mathbf{w}_k - \mathbf{w}_j\|)$. However, noisy positions in the training data can give rise to violations of linear separability (Figure 4(b)). To improve resilience in the presence of outliers, slack variables are often introduced and the constraints in (2) become

$$(\mathbf{w}_{l_i} - \mathbf{w}_j)^T\mathbf{x}_i + (b_{l_i} - b_j) \ge 1 - \xi_{ij},$$
$$\xi_{ij} \ge 0, \qquad (3)$$
$$i = 1, \dots, n \quad j = 1, \dots, m \quad j \neq l_i$$

Standard SVMs need to solve a quadratic programming problem, which is expensive on large training datasets compared with linear problems. Fortunately, there exist much more efficient variants of multicategory SVMs, including least-squares SVMs [Suykens and Vandewalle 1999] and linear programming SVMs [Oladunni and Singhal 2009]. Their training time is much shorter than standard SVMs while their classification performance is still comparable to a standard SVM. Inspired by linear programming SVMs, we formulate our objective function based on the L_1 norm as follows.

$$\min_{\mathbf{w},b,\xi} \sum_{j=1}^{m} \|\mathbf{w}_j\|_1 + \lambda \sum_{ij} \xi_{ij}, \qquad (4)$$

where $\mathbf{w}_j$'s and b_j's are considered as the parameters of the multiphase implicit function $\mathcal{F}$. The first term is a regularization term, preventing $\mathbf{w}_j$'s L_1 norm from becoming too large, and the second term minimizes the classification error by minimizing the sum of the slack variables. The first term also has an indirect effect, which maximizes the aforementioned margins between parallel planes. This is because when the norm of both $\mathbf{w}_j$ and $\mathbf{w}_k$ becomes smaller, so does the norm of $\mathbf{w}_j - \mathbf{w}_k$. The objective function in (4) subject to constraints in (3) is equivalent to the following linear program-

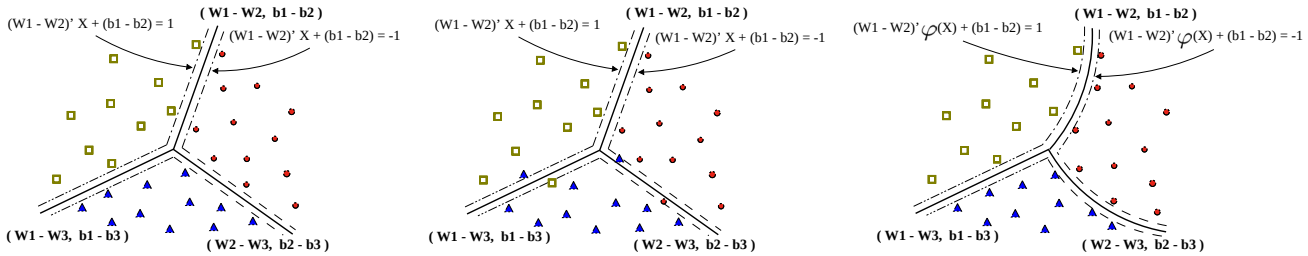


Figure 4: *Multicategory Support Vector Machines. (a) Linear separators with margins; (b) linear separators with slack variables; (c) polynomial separators with margins (separating hyperplanes in a higher-dimensional space).*

ming problem,

$$\begin{aligned} \min_{\mathbf{w},\mathbf{s},b,\xi} & \sum_{j=1}^{m}\sum_{l}\mathbf{s}_j^l + \lambda\sum_{ij}\xi_{ij} \\ s.t. \quad & (\mathbf{w}_{l_i}-\mathbf{w}_j)^T\mathbf{x}_i + (b_{l_i}-b_j) \geq 1-\xi_{ij}, \\ & \mathbf{w}_j \leq \mathbf{s}_j, \quad \mathbf{w}_j \geq -\mathbf{s}_j, \\ & \xi_{ij} \geq 0, \quad \mathbf{s}_j \geq \mathbf{0}, \\ & i=1,\ldots,n, \quad j=1,\ldots,m, \quad j \neq l_i, \end{aligned} \tag{5}$$

where $\mathbf{s}_j$ is an auxiliary vector variable representing the upper bounds of $\mathbf{w}_j$'s elements, and $\mathbf{s}_j^l$ is its l-th component. This linear programming problem can be solved using many software packages such as GLPK (GNU Linear Programming Kit) [Makhorin 2006].

Very often region boundaries are curved and not linearly separable (Figure 4(c)). A polynomial separating surface would be more desired. We will show that the above optimization framework can be generalized to polynomial separating surfaces. This generalization needs the assistance of a nonlinear function $\varphi(\mathbf{x})$, which maps $\mathbf{x} = [x\ y\ z]^T$ into a higher dimensional space. For example, if we would like to solve for quadratic separating surfaces, $\varphi(\mathbf{x}) = [x^2\ y^2\ z^2\ xy\ xz\ yz\ x\ y\ z]$, which is the second-degree polynomial basis without the constant term. Once all the data points have been mapped to such a higher dimensional space, we still follow the above optimization framework to solve for separating hyperplanes. Such hyperplanes in the higher dimensional space actually represent polynomial separating surfaces in the three-dimensional space. Thus, polynomial scalar components in our MIF can be expressed as

$$F_j(\mathbf{x}) = \mathbf{w}_j^T\varphi(\mathbf{x}) + b_j, \ j=1,\ldots,m, \tag{6}$$

And the constraints in (3) should be rewritten as follows,

$$\begin{aligned} (\mathbf{w}_{l_i}-\mathbf{w}_j)^T\varphi(\mathbf{x}_i) + (b_{l_i}-b_j) &\geq 1-\xi_{ij}, \\ \xi_{ij} &\geq 0, \\ i=1,\ldots,n \quad j=1,\ldots,m \quad j &\neq l_i \end{aligned} \tag{7}$$

Therefore, the linear programming problem for polynomial separating surfaces should be

$$\begin{aligned} \min_{\mathbf{w},\mathbf{s},b,\xi} & \sum_{j=1}^{m}\sum_{l}\mathbf{s}_j^l + \lambda\sum_{ij}\xi_{ij} \\ s.t. \quad & (\mathbf{w}_{l_i}-\mathbf{w}_j)^T\varphi(\mathbf{x}_i) + (b_{l_i}-b_j) \geq 1-\xi_{ij}, \\ & \mathbf{w}_j \leq \mathbf{s}_j, \quad \mathbf{w}_j \geq -\mathbf{s}_j, \\ & \xi_{ij} \geq 0, \quad \mathbf{s}_j \geq \mathbf{0}, \\ & i=1,\ldots,n, \quad j=1,\ldots,m, \quad j \neq l_i. \end{aligned} \tag{8}$$

And the dimensionality of $\mathbf{w}_j$'s and $\mathbf{s}_j$'s has become larger.

3.2 Distance Transform as a Multiphase Implicit

Many operations, such as blending, deformation and collision prediction, not only need the region membership of a given point, but also require an estimation of closeness between the point and the implicit surface under a distance measure. In the following, we present a revised multiphase implicit function that facilitates distance estimation from a given point to the multiphase implicit surface it represents in the higher-dimensional space $\varphi(\mathbf{x})$ maps to.

Computing the distance from a mapped point $\varphi(\mathbf{x}_0)$ to the entire mapped multiphase implicit surface is equivalent to computing its distance to the boundary of the mapped region that encloses $\varphi(\mathbf{x}_0)$. We can obtain the region membership of $\mathbf{x}_0$ using our MIF defined in (6). Suppose $\mathbf{x}_0$ belongs to region Ω_j, and N_j denotes the set of regions adjacent to Ω_j. According to the definition of our MIF, the hyperplane that separates $\varphi(\Omega_j)$ from an adjacent region $\varphi(\Omega_k)(k \in N_j)$ is

$$P_{jk} : (\mathbf{w}_j - \mathbf{w}_k)^T\varphi(\mathbf{x}) + (b_j - b_k) = 0.$$

The signed distance from $\varphi(\mathbf{x}_0)$ to one of the bounding hyperplanes of $\varphi(\Omega_j)$, P_{jk}, can be computed as follows,

$$D(\varphi(\mathbf{x}_0), P_{jk}) = \frac{(\mathbf{w}_j - \mathbf{w}_k)^T\varphi(\mathbf{x}_0) + (b_j - b_k)}{\|\mathbf{w}_j - \mathbf{w}_k\|}. \tag{9}$$

Since the mapped training points are linearly separable in the higher dimensional space (otherwise we need to further increase the dimension of the training space), a mapped region bounded by separating hyperplanes is always convex. Thus the distance between $\varphi(\mathbf{x}_0)$ and the boundary of $\varphi(\Omega_j)$ is $\min_{k\in N_j}(D(\varphi(\mathbf{x}_0), P_{jk}))$. Based on this observation, we define the following revised multiphase implicit function:

$$F_j'(\mathbf{x}) = \min_{k\in N_j}(D(\varphi(\mathbf{x}), P_{jk})), \ j=1,\ldots,m, \tag{10}$$

which represents exactly the same multiphase implicit surface as that represented by Equation (6). However, its function value is much more meaningful, and has the following properties.

- $\forall \mathbf{x} \in \Omega_j$, $F_j'(\mathbf{x}) > 0$ and $F_k'(\mathbf{x}) < 0$ $(k = 1,\ldots,m, k \neq j)$.
- If $\mathbf{x}_0$ lies on the separating surface between Ω_j and Ω_k, $F_j'(\mathbf{x}_0) = 0$ and $F_k'(\mathbf{x}_0) = 0$.
- $\forall \mathbf{x} \in \Omega$, $\max_j(F_j'(\mathbf{x}))$ $(j = 1,\ldots,m)$ is the unsigned distance from $\varphi(\mathbf{x})$ to the entire mapped multiphase implicit surface.

Note that if linear separators are used, the distance in Equation (9) is exactly the 3D Euclidean distance. This is not the case for higher-degree polynomial separating surfaces. However, in applications

that do not require exact Euclidean distance, we can use the distance in (9) as an estimation of closeness since computing the Euclidean distance from a point to a higher-degree polynomial surface is much more computationally expensive.

4 Piecewise Blended Multiphase Implicits

Inside a heterogeneous volume object, the boundary surface of a region may have local geometry that is hard to model using polynomials. Furthermore, 'region' is only a logical concept and one region may have multiple disconnected subregions assigned with the same region label. It would also be hard to model such disconnected subregions using a polynomial. Because the level of structure complexity that can be modeled with a polynomial is limited, we propose to perform adaptive spatial subdivision and represent subdivided boundary surfaces with a piecewise polynomial multiphase implicit.

We choose octree-based adaptive subdivision because every subdivided node in an octree is an axis-aligned cube, which facilitates subsequent blending operations. For every octree node, we compute a local polynomial multiphase implicit by solving the linear programming problem in (8) on the subset of training data falling into a bounding sphere of the octree node. The radius of the bounding sphere is $r = \alpha u$, where u is the size of the node, and α is a constant. We make the sphere larger than the tightest bounding sphere so that we can include more training points to make the resulting local multiphase implicit surface more spatially coherent. In all experiments, we set $\alpha = 2$. A kd-tree is used for efficiently collecting the subset of training points. If the bounding sphere is completely located inside a region, in another word, all the training points inside the bounding sphere have the same label, there is no need to train a local MIF. In that case, we simply store the label of the region in the node. In addition, we can also save other spatially varying attributes in the node by fitting an analytic model, such as a polynomial, to such attribute values.

According to the result of linear programming in (8), if the maximum slack variable $\max(\xi_{ij})$ in (8) exceeds a prescribed threshold ξ_{max} or the number of misclassified training points is larger than a predefined value n_{miss}, we subdivide the octree node and recurse on every child node. In our experiments, we typically set $\xi_{max} = 0.01$ and $n_{miss} = 10$.

Note that even when there are a large number of regions in a volume, the number of different regions within an octree leaf node is unlikely to exceed four, and we only need to represent the scalar components corresponding to these regions while leaving the other scalar components undefined. This is because at a non-manifold feature, there are typically only three or four simultaneously adjacent regions.

Solving a large linear programming problem when training a local MIF can still be time-consuming. We use the following strategies to prune training data. First, in most cases, data points near region boundaries are sufficient to train a local MIF. Therefore we discard most data points that are not closest to region boundaries. To prevent the trained MIF from having incorrect labels away from region boundaries, we take some sample points on the bounding sphere under consideration as training points if they do not introduce additional labels into the local training dataset. Those sample points are generated by projecting the eight corners and six face centers of the octree node to the bounding sphere. Second, if the total number of data points inside a bounding sphere is larger than a predefined value, n_{max}, we randomly choose n_{max} points from them as the training points. However, all the original data points will participate in classification error evaluation to help determine whether the octree node needs to be subdivided. In our experiments, we set $n_{max} = 150$.

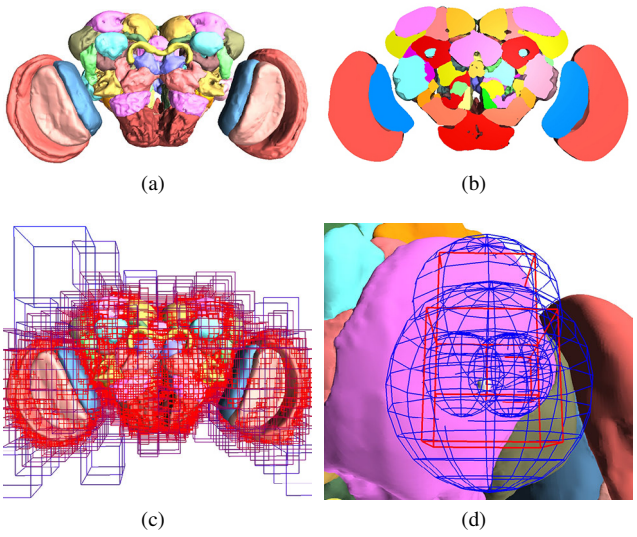


Figure 5: *(a) A MIF for a fruit fly brain dataset, (b) A vertical cross section of (a), (c) octree leaf nodes containing local MIFs. Color codes represent the size of the leaf nodes. The smallest leaf nodes are shown in red, and the largest ones are shown in blue. (d) illustrates the blending process. The white sphere is a query point. The octree leaf nodes participating in the blending are shown in red. Their support spheres are shown in blue.*

Once we have computed all local MIFs, we need to blend them together to form a spatially coherent global MIF. For ease of understanding, we illustrate our blending technique using a 2D example shown in Figure 6. In this example, there are two MIFs from two adjacent octree nodes. Inside each node, there are three regions separated by three planes. Each plane has its own signed distance function (SDF). Once we have blended the SDFs of every pair of corresponding planes in the two MIFs, we define the separating planes (shown in red) of the blended MIF using the blended SDFs. In general, there might be many local MIFs participating in the blending. We use the blending function in [Ohtake et al. 2003] to blend signed distances to corresponding hyperplanes from different local MIFs. Our global blended MIF is defined as

$$F_j^b(\mathbf{x}) = \min_{k \in N_j} \left(\Sigma_{i=1}^{n_f} \alpha^{(i)}(\mathbf{x}) D\left(\varphi(\mathbf{x}), P_{jk}^{(i)}\right) \right), \; j = 1, \ldots, m,$$

$$\alpha^{(i)}(\mathbf{x}) = \frac{\rho^{(i)}(\mathbf{x})}{\Sigma_{j=1}^{n_f} \rho^{(j)}(\mathbf{x})}, \tag{11}$$

where $P_{jk}^{(i)}$ is the hyperplane separating Ω_j from Ω_k in the local MIF, $\mathcal{F}_i$, n_f is the number of local MIFs, $\alpha^{(i)}(\mathbf{x})$ is a nonnegative compactly supported function associated with $\mathcal{F}_i$ and $\rho^{(i)}(\mathbf{x})$ is a compactly supported weighting function for $\mathcal{F}_i$. In our implementation, we define the weighting function $\rho^{(i)}(\mathbf{x})$ using a quadratic B-spline $B(t)$ as in [Ohtake et al. 2003]:

$$\rho^{(i)}(\mathbf{x}) = B\left(\frac{3|\mathbf{x} - \mathbf{c}_i|}{2\beta u_i}\right), \tag{12}$$

where $\mathbf{c}_i$ is the center of the corresponding leaf node of the octree, u_i is the size of the node and β is a constant. In all our experiments, we set $\beta = 1.75$. Note that the MIN operator is applied after blending. It is used to choose the closest blended separating surface,

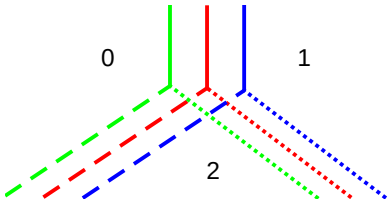


Figure 6: *2D example to illustrate our blending scheme. There are three regions labeled 0, 1 and 2 respectively and two SVMs shown in green and blue respectively. The corresponding separating planes from the two SVMs are shown in the same line style. The blended result is shown in red.*

which changes only at sharp features and non-manifold features. Thus the MIN operator does not affect the degree of continuity on the rest of the implicit surface. Since the weighting function is a quadratic B-spline, the final implicit surface in (11) is C^0 at sharp features and non-manifold features and C^1 elsewhere.

Figure 5 shows a MIF for a fruit fly brain. And Figure 7 shows a MIF constructed from another 3D segmented medical dataset from [3D-IRCADb].

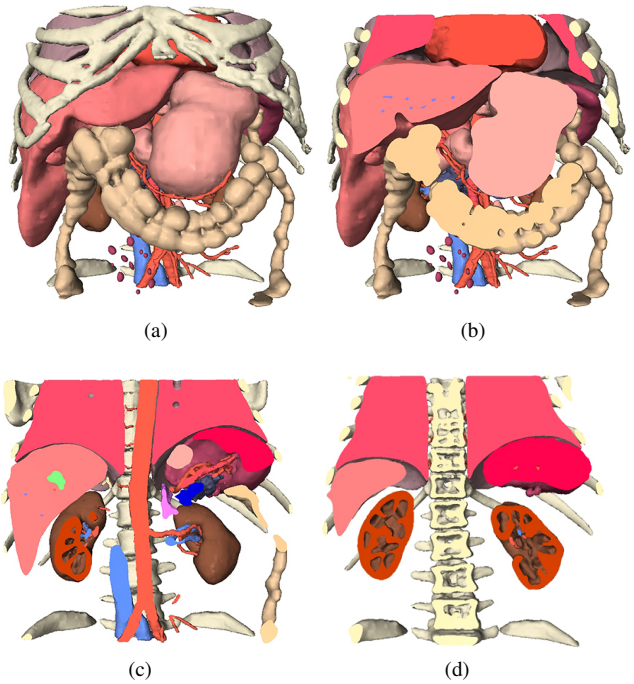


Figure 7: *(a) A MIF constructed from a segmented 3D medical image of a section of the human body, (b)-(d) three vertical cross sections. Original segmented data courtesy of IRCAD.*

In all our results in this paper, we use piecewise quadratic MIFs. In comparison with piecewise linear MIFs, piecewise quadratic MIFs significantly reduce the number of leaf nodes in an octree, making the representation much more compact. On the other hand, higher-order polynomials are much more expensive to compute while only marginally more compact. A quadratic separating surface in an octree node could be a hyperbolic surface with two sheets, one of which is not generated from training but simply as the companion of the other. When this happens, we further check whether both sheets are inside the bounding sphere. If the companion sheet is outside the bounding sphere, it has zero weight and does not affect the shape of the final implicit surface. Otherwise, we simply subdivide the octree node. If we obtain a hyperbolic surface at the prescribed maximum depth, d_{max}, of the octree, we fall back to compute a linear MIF. We typically set $d_{max} = 9$.

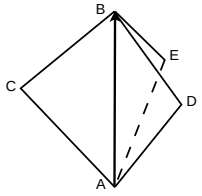
5 Computing MIFs from Meshes

Meshes are the most popular choice for representing 3D objects probably because it is convenient to construct and manipulate a mesh during interactive modeling or after data acquisition from scanning devices. Therefore meshes, either manifold or non-manifold ones, are one of our most important data sources for creating multiphase implicit functions.

5.1 MIFs from Closed Non-Manifold Meshes

A closed non-manifold mesh partitions the 3D space $\mathcal{R}^3$ into a set of regions $(\Omega_1, \Omega_2, \ldots, \Omega_m)$. Such a non-manifold mesh can be either directly extracted from segmented medical images or created by fusing multiple cross sections [Boissonnat and Memari 2007; Liu et al. 2008]. In this subsection, we focus on converting such non-manifold meshes to MIFs. During such a conversion, the most important unsolved issue is how to obtain a set of labeled sample points within every region of the mesh. This is because the algorithm developed in Section 3.1 can be readily applied to obtain a MIF once such labeled sample points become available. In the following, we present an algorithm for extracting labeled sample points from a closed non-manifold mesh, which is assumed to be free of holes and foldovers.

A face in a closed non-manifold mesh is always on the boundary of two adjacent regions since the space outside the mesh is also counted as a region. Because of this, we conceptually split every face into two half faces, each of which belongs to one region. The normal of a half face always points to the interior of the region the half face belongs to. Suppose we adopt the right hand rule when defining the normal of a half face. That means the vertices of the half face follow counterclockwise order around its normal. We adopt a clustering approach to determine the region membership of every half face. We first create an initial cluster for every half face, then repeatedly perform the following cluster merge step until convergence. Let $\triangle_h ABC$ be one of the half faces. There may be more than two faces sharing the same edge AB when the mesh we process is a non-manifold surface. We sort these faces so that they follow a counterclockwise order around the vector $\vec{AB}$. If $\triangle ABD$ is the face following $\triangle ABC$ in this sorted order and $\triangle_h BAD$ belongs to a cluster that is different from the one $\triangle_h ABC$ belongs to, we merge the two clusters $\triangle_h ABC$ and $\triangle_h BAD$ belong to. Convergence has been reached if such half face pairs do not exist any more. The final number of clusters is equal to the number of regions in the non-manifold mesh. We assign a distinct region label to each of them. All the half faces in a cluster are assigned the same region label.



In the next stage, we build an octree for the mesh. We start from the bounding cube of the non-manifold mesh, and recursively subdivide the octree nodes overlapping with the mesh until a predefined depth d_{max} is reached. All the leaf nodes intersecting with the mesh are marked as *boundary nodes* and the rest of the leaf nodes are marked as *internal nodes*. To determine the region label of an internal node,

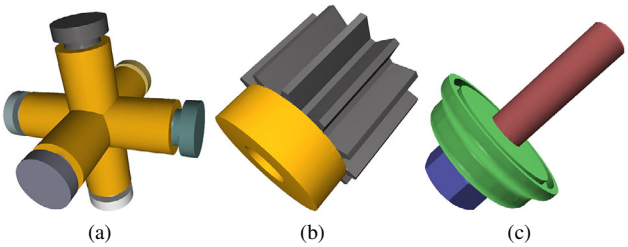


Figure 8: *MIFs converted from non-manifold meshes of mechanical parts. Original meshes courtesy of Drexel University.*

we trace a ray from the center of the node along a random direction until it hits a half face, whose region label is then assigned to the internal node.

Once we have obtained the region membership of all half faces and internal leaf nodes, we can easily use such information to determine the region label of any sample point, $\mathbf{x}$, as follows. If $\mathbf{x}$ falls in an internal leaf node, we can simply assign it the label of the node. If it is outside the octree, we assign it the label of the outmost region. If it falls in a boundary leaf node, we start ray tracing from $\mathbf{x}$ following a random direction. The ray either enters an internal leaf node or hits a half face. In either case, the label of the internal node or half face is assigned to $\mathbf{x}$.

Finally, we draw sample points within every internal leaf node as well as on every half face of the mesh. For each sample point on a half face, we further offset it along the normal direction of the half face to obtain a final sample point. The region label of every sample point is determined using the aforementioned method. Then we can use the method in Section 3.1 to compute a MIF from these labeled training points.

It is important to preserve sharp features in a MIF. Note that a sharp feature shared by three or more regions can be well preserved by the MIF because, by definition, the feature is the intersection of multiple separating hyperplanes. However, it is hard to use a single MIF to preserve sharp features on a boundary shared by exactly two regions. In that case, we detect sharp features by checking the dihedral angle between every pair of adjacent faces. If the dihedral angle between two faces is smaller than a predefined value θ, we mark their shared edge as a feature edge. We typically set $\theta = \frac{2\pi}{3}$. All feature edges inside an octree node separate the triangle faces therein into several groups. We optimize a distinct elementary MIF for each group. Then, as in [Ohtake et al. 2003], we construct a composite MIF with sharp features from these elementary MIFs using Boolean operators [Ricci 1973]. This composite MIF is the final local MIF stored in the octree node and used for blending with other local MIFs.

Figures 8 and 9 show examples of MIFs generated from non-manifold meshes.

5.2 MIFs from Multiple Manifold Meshes

One of the most obvious choices would be using multiple manifold meshes to create a volume with multiple internal regions, and further training a MIF to represent the geometry of this complex volume. To facilitate region label determination in a later step, we again build an octree for each of the manifold meshes, and obtain a binary label for every internal leaf node of the octree and every half face in the mesh by following a similar procedure as in the previous subsection. This binary label indicates whether an octree node or a half face is inside the mesh object or not. We further define a distinct Boolean variable for every participating mesh. Any region created from the meshes can be defined with a Boolean expression over these variables.

We draw sample points inside the entire volume as well as on every face of the individual meshes. For each sample point on a face, we duplicate it and further offset the resulting two points respectively along the positive and negative normal direction of the face to obtain two final sample points. Then we test every sample point against each of the constructed octrees to determine whether it is inside each of the original meshes. The result is a sequence of binary values. The region membership of a sample point can be determined straightforwardly by evaluating the aforementioned Boolean expressions using these binary values. Finally we apply the method in Section 3.1 to compute a MIF from these labeled training points.

Figures 10, 11 and 1-bottom show examples of MIFs generated from manifold meshes.

6 Experimental Results

We have implemented the algorithms presented in previous sections and successfully used them to compute MIFs from the following three types of datasets, segmented 3D medical images (Figs. 5 and 7), non-manifold meshes (Figs. 8 and 9), and manifold meshes (Figs. 10, 11 and 1-bottom). Our algorithms only involve a small number of parameters, and their setting has been discussed in previous sections. In particular, the parameter λ in (8) can be used as a control knob to provide tradeoff between visual quality and numerical accuracy. Results generated with smaller λ values are smoother and visually more appealing while those generated with larger λ values are numerically more accurate. All experimental results were generated on an Intel Xeon X5690 3.47GHz processor with 24GB RAM. Table 1 summaries the number of regions, the number of local MIFs computed, and the storage cost for all datasets and results.

Note that, when working with SVMs, appropriate scale normalization of the training points can significantly improve classification accuracy. To perform this normalization, we scale the bounding sphere of the training points to a unit sphere. The coordinates of the training points inside the bounding sphere are scaled accordingly before solving the linear programming in (8). When we use the resulting MIF to compute the region label of a given point, the same scaling must be applied to the point as well.

Images shown in this paper were generated using ray tracing. The octree data structure not only holds our local MIFs, but also pro-

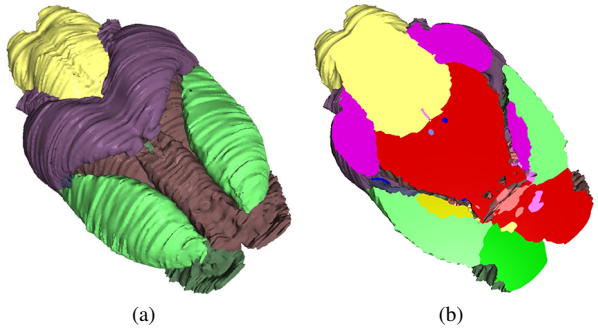
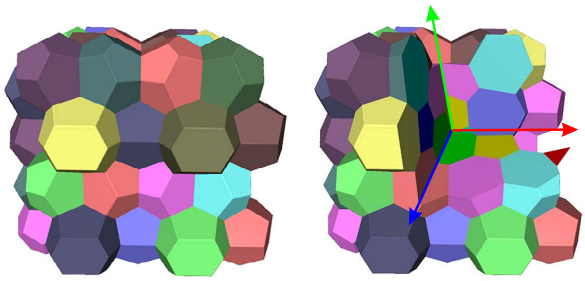


Figure 9: *(a) A MIF converted from a non-manifold mesh modeling a segmented mouse brain, (b) a cross section of (a). Original mesh courtesy of Tao Ju (WUSTL).*



(a) (b)

Figure 10: *A MIF constructed from a section of the Weaire-Phelan structure. The Weaire-Phelan structure uses cells of equal volume to fill up the entire 3D space [Weaire and Phelan 1994]. An important property of the cells is that the ratio between their surface area and volume is near-optimal, i.e. close to the minimum, if not already the minimum, among all similar space-filling cells. Original mesh courtesy of Lvdi Wang (MSRA).*

vides a space partition that accelerates ray tracing. We first record the region enclosing the start point of the ray as current region. If a ray enters the bounding sphere of a leaf node holding a local MIF, we compute the intersection between the ray and the separating surface defined by the local MIF. In this way, we can identify the closest local MIF intersecting the ray as well as the intersection point $\mathbf{x}$. Note that this intersection may not lie on the blended global MIF. Thus, we first search around $\mathbf{x}$ for two points $\mathbf{x}_{in}$ and $\mathbf{x}_{out}$ lying inside and outside the current region, respectively, and then use binary search to find an approximate intersection between the ray and the global MIF. The final pixel color is computed using the predefined material color of the region which the ray enters.

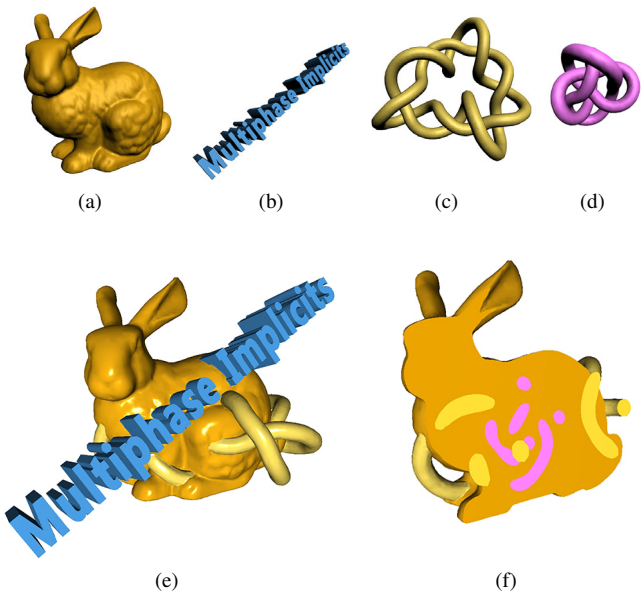
6.1 Comparisons

There has been little work on true object-space multiphase implicits. We have compared our piecewise polynomial multiphase implicits with two popular implicit approaches for representing multiple regions on representation (classification) accuracy, memory consumption and running time. One of these approaches defines a two-phase implicit function for every region and stacks them together to "simulate" a multiphase implicit function. A point is classified to the region whose corresponding two-phase implicit function returns the maximum value. We chose the multi-level partition of unity implicits (MPU) in [Ohtake et al. 2003] as the two-phase implicit function in this approach.

The second approach we compare with is the binary SDF tree in [Wang et al. 2011]. In this approach, the regions are organized as

	#local MIFs	#regions	memory cost
Fruit fly	36617	63	6.11MB
Human torso	47948	21	7.85MB
Mouse brain	55554	46	9.42MB
Mechanical A	2818	8	0.59MB
Mechanical B	3386	3	0.75MB
Mechanical C	6840	4	1.31MB
Weaire-Phelan	21232	65	3.34MB
Bunny et al.	35967	5	6.71MB
Neptunes	200694	334	30.83MB

Table 1: *Summary statistics for all datasets and results.*



(a) (b) (c) (d)

(e) (f)

Figure 11: *(e) A MIF converted from four manifold meshes shown in (a)-(d). (f) A cross section of (e). Note that the pink knot model is inside the* BUNNY. *Original meshes courtesy of Stanford University Computer Graphics Lab (a) and The University of British Columbia Imager Lab (c-d).*

leaf nodes in a binary tree. At each intermediate node of the tree, there is an SDF that defines the boundary between the regions in its left subtree and those in its right subtree. In our implementation, each SDF in an SDF tree is again represented as an MPU implicit instead of an interpolated uniform grid as in [Wang et al. 2011] because an MPU implicit is more memory efficient.

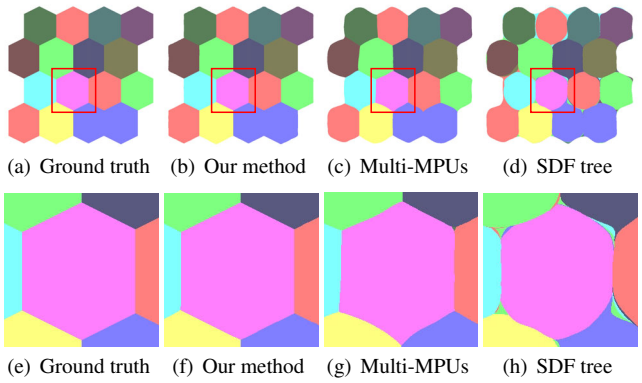
The comparison was performed on three representative datasets with a reasonably large number of regions. Numerical results for typical parameter settings are shown in Table 2. According to the comparison results, with similar or even less memory consumption, our multiphase method achieves an error rate one order of magnitude smaller than the two alternatives, which leverage two-phase implicits. Note that in all three methods the classification error varies with the error tolerance/threshold used. Smaller error tolerance values/thresholds in general increase memory consumption. Nevertheless, as long as memory consumption is kept the same among all methods, our method exhibits similar degrees of improvement in accuracy for other error tolerance settings.

7 Conclusions and Future Work

In this paper, we have presented a novel class of object-space multiphase implicit functions that are capable of accurately and compactly representing objects with multiple internal regions. Our proposed multiphase implicit functions facilitate true object-space geometric modeling of heterogeneous objects with non-manifold features. We have also presented multiple methods to create object-space multiphase implicit functions from existing data, including meshes and segmented medical images. Our algorithms were inspired by machine learning algorithms for training multicategory max-margin classifiers. Comparisons with existing techniques demonstrate the superiority of our approach.

dataset	method	error rate	memory cost	constr. time	query time (10^6 pts)
I. Fruit fly brain	Our method	0.255%	6.11MB	158.84s	2.02s
	Multi-MPUs	2.29%	14.28MB	149.11s	156.38s
	SDF tree	3.63%	29.79MB	36.24s	14.77s
II. Human torso	Our method	0.159%	7.85MB	205.19s	1.44s
	Multi-MPUs	1.38%	37.49MB	89.03s	53.54s
	SDF tree	2.26%	61.76MB	39.83s	7.17s
III. Weaire-Phelan	Our method	0.139%	3.34MB	329.03s	2.33s
	Multi-MPUs	1.31%	3.37MB	667.28s	111.92s
	SDF tree	2.31%	7.95MB	191.72s	12.00s

Table 2: *Comparison of overall accuracy, memory cost and running times (construction time & query time) among three methods over three datasets, fruit fly brain, human torso, and the Weaire-Phelan structure. The error tolerance values used in MPU for the three datasets are 0.015, 0.0095, and 0.022, respectively. In our method, we set* $\lambda = 200$, $\xi_{max} = 0.01$, $n_{miss} = 10$ *and* $d_{max} = 9$. *Note that our method achieves much higher accuracy and much faster query processing than the other two alternatives.*



(a) Ground truth (b) Our method (c) Multi-MPUs (d) SDF tree

(e) Ground truth (f) Our method (g) Multi-MPUs (h) SDF tree

Figure 12: *A qualitative comparison of internal region boundaries between our method and two alternatives (multi-MPUs and SDF tree) using a slice of implicit models constructed for the Weaire-Phelan structure. (e)-(h) are magnified views of the highlighted regions in (a)-(d), respectively. Our result is very close to the ground truth while distorted boundaries and tiny disconnected regions may exist in results from the other two alternative methods.*

Limitations. There are aspects of our algorithms that require future work. For example, our current algorithm for converting meshes to multiphase implicit functions assume the meshes are "clean", i.e. they are free of defects, such as gaps, foldovers, holes and "slivers". In fact, removing defects is one of the most important benefits of converting meshes to implicit surfaces [Shen et al. 2004]. A common practice for cleaning up a mesh is to first convert it to an implicit surface and then polygonize the implicit surface. We would like to generalize this pipeline to non-manifold meshes and use our multiphase implicit functions as the intermediate implicit representation.

Acknowledgements

We would like to thank Hanchuan Peng and co-authors [Peng et al. 2011] for sharing the fruit fly dataset, Tao Ju for the mouse brain dataset, Lvdi Wang for a polygonal model of the Weaire-Phelan structure, and the anonymous reviewers for their valuable comments. This work was partially supported by National Science Foundation (IIS 09-14631) and Hong Kong Research Grants Council under General Research Funds (HKU718712).

References

3D-IRCADB. 3d-ircadb (3d image reconstruction for comparison of algorithm database). `http://www.ircad.fr/softwares/3Dircadb/3Dircadb1/index.php`.

ANDERSON, J., GARTH, C., DUCHAINEAU, M., AND JOY, K. 2010. Smooth, volume-accurate material interface reconstruction. *Visualization and Computer Graphics, IEEE Transactions on 16*, 5, 802–814.

BLINN, J. 1982. A generalization of algebraic surface drawing. *ACM Transactions on Graphics 1*, 235–256.

BOISSONNAT, J.-D., AND MEMARI, P. 2007. Shape reconstruction from unorganized cross-sections. In *Fifth Eurographics symposium on Geometry processing*, 89–98.

BREDENSTEINER, E., AND BENNETT, K. 1999. Multicategory classification by support vector machines. *Computational Optimization and Applications 12*, 1, 53–79.

CARR, J., MITCHELL, T., BEATSON, R., CHERRIE, J., FRIGHT, W., MCCALLUM, B., AND EVANS, T. 2001. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH 2001 Conference Proceedings*, 67–76.

FENG, P., JU, T., AND WARREN, J. 2010. Piecewise tri-linear contouring for multi-material volumes. *Advances in Geometric Modeling and Processing*, 43–56.

FRISKEN, S., PERRY, R., ROCKWOOD, A., AND JONES, T. 2000. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *Proceedings of SIGGRAPH 2000*, 249–254.

GASCUEL, M.-P. 1993. An implicit formulation for precise contact modeling between flexible solids. In *Proceedings of SIGGRAPH 93*, Computer Graphics Proceedings, Annual Conference Series, 313–320.

LIU, L., BAJAJ, C., DEASY, J., LOW, D. A., AND JU, T. 2008. Surface reconstruction from non-parallel curve networks. *Computer Graphics Forum 27*, 2, 155–163.

LORENSEN, W. E., AND CLINE, H. E. 1987. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics 21*, 4, 163–169.

LOSASSO, F., SHINAR, T., SELLE, A., AND FEDKIW, R. 2006. Multiple interacting liquids. In *ACM Transactions on Graphics (TOG)*, vol. 25, ACM, 812–819.

MAKHORIN, A., 2006. GLPK (GNU linear programming kit). http://www.gnu.org/software/glpk/.

OHTAKE, Y., BELYAEV, A., ALEXA, M., TURK, G., AND SEIDEL, H.-P. 2003. Multi-level partition of unity implicits. In *ACM Transactions on Graphics (TOG)*, vol. 22, ACM, 463–470.

OLADUNNI, O., AND SINGHAL, G. 2009. Piecewise multi-classification support vector machines. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, IEEE, 2323–2330.

PENG, H., CHUNG, P., LONG, F., QU, L., JENETT, A., SEEDS, A., MYERS, E., AND SIMPSON, J. 2011. Brainaligner: 3d registration atlases of Drosophila brains. *Nature Methods 8*, 6, 493–498.

RICCI, A. 1973. A constructive geometry for computer graphics. *The Computer Journal 16*, 2, 157–160.

SCHÖLKOPF, B., GIESEN, J., AND SPALINGER, S. 2005. Kernel methods for implicit surface modeling. In *Advances in Neural Information Processing Systems 17*, MIT Press, Cambridge, MA, 1193–1200.

SHEN, C., O'BRIEN, J., AND SHEWCHUK, J. 2004. Interpolating and approximating implicit surfaces from polygon soup. *ACM Transactions on Graphics 23*, 3, 896–904.

STALLING, D., ZÖCKLER, M., AND HEGE, H. 1998. Interactive segmentation of 3d medical images with subvoxel accuracy. In *Proc. CAR'98 Computer Assisted Radiology and Surgery*, 137–142.

STEINKE, F., SCHLKOPF, B., AND BLANZ, V. 2005. Support vector machines for 3d shape processing. *Computer Graphics Forum 24*, 3, 285–294.

SUYKENS, J., AND VANDEWALLE, J. 1999. Multiclass least squares support vector machines. In *International Joint Conference on Neural Networks (IJCNN'99)*.

TAKAYAMA, K., SORKINE, O., NEALEN, A., AND IGARASHI, T. 2010. Volumetric modeling with diffusion surfaces. In *ACM Transactions on Graphics (TOG)*, vol. 29, ACM, 180.

TURK, G., AND O'BRIEN, J. 2002. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics 21*, 4, 855–873.

VAPNIK, V. 1998. *Statistical learning theory*. John Wiley & Sons, Inc.

WANG, L., YU, Y., ZHOU, K., AND GUO, B. 2011. Multiscale vector volumes. *ACM Trans. Graph. 30*, 6, 167.

WEAIRE, D., AND PHELAN, R. 1994. A counter-example to kelvin's conjecture on minimal surfaces. *Philosophical Magazine Letters 69*, 2, 107–110.

WYVILL, G., MCPHEETERS, C., AND WYVILL, B. 1986. Data structure for soft objects. *Visual Computer 2*, 4, 227–234.

WYVILL, B., GUY, A., AND GALIN, E. 1999. Extending the csg tree. warping, blending and boolean operations in an implicit surface modeling system. *Computer Graphics Forum 18*, 2, 149–158.

YAMAZAKI, S., KASE, K., AND IKEUCHI, K. 2002. Non-manifold implicit surfaces based on discontinuous implicitization and polygonization. In *Proc. Geometric Modeling and Processing 2002*, 138–146.

ZHANG, Y., HUGHES, T., AND BAJAJ, C. 2010. An automatic 3d mesh generation method for domains with multiple materials. *Computer methods in applied mechanics and engineering 199*, 5, 405–415.

ZHAO, H.-K., CHAN, T., MERRIMAN, B., AND OSHER, S. 1996. A variational level set approach to multiphase motion. *Journal of Computational Physics 127*, 179–195.