Text Scaffolds for Effective Surface Labeling

Gregory Cipriano, *Student Member, IEEE*, and Michael Gleicher

Abstract—In this paper we introduce a technique for applying textual labels to 3D surfaces. An effective labeling must balance the conflicting goals of conveying the shape of the surface while being legible from a range of viewing directions. Shape can be conveyed by placing the text as a texture directly on the surface, providing shape cues, meaningful landmarks and minimally obstructing the rest of the model. But rendering such surface text is problematic both in regions of high curvature, where text would be warped, and in highly occluded regions, where it would be hidden. Our approach achieves both labeling goals by applying surface labels to a 'text scaffold', a surface explicitly constructed to hold the labels. Text scaffolds conform to the underlying surface whenever possible, but can also float above problem regions, allowing them to be smooth while still conveying the overall shape. This paper provides methods for constructing scaffolds from a variety of input sources, including meshes, constructive solid geometry, and scalar fields. These sources are first mapped into a distance transform, which is then filtered and used to construct a new mesh on which labels are either manually or automatically placed. In the latter case, annotated regions of the input surface are associated with proximal regions on the new mesh, and labels placed using cartographic principles.

Index Terms—surface labeling, computational cartography, text authoring, annotation

1 Introduction

Text labels are useful to annotate features when viewing 3D models. Good labelings must address several sometimes conflicting issues: they must be legible (the text is readable), visible (the text can be seen), proximal (the labels are visually connected to the feature they annotate), and shape-conveying (they should help, not hinder the understanding of the underlying data). Cartographers have considered similar issues on 2D maps for millennia, forming a highly developed art. However, for viewing of 3D models, especially free-form models or applications where the user controls the viewpoint, many of the issues become more challenging. In this paper, we introduce *Text Scaffolds*, an approach for textual labeling in 3D model visualization that offers better control over the tradeoffs, often allowing all 4 goals to be met simultaneously.

There are two general strategies for annotating 3D models: placing the text on screen-aligned "billboards" or applying the text as a surface-adhering label (i.e. as a texture). The former easily meets the goals of legibility and visibility and is easy to implement. However, connection to the model can be difficult, especially as the shapes grow more complex or the number of annotations grows, and they can detract from the perception of shape (by occluding it) without the potential for elucidating it.

In contrast to screen-aligned labels, surface labels can add to shape perception by providing texture cues and motion cues (if the viewpoint is moved). However, text placed on curved surfaces may be warped (hurting legibility), or occluded (hurting visibility). Sometimes there may not even be an appropriate surface at the scale of a feature. Finally, applying the labels to free form surfaces can be challenging as it requires low-distortion parameterizations and anti-aliasing to give the crisp edges helpful for text legibility.

In this paper, we introduce a new approach for the display of textual labels to annotate features on 3D models. Our key insight is that surface labels are best at roughly conveying large-scale shape, and that the issues are most challenging when the features are small. Effectively, surface text labels work well on some surfaces, and not on others. Our basic premise is to construct a surface that is similar to the original yet amenable to surface labeling and apply surface labels to this new surface. We call such surfaces *text scaffolds.*

• *Authors are with the Department of Computer Sciences, University of Wisconsin, Madison, E-mail: gregc@cs.wisc.edu gleicher@cs.wisc.edu.*

Manuscript received 31 March 2008; accepted 1 August 2008; posted online 19 October 2008; mailed on 13 October 2008.
For information on obtaining reprints of this article, please send e-mailto:tvcg@computer.org.

Figure 1 shows some examples of labels that illustrate the advantages of our approach. Unlike screen-space labels, our scaffold labels convey a sense of the rough shape of the underlying model and features, especially when the viewpoint is moved. While they do not precisely show where the feature lies on the surface, they are often close enough to provide a visual connection. Their proximity also helps with managing clutter as the number of labels increases. Unlike direct surface labeling (Fig 2), the scaffold text retains its legibility over small surfaces features as it is not warped by curvature details. Small pockets or bumps do not cause occlusion problems, and the label is visible when the feature is visible. Scaffolds can be constructed even when there is no obvious surface for the label (Figure 1c), and our methods can construct scaffolds from a wide variety of data.

Our discussion of the text scaffold approach begins by considering how the goals for good labeling dictate the kinds of surfaces that should be used as scaffolds. §3 introduces an approach for producing such surfaces from various source data. The approach uses signed distance fields internally, but produces a smooth triangle mesh that both approximates the original data yet is amenable for labeling. §4 then considers the problem of creating the labels on this surface. Methods are described that automatically choose paths for text to follow along the surfaces. Finally, we present an implementation that uses recent advances in surface parametrization and texture filtering to display the labels in an efficient and visually crisp manner.

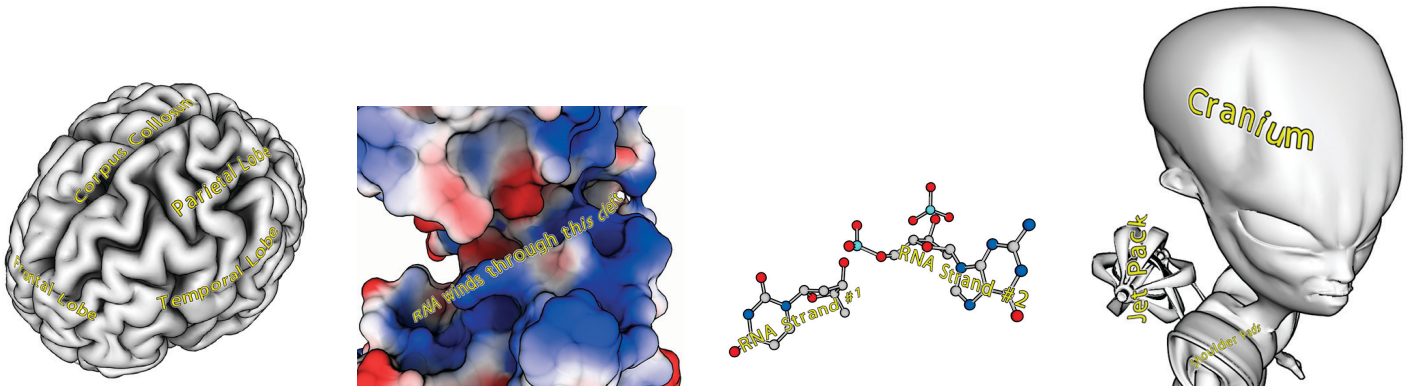
1.1 Contributions

The core contribution of this paper is a novel approach to presenting textual annotations on arbitrary 3D surfaces, one which conveys shape and maintains readability to a degree not possible with prior approaches. In creating this approach, we make the following contributions:

- we explore the issues in effectively presenting text on surfaces, defining the types of surfaces that are likely to be amenable for labeling;
- we introduce a methodology for constructing surfaces with these properties that is efficient enough to allow for experimentation, yet flexible enough to work from a wide-variety of source data;
- we consider the problem of label placement on 3D surfaces, providing an extension of cartographic principles and algorithms;
- we consider the problem of rendering textual labels on 3D surfaces, providing an implementation that is simple and efficient.

2 Related Work

Many papers deal with the task of automated label placement as 2D external labels, some for non-interactive cartographic applications [4]



(a) labeling MRI volume data (b) labeling a protein surface (c) labeling atomic structure (d) labeling an alien

Fig. 1. Examples illustrate Text Scaffold Labeling on a variety of data types and application domains. In all figures, $w_a = 1.0$ and $w_d = 2.0$.

others for interactive 3D illustrations [1, 27]. These techniques can generally work independently from underlying geometry, and have the advantage of being generalizable across a wide range of structures [25].

Work by Ropinski, et al., most resembles our own [31]. Their method fits labels to segmented regions of a smooth approximation of the depth map of a particular view. Unlike our method, which considers the surface as a whole, this optimizes for only one viewpoint, and further, does not allow label placement over disconnected regions, such as those of Figure 1 (a, c and d).

Papers by Götzelmann, et al., have dealt with the concerns of data management for annotation [15], integrating internal and external labels [12] and the need for methods to annotate animated 3D objects [16].

While most cartography texts (c.f. [35]) discuss the issues in textual labeling on maps, the work of Imhof [22] is often considered the seminal authority. The guidelines are typically very specific to 2D maps, although cartographers are beginning to consider their extension to interactive of animation maps [17], and 3D flyovers [18, 19]. We have used these discussions to inform our design.

While there has been considerable interest in automating cartographic label placement, the focus has generally been on labeling large numbers of point features to provide labels that are as close as possible, without overlap. We do not consider interactions between labels (yet), although this problem may be considered in our framework in the future. Automating the labeling of lineal and areal features in order to achieve legible labels that convey feature shape has received much less attention. An early, but comprehensive solution was demonstrated by Freeman and Ahn [8]. Plumer, et al. [7] consider several approaches for automatically placing labels in areal regions, and suggest the skeletonisation approach of Freeman and Ahn as the most effective at conveying shape. We employ this approach in §4.4.2.

Quantitative metrics for assessing labeling, such as [38] and [20], have been proposed. However, these mainly consider interactions between labels, and the aesthetic issues particular to 2D maps.

3 Text Scaffolds

The key element of our approach is to create a scaffold surface specifically designed to accept labels. This surface is constructed such that labels applied anywhere on it will meet our goals for good labels. Conversely, the specific goals dictate the properties this surface must have:

- For the labels to be *legible*, the surface must be smooth (or developable) enough that text on it will not be distorted.
- For the labels to be *visible* from a wide range of viewpoints, all points on the surface should be visible from a range of directions. Holes and divots should be eliminated to avoid highly-occluded regions. The scaffold must remain outside of the initial surface (otherwise the surface would occlude it).
- For the labels to be *proximal*, the scaffold must be close to the original surface. This is important to facilitate connection between the label and the feature it annotates.
- For the labels to be *shape conveying* they should follow the basic shape of the original surface. Approximation partially implies shape conveyance. There is evidence that texture can help in shape perception (c.f. [5, 26, 11]), particularly when then object moves. Textures (particularly ones not specifically designed to convey shape) primarily convey low frequency shape information. This suggests the use of texture to convey low frequency shape by applying it to a smoothed version of the surface (i.e. one that only has the low frequencies).

These goals necessarily conflict when the surface has high frequency detail. The scaffolds, and therefore the labels applied to them, deviate from the original surface and do not convey its high frequency details. Control of this tradeoff is a key element in our design.

To meet our goals, scaffold generation must create a smooth approximation to the input surface where all points on the scaffold are visible from a wide range of directions. The process of scaffold generation takes as input a description of the initial surface in a variety of forms (our implementation supports triangle meshes, constructive solid geometry and scalar fields). Triangle meshes are used as the output as they facilitate the application of text in later phases.

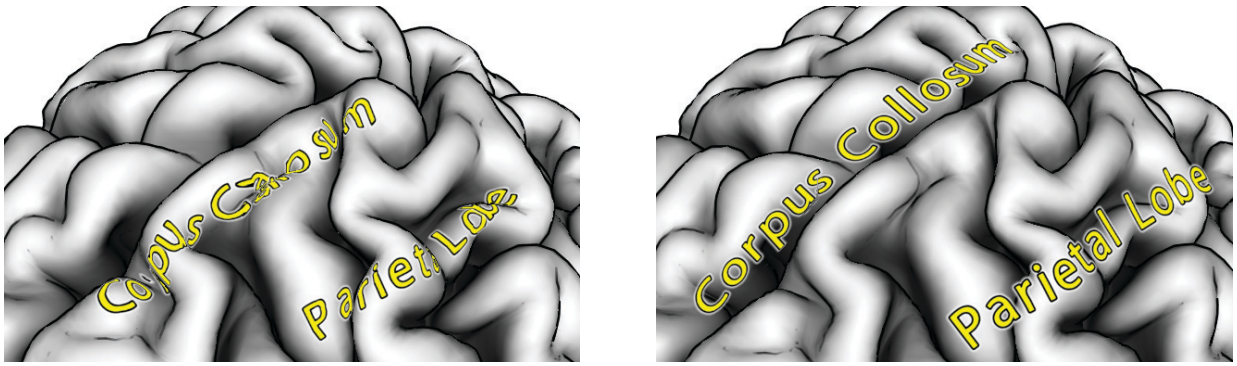
Figure 3 provides an overview of our process. The key idea in the method is to use a signed distance field to represent this surface. This representation simplifies the processing to create smooth, highly-visible surfaces. In particular, it is able to fill holes and make other topological alterations more simply than can be achieved using methods such as [14, 39]. Signed distance fields also facilitate our drastic contraction-proof smoothing. As a final step, the distance field is converted to a triangle mesh.

For our application, the drawbacks of distance fields are not important. In particular, distance fields have limited resolution, and therefore cannot provide perfect reconstructions of the source data. This problem does not affect us as we only seek approximations. However, our implementation uses sub-voxel precision throughout to allow for better representations and smoother surfaces.

3.1 Distance Fields

The first step of our approach converts the source data to a distance field. To do this, we first create an implicit representation of the source data, and then transform it to a distance field.

For data provided as a scalar field, conversion is straight-forward, requiring only resampling to the desired dimensions of a scaffold field. Specific iso-contours can be set as the surface by subtracting that value over the field. Constructive solid geometry is also straightforward to handle. Currently, we allow for union, subtraction, intersection of cylinders and spheres. These operations are achieved by first rasterizing each object into a separate distance field [9].



(a) Brain, with text rendered directly onto the surface.

(b) Our approach, rendering text onto a scaffold.

Fig. 2. Labeling the brain surface: labels placed directly on the surface (a) can be illegible as the surface details cause the text to be distorted and partially occluded. Text scaffolds (b) avoid these issues, while still conveying the overall shape.

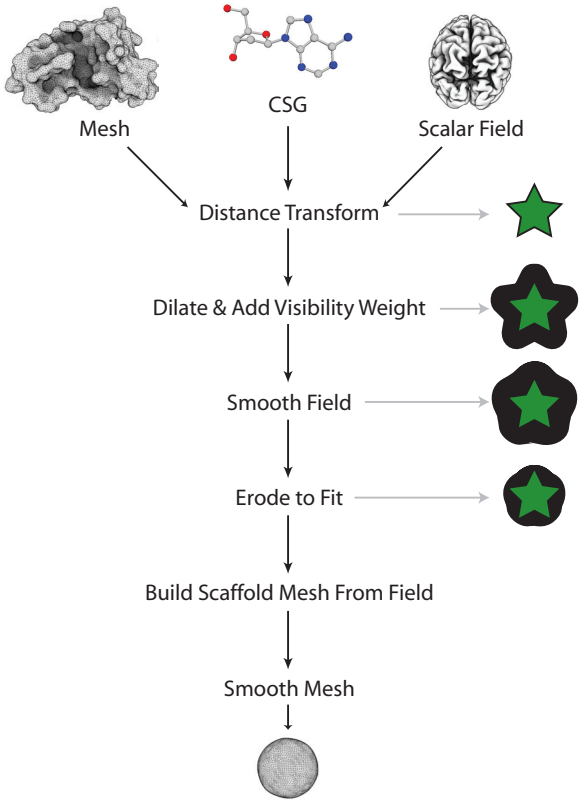


Fig. 3. The text scaffold generation algorithm.

For triangle meshes, a scan-line conversion method similar to [6] is used, where first triangles are scanned in raster order into the field to find intersections. Voxels that are determined to intersect with a triangle are given a value according to the signed distance from the center of that voxel to that triangle. For voxels containing multiple triangles, we take advantage of the fact that scaffolds must bound the surface itself. We also do not require strict fidelity, so so our system can simply find the minimum signed distance over all triangles, and assign that value to the voxel. This avoids costly inside/outside tests, at the expense of having a scaffold boundary that slightly deviates outward from the original surface.

Once the boundary is established, all other voxels are scan converted by casting rays along rows. Voxels are given values as in Eq. 2. Those encountered after passing an odd number of boundary points are given an initial value of -1 to mark them as inside the surface. Remaining voxels are given a value of 1 to mark them as outside. To be robust against meshes that do not form closed two-manifold surfaces, our system casts rays from multiple projections, as in [29], and assigns to each voxel the most probable value given by voting among the results of each projection.

After loading our source surface, the band of voxels closest to the surface are considered to contain a valid distance to the nearest point on that surface.

In our method, we will need to retain the original field as well as the field which will be operated on throughout the text scaffold generation process. These will be designated F_{orig} and F_{cur}, respectively, in the following sections.

3.1.1 Building the Distance Field

Once obtained, the scalar field must be converted to a distance field. In other words, for a given bounded solid S, we require each voxel to contain its distance to the boundary ∂S. We use the following form for the signed distance function [24]:

$$\mathrm{d}(\mathbf{p}) = \mathrm{sgn}(\mathbf{p}) \inf_{\mathbf{x} \in \partial S} \|\mathbf{x} - \mathbf{p}\| \tag{1}$$

where

$$\mathrm{sgn}(\mathbf{p}) = \begin{cases} -1 & \text{if } \mathbf{p} \in S \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

During each pass of the algorithm, the distance field will need to be updated. Our system assumes that the narrow band around the zero isocontour, also known as the set of boundary cells, contains a valid distance value. Remaining cells are then filled in using the Vector-City Vector Distance Transform [33]. In this scheme, a vector is maintained for each voxel which points to the nearest point on the surface. This allows for higher accuracy than Euclidean techniques (such as Chamfer distance [2, 3]), which tend to amplify the discretization errors that arise as distance is propagated away from the surface boundary.

Our method initializes boundary-cell distance vectors to be the gradient vector for each cell, weighted by its field value. This gives a starting set of vectors, which are then propagated outward using a sweeping scheme. This is first done for interior voxels, then repeated for exterior voxels, producing a signed distance transform.

3.2 Scaffold Generation

Once a distance transform is generated, our algorithm constructs a final scaffold by performing the following sequence of steps, described in more detail below. First, F_{orig} is copied into F_{cur}. Then, ambient visibility values are calculated on F_{cur}, which are used to expand the surface to close up regions of high occlusion. F_{cur} may be further dilated at this point, to close up any small topological holes.

F_{cur} is smoothed to remove creases and other artifacts that may have crept in, and then eroded back, contracting as far as possible while still remaining above the original surface. Finally, a scaffold is generated by first performing marching cubes [28] on the field to

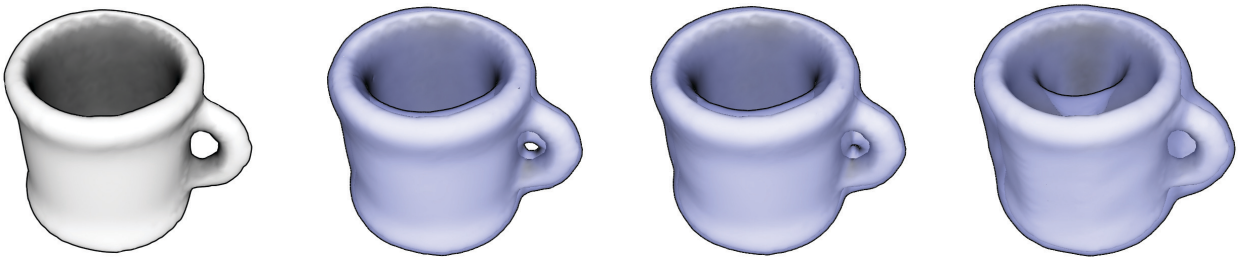


Fig. 4. Holes in the source mesh, such as in the handle of this coffee cup, can be closed by either increasing the dilation constant or by increasing the ambient weight term. Depicted here are the results of increasing, from left to right, the ambient weight term. Note that in the figure on the right, the hole formed by the handle has completely closed up in the scaffold. In this example, $w_d = 0$, while $w_a = 0$, 1.0 and 3.0, from left to right.

produce an initial mesh, and then a final pass of Taubin smoothing [37] is applied to produce a final, smooth mesh.

After every step above, F_{cur} likely no longer represents a valid distance transform. So its distance field is regenerated according to the scheme described in §3.1.1.

Another subtlety is that, especially when dilating, the surface may expand outside of the bounds of the field itself. Our system handles this problem by first computing how far the surface would leave the field during an operation. The boundaries of the field are then expanded (or padded) by adding new cells before actually performing that operation. This ensures that the zero surface is still contained within the field at all times.

3.2.1 Dilate Surface and Weight by Ambient Visibility

Dilation of a surface in a distance transform can be accomplished by simply subtracting the desired expansion constant value w_d from the scalar value in every voxel. This effectively moves the zero-surface of F_{cur} outward to lie on the surface cutting through w_d. After recomputing the distance transform, small pockets and holes may disappear, and components separated by a distance less than w_d will become joined.

Our algorithm further dilates F_{cur} to force it to avoid occlusion. To accomplish this, it first calculates ambient visibility for each voxel by traversing the grid in the direction of each of 26 neighbors. For each direction, if the edge of the field is reached before an interior voxel, then that voxel's ambient visibility value $vis_{cur}(x, y, z)$ is incremented.

Once all ambient values are computed, ambient field values are combined with the dilation constant to produce a dilated surface in the following way:

$$amb(x, y, z) = max\left(0, vis_{thresh} - \frac{vis_{cur}(x, y, z)}{26}\right) \quad (3)$$

$$F_{cur}(x, y, z) = F_{cur}(x, y, z) - w_a * amb(x, y, z) - w_d \quad (4)$$

Here, vis_{thresh} corresponds to the minimum percentage of rays that must intersect the surface to consider the voxel to be occluded. Values for vis higher than this number are not considered to be in a pocket, and so will not contribute to the dilation of the field. In this paper, we set $vis_{thresh} = .5$, which leaves voxels unaffected that are mostly outside the mesh, but dilates those voxels that are inside proportionally to their degree of occlusion.

Weights w_a and w_d can be adjusted to tune the desired degree of occlusion-avoidance and dilation, respectively. Figures 4 and 6 show the effects of different parameters on a surface.

Using ambient occlusion allows our method to directly improve overall text visibility, as surface regions that are not visible will be pushed outward, while the outer boundary remains unaffected. In contrast to dilation, areas that are within significant pockets, but nevertheless far from any part of the surface, are still filled in by utilizing ambient occlusion.

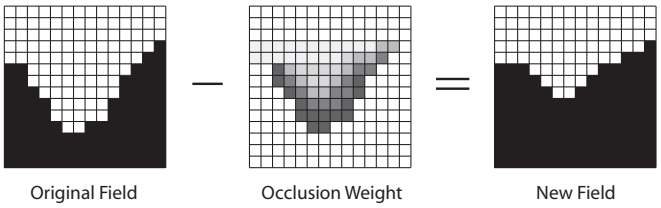


Fig. 5. Shown here is a 2D representation of the results of adding ambient weight to the field surrounding a pocket. At left, the original field, with black depicting voxels interior to the mesh, and white depicting exterior voxels. Occlusion weight, shown at center is calculated for all voxels and subtracted from these values. External voxels close to the surface of the mesh may now be negative, causing them to become internal voxels.

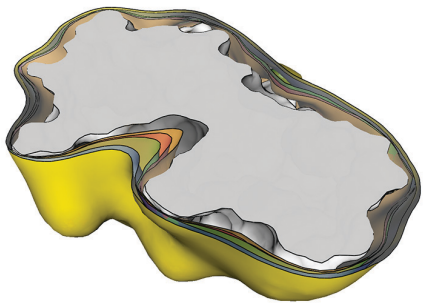


Fig. 6. Shown here is a cutaway view of a protein surface with multiple scaffolds, each with a different visibility weight. Note that the cleft, inside which voxels are highly occluded, is followed most closely by the orange surface, which has no ambient weight applied. Each successive surface is generated by an increasingly larger ambient weight ($w_d = 0$ and $w_a = 0$, 2, 4, 8 and 12, respectively), causing the surface to be pushed out of the cleft.

F_{cur} now no longer represents a valid distance field, so our method now follows the same steps as in §3.1.1 by throwing away all distances except those within the narrow band around the zero-surface and reinitializing the field.

3.2.2 Smooth Field

By dilating F_{cur}, our method has now satisfied our visibility goal, but it still needs to ensure a smooth scaffold on which to place text. Though a separate mesh smoothing pass is applied later in §3.2.4, we desire to have our field as smooth as possible, as we find that this both improves field quality after erosion, as well as the quality of the mesh that we can extract from the field.

In this step, smoothing is a Gaussian filter of size σ_b is convolved with the scalar field values of F_{cur}, which affects them in much the same way that it would affect an ordinary image: values blur with their neighbors, and sharp features tend to diffuse into surrounding areas.

σ_b is another user-controllable parameter, and its effect is to smooth

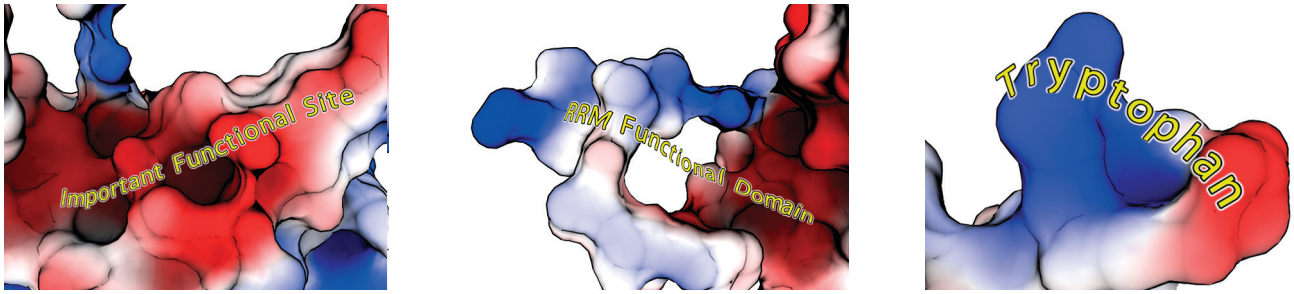


Fig. 7. Problem cases for traditional text rendering: at left, a bumpy surface, with text placed directly over surface noise. At center, text is allowed to flow over the hole. At right, text is allowed to lift off a set of peaks. In these examples, $w_d = 1.0$ and $w_a = 2.0$.

out peaks and fill in valleys. Larger values for σ_b, then, reduce undulation in the scaffold, which contributes to the overall visibility of labels placed on that scaffold, at the expense of less fidelity: the scaffold will diverge from the original surface in high-frequency regions. All figures in this paper are generated with $\sigma_b = .8$.

As is the case after §3.2.1, F_{cur} no longer represents a distance field after smoothing, so again the field is regenerated §3.1.1.

3.2.3 Erode Surface

The surface contained in F_{cur} now avoids pockets, and is smooth, but due to dilation, it no longer tightly bounds the original surface. The next step is to erode F_{cur} back as far as is possible without intersecting the surface contained in F_{orig}.

In order to accomplish this, we want to find the isosurface contained within F_{cur} with the smallest value that is still completely contains F_{orig}. This amounts to searching for the largest field value of F_{cur} over all positive boundary voxels in F_{orig}. This value is subtracted from F_{cur} to produce a final bounding surface, and again the distance field is recomputed.

3.2.4 Create Scaffold Mesh

The final step in creation of a scaffold mesh is to convert the distance transform into a triangular mesh. Our algorithm uses marching cubes [28] for this purpose. Since the field now defines a set of smooth isosurfaces, marching cubes will produce a mesh that is smooth at larger scales, but it still contains minor stair-step artifacts along each voxel boundary.

To remove these artifacts, Taubin smoothing [37] is applied. We find that two iterations suffice, using $\lambda = .8$ and $\mu = .87$, to flatten any residual stair steps. Care should be taken to avoid scaffold intersection with the underlying model, which would cause text to disappear under the surface. In practice, because these vertices are not moving far, we have not found this to be a problem, though the smoothing process can be halted if such a condition occurred.

One small issue with marching cubes is that the method often produces irregular surfaces. While our system could process the mesh to correct this, as in [40], we find that for the purposes of text layout, the irregularity of our scaffold meshes does not cause any issues.

4 Cartographic Labeling

After creating the scaffold surface, the system must place the text on the surface. This task requires first choosing where the labels should be placed, and then drawing the letters on the surface. Our approach divides the task by having a first phase provide a smooth curve that the letters should follow to a second phase that constructs parameterizations along this curve that allow for texture mapping, and a display component that enhances the display of the textures. In keeping with cartographic terminology, We refer to the first task as *labeling* and the latter as *lettering*.

Cartographic labeling considers three types of features: point features, lineal features, and areal features. For each feature type, our approach generates a smooth path on the scaffold surface onto which lettering is applied. In our implementation, features can be either supplied manually, or taken from the data.

In traditional cartographic labeling, the preferred "up" direction of the map suggests that most labels are placed on straight horizontal lines, unless there is good reason to do otherwise [22]. For surface labeling, there is often no "up" direction (as the object may be rotated), or even "straight horizontals." Additionally, there are often other concerns for directionality, such as conveying shape or maximizing visibility in non-flat regions.

4.1 Smooth Paths

Because the scaffold surface was explicitly constructed to have necessary smoothness and visibility properties, labeling methods can place text anywhere on the surface. For this text to be legible, the path that it follows should be smooth. Because the mesh representing the scaffold surfaces may be coarsely and irregularly triangulated, paths limited to the vertices of the mesh are often not smooth enough, leading to lettering that looks irregular and jagged. Therefore, paths are represented as a series of nodes, each lying on a triangle face. Pairs are implicitly connected by a path running along the surface. This representation keeps nodes on the scaffold (necessary as it is this scaffold onto which labels are applied), while providing more flexibility for path placement.

To create a smooth path given a (potentially not-smooth) input path, our approach first adds additional nodes along the path by linear interpolation where necessary to insure sufficient sampling of the path. This path is then smoothed by first smoothing the curve in 3D, unconstrained by the mesh, and then projecting the points back onto the mesh. These smoothing steps are iterated a fixed number of times. While this procedure only approximates smooth curves on the surface, it is sufficient in practice.

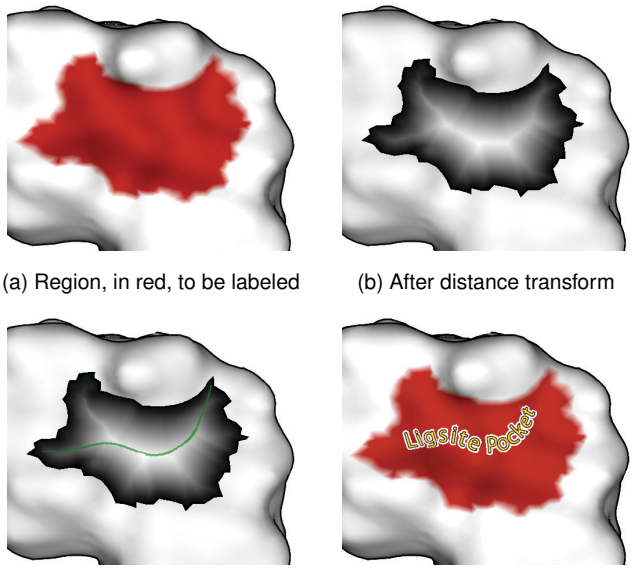
4.2 Point Labels

When the feature to be labeled is a point on the surface, we create a path by fitting a path along the smallest axis of principle curvature. This corresponds to a vector pointing to the flattest local direction of travel from that point. That vector is expanded in both directions. Points are then sampled along the line segment, and projected back onto the surface, where they are iteratively smoothed using the method in §4.1.

4.3 Labeling Lineal Features

Lineal features have two forms: "lines" connecting two endpoints, and strokes, where more detail about a lineal feature is given. Either type may be provided interactively (the former by specifying two vertices, the latter by sketching), or may come from an annotation in the data.

The connection between two points is not necessarily a line because the path must follow the surface. In principle, the straightest path would minimize text curving (which increases readability) and may appear to be the most direct. On a mesh, the straightest path is not necessarily the shortest path [30]. In practice, shortest paths are easier to compute and seem close enough for our application. Our implementation uses the geodesic distance algorithm of Surazhsky, et al., [36] to find the shortest path between the two vertices. This algorithm "cuts across" the faces of triangles, where necessary, to create paths. These paths require no further smoothing.



(a) Region, in red, to be labeled (b) After distance transform

(c) Path fit to longest medial axis (d) Resulting text

Fig. 8. Our automatic labeling algorithm first applies a distance transform to the 2D parameterization of a region. A path is then fit along the longest medial axis of that transform, smoothed, and text placed in the center of that path. The resulting label fits the contours of the region, while avoiding proximity to region boundaries.

For many lineal features, a more complete curve is provided. For example, there might be a lineal feature in the data (such as a ridge) or the user might sketch a curve to indicate a shape to convey with the label. These curves may not provide good paths for labels, as they may not be smooth enough which would lead to unattractive or illegible text. Therefore, our system smooths the provided curve (using the method described §4.1) to create the text paths.

4.4 Labeling Areal Features

Areal features (regions on the surface) are marked in our system as a collection of vertices on the initial mesh. Our system uses a variant of the method of Freeman and Ahn [8] (detailed in Plumer, et al. [7]) to convert an areal region into a path as described below.

4.4.1 Mapping Regions To Scaffolds

The first step in this process is to, for each annotated region, find an associated region on each scaffold. This is done by projecting each point in a region onto the nearest few vertices on the surface. The new scaffold region, then, is the union of these projections. Our system uses Approximate Nearest Neighbors [23] to speed up this process.

After the region is created, it may contain small holes due to sampling artifacts. Their presence can foil the path-creation process, so a "close" morphological operator [10] is applied, which first dilates the region two vertices outward along the mesh, and then erodes by two vertices.

4.4.2 Building The Path

Now that regions have been built on each scaffold, the next step is to construct a path that conforms to the contours of a region. We desire this path to take the form of the underlying region: if the region is long and thin, it should lie along the major axis. If the region is more circular, it should stay roughly in the center. In either case, the path should avoid boundaries whenever possible.

A well-known technique in automatic 2D map-labeling [7, 8] is to construct such a path from the medial-axis transform. The idea is to first construct a distance transform of the region as a 2D texture. Then, a connected graph, or skeleton, is built by finding all maximal points in that transform. This skeleton can be noisy, but the longest path will approximate the "spine" of the region.

We adapt this technique for finding text paths on the surface, as depicted in Figure 8. First, a region is parameterized into a 2D texture. A spine is found, as described above. This spine is then placed back onto the surface, according to the same parameterization, and smoothed using the same method as described in §4.3.

4.5 Applying letters

Regardless of which method is used to construct label paths on the surface, our system parameterizes each character individually. This choice, rather than parameterizing the label as a whole or even the entire surface, allows for flexibility over character placement, and ensures that even long strings of text are unlikely to suffer from a bad parameterization. We have found the Exponential Maps method [34] to be well suited to the creation of small, local character parameterizations.

One downside of this decision, however, is that it becomes harder to choose a good relative orientation for letters, as parameterization of one letter cannot be used to inform the others. Our system orients letters by averaging over all piecewise-linear path segments that are contained within a given letter's parameterization.

As text may be displayed at any size, letters need to be legible at high magnification, with minimal pixelation or other artifacts. Since a surface labeling could contain an arbitrary number of labels, memory footprint was also a concern, so we wanted to keep texture usage to a minimum.

We chose to use Green's alpha-tested magnification [13] method, which uses texture hardware to subsample the distance transform of the shape of a given character. Both character boundaries and stroke boundaries are established at specific isocontours of this distance transform. The primary advantage of this method is that it avoids the characteristic step-blurring effect that results from bilinear filtering of a hard boundary edge, and produces sharp boundaries from relatively small textures.

All characters are packed into one texture, which is 1024x1024 pixels in dimension. When rendering a character, our system adjusts the parameterization to sample the correct character from this texture. Screen space anti-aliasing, if available in hardware, is used to further smooth character edges.

5 Results

We have implemented our text scaffolding techniques in a visualization testbed that runs under Windows. For all molecular examples in this paper, we use MSMS [32] to generate an initial molecular surface as a triangle mesh, and some regions are identified using an implementation of Ligsite [21] to identify putative binding pockets.

Figures throughout the paper show the effectiveness of our approach in legibly conveying annotations on a complex surface. Figure 7 shows several problem cases for traditional surface text rendering, and how our solution handles them. Figure 9 shows how our system can automatically construct a path to place an label for an annotated input region. Figures 10 and 11 show more examples of how our system handles various types of input surfaces.

In all figures, the original model was inserted into 40x40x40 voxel field. We found that a field of this size results in good labels for a wide range of model sizes. Larger fields, in general, remain more faithful to the original surface, but at the cost of increased runtimes. All figures also used values of -1 for w_d, and varying values for w_a. In our experience, higher values for w_d produce a smoother result, as isocontours tend to smooth out farther from the surface. This is not always desirable, as a smoother surface is necessarily a less-faithful surface, and further, its results are often not as intuitive as with other parameters. For this reason, we chose to fix this value.

We have found our scaffold-generation algorithm to take time roughly linear in the number of voxels in the distance field. Scan-line rasterization of a triangle mesh is also linear in the number of voxels and the number of triangles. In our un-optimized implementation, all figures in this paper took less than a five seconds to finish this step,

most less than two. In practice, this is efficient enough that manual specification of the small number of parameters is feasible. However, our approach is not currently suitable for dynamic surface adjustment, such as changing transfer functions in a volumetric model.

Text path planning and smoothing also take a few seconds, though the geodesic algorithm may take as long as ten seconds to find the shortest path between two vertices that lie a long distance from one another along the surface. We implement the exact form of Hoppe's geodesics algorithm [36]; the approximate form would reduce this time significantly.

All performance testing was done on a HP dv2500t laptop, with a Intel T7100 CPU and Nvidia 8400M graphics processor. Text rendering is implemented using pixel shaders in only a few dozen instructions, so on this machine, our implementation can render several hundred labels with no perceptible loss of performance over rendering without the labels.

6 Discussion

Our initial prototype shows that Text Scaffolds can provide for labels that give shape cues for curved surfaces, yet remain legible despite surface features.

Labeling by its nature involves tradeoffs. While the text scaffold approach provides some control over these tradeoffs through its parameters, other tradeoffs have been made explicitly in its design. For example, we have chosen to fix the labels on the surface rather than have them move as the viewpoint changes. This means that text may be presented at undesirable orientations, but also means it is better able to serve as a shape cue and positional landmark. The "correct" solution to these tradeoffs depends on the viewer's perception, their subjective preference, and the needs of the particular data. Therefore, true assessment of the effectiveness of the approach, in particular to guide the choice of when it is appropriate, will require user evaluation.

To date, we have not studied user reactions to text scaffold labeling. Anecdotally, domain collaborators (particularly structural biologists), have shown interest in better surface labeling and have reacted positively to our prototypes. However, user studies are required to confirm that Text Scaffolds provide for effective labeling. In particular, studies can provide better understanding of the readability of text on curved surfaces and how well it conveys shape. A better understanding of how proximity fosters mental connection will be useful not only in tuning our methods, but also in understanding where the method is more appropriate than disembodied billboards or direct surface labeling.

At present we have not explored automating the process of tuning the algorithm to control tradeoffs. In principle, smaller text and smaller features may suggest scaffolds that are closer approximations to the data. We find that tuning w_a to produce text that either avoids or follows pockets seems to allow for enough flexibility to label a wide variety of scenarios, including all depicted in this paper.

Our current implementation contains limited methods for automatic label placement on the surfaces. Future work should extend this to better place labels in effective ways. Such an extension will require more consideration of how the rules of cartography extend to surfaces in 3D, as well as the development of algorithms that create labelings to follow these rules. In particular, our current approach considers each label separately. A more global approach could prevent overlapping labels, control label density, and provide consistency between labels. Cartographic aesthetic principles, such as constant-curvature areal labels could also be translated to 3D surfaces. We have not yet considered issues in the lettering itself, such as choosing text sizes, breaking long labels into multiple lines, or how to handle multi-line labels.

Our implementation strategy also introduces potential limitations. Because we divide the process into three independent steps, scaffold generation, path formation, and lettering, the steps have limited opportunity to influence each other. For example, the needs of a particular path might suggest places where surface should be altered, or the placement of letters may place additional restrictions on the smoothness of the path. Similarly, our consideration of letter positions in parametric space makes font hinting or precise control of kerning in 3D difficult.

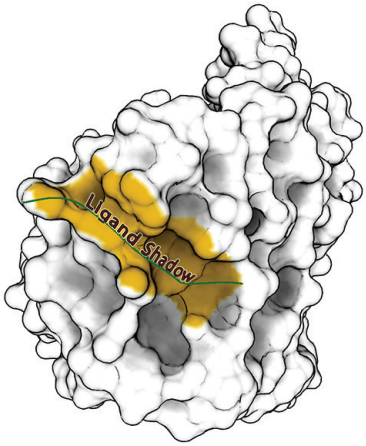


Fig. 9. A ligand shadow, with an automatically generated label. Its path, depicted here in green, is constructed by following the medial axis of the underlying region.

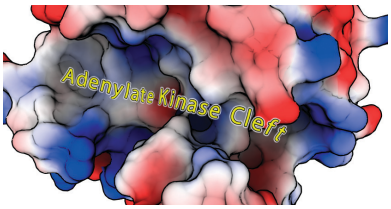


Fig. 10. An adenylate kinase cleft, with text avoiding regions of high occlusion. Here $w_d = 1.0$, while $w_a = 5.0$.

Despite these limitations, text scaffolding does provide an alternative to existing labeling techniques for providing annotations on 3D models. The approach uniquely provides legibility, visibility and proximity to the feature, while conveying of shape cues across a variety of data sources and model types. The approach affords an efficient implementation. The results of an initial prototype show that the text scaffold provides a compelling method for surface labeling that complements existing approaches.

Acknowledgements

We thank George Phillips Jr. and his lab for their feedback in this project. Cipriano was supported by NIH training grant NLM-5T15LM007359.

References

[1] K. Ali, K. Hartmann, and T. Strothotte. Label layout for interactive 3d illustrations. *WSCG (Journal Papers)*, pages 1–8, 2005.

[2] G. Borgefors. Distance transformations in digital images. *Comput. Vision Graph. Image Process.*, 34(3):344–371, 1986.

[3] G. Borgefors. On digital distance transforms in three dimensions. *Comput. Vis. Image Underst.*, 64(3):368–376, 1996.

[4] J. Christensen, J. Marks, and S. Shieber. An empirical study of algorithms for point-feature label placement. pages 286–309, 1998.

[5] B. G. Cumming, E. B. Johnston, and A. J. Parker. Effects of different texture cues on curved surfaces viewed stereoscopically. *Vision research*, 33(5-6):827–838, 1993.

[6] F. Dachille and A. Kaufman. Incremental triangle voxelization. In *Graphics Interface 2000*, pages 205–212, May 2000.

[7] D. Dörschlag, I. Petzold, and L. Plümer. Placing objects automatically in areas of maps. *In Proc. 23rd Internat. Cartographic Conf. (ICC'03)*, pages 269–275, 2003.

[8] H. Freeman and J. Ahn. On the problem of placing names in a geographic map. *Int. J. Pattern Recogn. Artif. Intell.*, 1(1):121–140, 1987.

[9] S. F. Frisken and R. N. Perry. Designing with distance fields. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, pages 60–66, New York, NY, USA, 2006. ACM.

(a) An armadillo (b) A chair (c) A porine molecule

Fig. 11. A set of example surfaces. In (a), labels track the contours of the armadillo's limbs. In (b), labels are allowed to float over empty space in the chair back. In (c), an annotating label is nestled within the protein's channel. In (a) and (c), $w_a = 1$, in (b), $w_a = 10$. In all images, $w_d = 2$.

[10] R. C. Gonzalez and R. E. Woods. *Digital image processing*. Prentice-Hall, 2002.

[11] G. Gorla, V. Interrante, and G. Sapiro. Texture synthesis for 3d shape representation. *IEEE Transactions on Visualization and Computer Graphics*, 9(4):512–524, 2003.

[12] T. Götzelmann, K. Hartmann, and T. Strothotte. Agent-based annotation of interactive 3d visualizations. In A. Butz, B. Fisher, A. Krüger, and P. Olivier, editors, *6th International Symposium on Smart Graphics 2006, LNCS 4073*, pages 24–35. Springer Verlag, 2006.

[13] C. Green. Improved alpha-tested magnification for vector textures and special effects. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, pages 9–18, New York, NY, USA, 2007. ACM.

[14] I. Guskov and Z. J. Wood. Topological noise removal. In *GRIN'01: No description on Graphics interface 2001*, pages 19–26, Toronto, Ont., Canada, Canada, 2001. Canadian Information Processing Society.

[15] T. Gtzelmann, M. Gtze, K. Ali, K. Hartmann, and T. Strothotte. Annotating Images through Adaptation: An Integrated Text Authoring and Illustration Framework. *Journal of the WSCG*, 15(1), 2007.

[16] T. Gtzelmann, K. Hartmann, and T. Strothotte. Annotation of Animated 3D Objects. In *Simulation and Visualization*, 2007.

[17] M. Harrower. Tips for designing effective animated maps. *Cartographic Perspectives*, 44:63–65, 2003.

[18] M. Harrower and B. Sheesley. Moving beyond novelty: creating effective 3-d fly-over maps. In *XXII International Cartographic Conference*, pages 40–47, 2005.

[19] M. Harrower and B. Sheesley. Utterly lost: Methods for reducing disorientation in 3-d fly-over maps. *Cartography and Geographic Information Science*, 34(1):19–29, 2007.

[20] K. Hartmann, T. Götzelmann, K. All, and T. Strothotte. Metrics for functional and aesthetic label layouts. *Smart Graphics: 5th International Symposium, SG 2005, Frauenwörth Cloister, Germany, August 22-24, 2005: Proceedings*, 2005.

[21] M. Hendlich, F. Rippmann, and G. Barnickel. LIGSITE: automatic and efficient detection of potential small molecule-binding sites in proteins. *J. Molecular Graphics and Modelling*, 15(6):359–363, 1997.

[22] E. Imhof. Positioning names on maps. *The American Cartographer*, 2(2):128–144, 1975.

[23] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proc. of 30th STOC*, pages 604–613, 1998.

[24] M. Jones, J. Baerentzen, and M. Sramek. 3d distance fields: a survey of techniques and applications. *Visualization and Computer Graphics, IEEE Transactions on*, 12(4):581–599, July-Aug. 2006.

[25] K. G. Kakoulis and I. G. Tollis. A unified approach to labeling graphical features. In *SCG '98: Proceedings of the fourteenth annual symposium on Computational geometry*, pages 347–356, New York, NY, USA, 1998. ACM.

[26] D. C. Knill. Contour into texture: information content of surface contours and texture flow. *Journal of the Optical Society of America A*, 18:12–35, Jan. 2001.

[27] W. Li, L. Ritter, M. Agrawala, B. Curless, and D. Salesin. Interactive cutaway illustrations of complex 3d models. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 31, New York, NY, USA, 2007. ACM.

[28] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, 1987.

[29] F. Nooruddin and G. Turk. Simplification and repair of polygonal models using volumetric techniques. *Transactions on Visualization and Computer Graphics*, 9(2):191–205, 2003.

[30] K. Polthier and M. Schmies. Straightest geodesics on polyhedral surfaces. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, pages 30–38, New York, NY, USA, 2006. ACM.

[31] T. Ropinski, J.-S. Prani, J. Roters, and K. H. Hinrichs. Internal labels as shape cues for medical illustration. In *Proceedings of the 12th International Fall Workshop on Vision, Modeling, and Visualization (VMV07)*, pages 203–212, nov 2007.

[32] M. F. Sanner, A. J. Olson, and J.-C. Spehner. Fast and robust computation of molecular surfaces. In *SCG '95: Proceedings of the eleventh annual symposium on Computational geometry*, pages 406–407, 1995.

[33] R. Satherley and M. W. Jones. Vector-city vector distance transform. *Computer Vision and Image Understanding*, 82:238–254, 2001.

[34] R. Schmidt, C. Grimm, and B. Wyvill. Interactive decal compositing with discrete exponential maps. *ACM Transactions on Graphics*, 25(3):603–613, 2006.

[35] T. A. Slocum, R. B. McMaster, F. C. Kessler, and H. H. Howard. *Thematic Cartography and Geographic Visualization*. Prentice Hall, 2005.

[36] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. J. Gortler, and H. Hoppe. Fast exact and approximate geodesics on meshes. *ACM Trans. Graph.*, 24(3):553–560, 2005.

[37] G. Taubin. A signal processing approach to fair surface design. In *Proceedings of SIGGRAPH 95*, pages 351–358, Aug. 1995.

[38] S. van Dijk, M. van Kreveld, T. Strijk, and A. Wolff. Towards an evaluation of quality for names placement methods. *International Journal of Geographical Information Systems*, pages 641–661, 2002.

[39] Z. Wood, H. Hoppe, M. Desbrun, and P. Schröder. Removing excess topology from isosurfaces. *ACM Trans. Graph.*, 23(2):190–208, 2004.

[40] Z. J. Wood, P. Schröder, D. Breen, and M. Desbrun. Semi-regular mesh extraction from volumes. In *VIS '00: Proceedings of the conference on Visualization '00*, pages 275–282, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.