IStar: A Raster Representation for Scalable Image and Volume Data

Joe Kniss, Warren Hunt, Kristin Potter, Pradeep Sen

Abstract— Topology has been an important tool for analyzing scalar data and flow fields in visualization. In this work, we analyze the topology of multivariate image and volume data sets with discontinuities in order to create an efficient, raster-based representation we call *IStar*. Specifically, the topology information is used to create a dual structure that contains nodes and connectivity information for every segmentable region in the original data set. This graph structure, along with a sampled representation of the segmented data set, is embedded into a standard raster image which can then be substantially downsampled and compressed. During rendering, the raster image is upsampled and the dual graph is used to reconstruct the original function. Unlike traditional raster approaches, our representation can preserve sharp discontinuities at any level of magnification, much like scalable vector graphics. However, because our representation is raster-based, it is well suited to the real-time rendering pipeline. We demonstrate this by reconstructing our data sets on graphics hardware at real-time rates.

Index Terms—Topology, Compression, Image Representation.

1 Introduction

In visualization and computer graphics, we are often interested in rendering multivariate functions parametrized over 2D (e.g. RGB images) or 3D (e.g. multimodal volume data) that contain important discontinuities which represent the boundaries between different materials (e.g. skin and bone, gray matter and white matter, etc.). These discontinuities can be preserved during rendering by representing them with implicit compositions of simpler functions. In 2D, for example, we can do this with vector graphics (e.g. Postscript [25]) where the function is represented by mathematical primitives such as circles, lines, and polynomial curves. In 3D, this process is performed by a standard 3D renderer, which generates images from an implicit composition of mathematically-defined primitives such as spheres, polygons, etc. These approaches allow for scale-invariant reconstruction of the original function while preserving important discontinuities. Because these representations encode individual primitives, they also provide a mechanism for attaching semantic meaning to the different primitives which is useful for visualization (e.g. the ability to shade a specific kind of object, such as bone, with a new color or to remove it altogether from the rendering process).

However, these geometry-based formats have several serious drawbacks. For one, they are too slow for rendering complex data sets at real-time rates. The inherent problem is that their rendering complexity grows linearly with the number of primitives and is potentially unbounded. This linear dependence makes it difficult for rendering systems to keep up with the real-time rates, especially when these images are used to texture map surfaces in a scene. In addition, it is not clear how to apply an arbitrary filter kernel to these implicit representations, like when pre-conditioning the signal for antialiasing.

Another way to represent these functions is with a uniformly sampled representation, such as raster or bitmapped images for 2D data sets and 3D textures for volume data sets. The advantage of raster images is that they require a known, constant time to evaluate the function at any point because it can be done with a fixed number of accesses into an array (enough to support the reconstruction kernel) plus some bounded computation to execute the kernel itself. This explains the popularity of using 2D bitmaps for texture mapping surfaces in interactive applications. In addition, operations such as convolution with a Gaussian kernel are well defined on raster data sets, which makes antialiasing raster images relatively simple. However, a significant drawback is that raster images must be band-limited to eliminate aliasing because they are, by definition, sampled representations. This means that important semantic information present in our original function, such as the sharp discontinuities between different materials, can be lost during the sampling process.

In this paper, we propose a new framework for representing functions with discontinuities efficiently in a raster format which we call *IStar*. This representation shares many of the advantages of both vector and raster formats. First, our approach allows discontinuities to be reconstructed precisely, independent of scale as can be done with vector formats. Second, the rendering algorithm is simple and its constant-time complexity is independent of the number of primitives in the IStar image. This allows us to render IStar images on graphics hardware and use them to texture map scenes at real-time rates. Third, IStar's raster representation can be compressed using conventional techniques, resulting in a data structure that is more space-efficient than traditional bitmaps yet can still reconstruct the sharp discontinuities of the original image. Fourth, because our structure preserves the semantic meaning of regions in the image, we can easily modify their characteristics during rendering. Finally, since our structure encodes information on the topology of the data, we can use it enforce constraints or fix classification errors, which is useful for visualization applications.

We begin the paper with an overview of the 2D version of our algorithm in Section 2 to give the reader with an intuitive understanding of our approach. In Section 3, we introduce a topology framework that we will use to describe the algorithm more formally in Section 4. In Section 5, we discuss details important for implementation of the technique, and compare it to previous work in Section 6. Section 7 presents our results and we conclude in Section 8 with future work.

2 Overview of Algorithm

The IStar encoding process takes as input an image function with regions of constant color, which we call the *primal image*. The top row of Figure 1 shows an example of the encode process with an original 2D image (a) and the primal image with uniquely defined regions (b). This primal image can be used to generate a graph-like structure, called the *dual complex*, whose nodes represent the unique regions and edges represent the boundaries between regions, as seen in (c). The next step is to embed the dual complex in a color space, (d), and relabel (recolor) the primal image based on the coordinates of the node positions in this new space, (e). Note that disjoint regions with the same color in the primal image are mapped to different nodes in the dual complex and are therefore embedded in different positions of the color space, giving the relabeled image a distinct color for every

- *Joe Kniss, Warren Hunt and Pradeep Sen are with the Advanced Graphics Lab at the University of New Mexico, E-mail: jmk@cs.unm.edu, whunt@cs.unm.edu, psen@ece.unm.edu.*
- *Kristin Potter is with the School of Computing at the University of Utah, E-mail: kpotter@cs.utah.edu.*

Manuscript received 31 March 2007; accepted 1 August 2007; posted online 27 October 2007. Published 14 September 2007.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

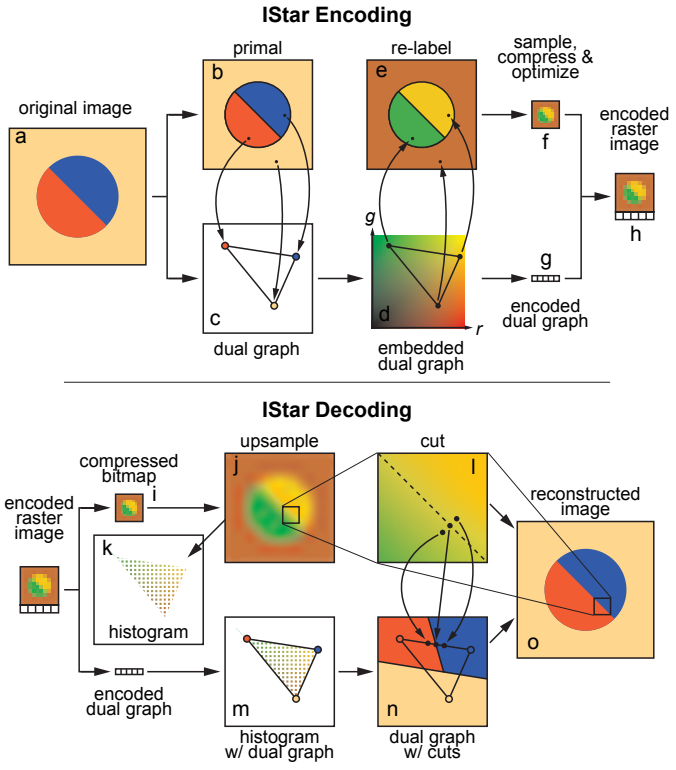

Fig. 1. Overview of IStar encoding/decoding described in Section 2.

unique region. This labeling helps overcome artifacts introduced during the compression process and allows regions to be uniquely identified during decoding. The relabeled image function is then sampled and compressed (e.g. through downsampling), resulting in a miniature bitmap representation, (f). At this point, the colors of the downsampled bitmap can be tweaked to improve reconstruction quality, an optimization which is discussed in detail in Section 5. The node and connectivity information are encoded by storing the coordinates of each node and a list of edges, (g), and because the original color is lost during relabeling, a color palette that associates each node with its color in the original image is generated. The downsampled bitmap, the graph structure and the color palette forms the complete IStar structure, (h).

The decode process, Figure 1 bottom row, takes as input the compressed bitmap, (i), and upsamples it using a reconstruction filter to create a blurry version of the relabeled image, (j). Note, the histogram of the upsampled image follows the structure of the dual complex, as seen in (k) and (m), because of the way the positions of the nodes were used to determine the color of the relabeled image in encoding. Sample values along the boundary of two regions are a linear combination of the colors of the regions and map to the line segment that connects the two nodes of each region, as demonstrated by the three samples in (l) mapping to the line in (n). In the case where three regions share a common boundary at a point, upsampling results in sample values between all three nodes, thereby mapping to points inside the triangle formed by the nodes in histogram space. Using the positions of samples within the dual complex, the upsampled image is "cut" into unique regions, and combined with the original color palette to recolor the image and produce the final, reconstructed image, (o).

3 Comparison to Previous Work

Topology is a field of mathematics that deals with abstract manifolds and their characteristics, such as interrelationships, canonical topological forms, algebraic operations, and mathematical definitions of shape. A solid introduction and reference for basic foundations of algebraic topology can be found in [7, 18]. The application of topology to general scientific problems is surveyed by Dey *et al.* [2]. In general, topological analysis focuses on the characteristics of a single manifold in isolation. In contrast, our work analyses the characteristics and embedability of multiple intersecting manifolds and their duals. A result of this work is the development of an image function that maps from the primal complex to its embedded dual.

Digital topology refers to image processing methods that apply the concepts of mathematical topology to the processing and understanding of digital images [12, 13] by looking at the shape induced by binary images in the integer domain $\mathbb{Z}$. The adjacency graph captures the topology related to manifold intersections and is often use in vision and object matching applications [17]. In contrast, our work develops image topology for continuous image and data domains, and rather than using an adjacency graph, uses a full dual CW-complex for adjacency information, which is essential for reparametrizing the image.

The process of identifying salient topological features in gray-scale (scalar) image data has generally relied on several techniques: skeletonization, Morse-Smale complex identification and simplification, and Reeb graph identification [5, 20]. Though the work of [1] does not explicitly develop a topological framework, it does apply a barycentric interpolation over simplicial complexes to extract geometry from a volume faction data. Topology has also played an important role in the analysis of vector flow fields[8, 22]. The survey on computational topology for shape modeling by Hart [6] covers many common and useful applications of topology in computer graphics. Our work extends scalar image topology methods by developing a framework for multi-variate image topology and techniques for creating images (encodings) that preserve topological characteristics by construction.

Most approaches to represent functions with discontinuities for rendering are based on implicit representations that composite primitives with known mathematical properties. However, the representation of discontinuities in raster-based approaches have recently become an active research. Sen *et al.* [24] provide an excellent survey of current approaches. The methods of Sen [23] and Tumblin and Choudhury [27] are examples of methods that explicitly handle boundary representations as geometric primitives tied to a pixel representation that render in real-time on graphics hardware. However, unlike our approach, they simplify curved features to be piece-wise linear which can produce artifacts when not properly sampled. The methods of Ray *et al.* [21] and Loviscach and Bremen [16] utilize an implicit representation of surface discontinuities, which are effectively level-sets of smooth scalar functions. The work of Ray *et al.* captures cusp discontinuities as the intersection of multiple boundary functions. Loop and Blinn [15] describe a framework for evaluating bounded regions represented as Bezier splines, which is well suited for hardware accelerated raster graphics applications.

4 Topological Framework

This section covers the mathematical foundations of this work. First we define the CW-Complex, which forms the basis of our representation. Next, we present a new topological description of images based on intersecting regions. We use this definition to analyze and extract the topology of an image's data-space, or histogram domain. This information can then be used to reparametrize the image so that unique pixel values identify unique regions, even when the image is been band-limited. The process of reparametrization and rendering are covered in Section 5.

4.1 The CW-Complex

A CW-complex is a fundamental construct from topology. The "CW" stands for *closure-finite weak topology*. CW-complexes are like generalized graphs with nodes (termed 0-cells) and edges (1-cells), however they can also contain surfaces (2-cells), volumes (3-cells) and so on. Each N-cell in the complex must be completely bounded by $(N-1)$-cells and homeomorphic to an N-ball (the interior of an $N-1$ sphere). For instance, each 2-cell is homeomorphic to a disk and bounded by 1-cells, each 1-cell is bounded by 0-cells, etc. The boundary of a 0-cell is simply 0. A CW-complex χ can be deconstructed as set of N-cell sets, or a set of skeletons, $\chi = \{\chi^0, \chi^1, \chi^2, \dots\}$, where the superscript indicates the dimension of cells in the set. We call χ^0 the 0-skeleton

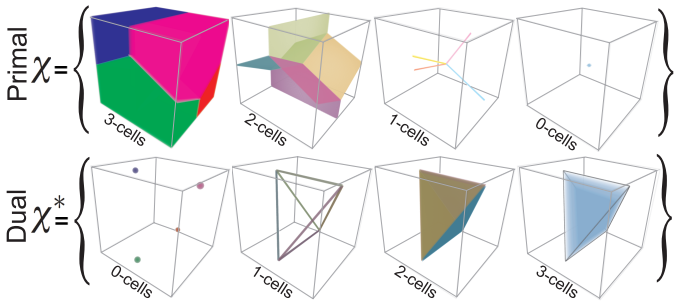

Fig. 2. The relationship between the primal complex and its dual for a 3D image with four "objects." (top) primal complex composed of the 3, 2, 1, and 0-skeletons. (bottom) the corresponding dual complex.

of χ, or the set of all points in the complex. A subscript will indicate a particular element of the skeleton. CW-complexes are useful because they capture both the concrete geometric and abstract algebraic/combinatoric characteristics of manifolds. For a more complete description of CW-complexes and their properties, see Hatcher [7] Appendix, page 519.

4.2 The Primal CW-Complex

Assume for a moment that objects in the real-world are spatially distinct, i.e. they do not overlap except at infinitesimally thin boundaries. Objects are things that are semantically different like air, skin, soft-tissue, and bone. A region of vacuum would still be considered an object since it is semantically distinct from the others. Then, for some finite volume of space $D \subset \mathbb{R}^d$, we say that the N objects A_i that occupy this space fill the entire volume: $D = \bigcup_{i=1}^{N} A_i$. Formally, we state that the union of objects forms a closed cover of the compact set D.

From these objects, we can construct a CW-complex χ which we call the *primal complex*. Figure 2 (top row) illustrates the primal complex for a 3D image. For a d-dimensional image, the primal complex is defined as $\chi = \{\chi^q \,|\, q = 0, \ldots, d\}$, where

$$\chi^q = \left\{ \bigcap_{j=1}^{d+1-q} A_{i_j} \neq \varnothing \right\}. \tag{1}$$

Equation 1 indicates that lower-order q-skeleton elements are formed by object intersections. For 3D volume data, 3-skeleton elements are the objects themselves, 2-skeleton elements are formed by the intersection of two objects, 1-skeleton elements by the intersection of three objects, and 0-skeleton elements by the intersection of four objects.

4.3 The Dual CW-Complex

Given an image of N primal objects, assume we have an algorithm that automatically segments the image and produces N binary images, each of which identifies a unique object with a value of 1 if the object is present at a pixel, 0 otherwise. We can stack these images together to form a single image with an N dimensional vector at each pixel. We call this vector the "indicator vector" since it indicates which object is present at each pixel. We can treat the discrete image as a continuous function, $I_v : D \rightarrow \mathbb{R}^N$, by convolving with an interpolation kernel k:

$$I_v(\vec{x} \in D) = \int_D k(\vec{x} - \vec{s}) \sum_{i=1}^{N} \delta(\vec{s} - \vec{p}_i) V_i \, d\vec{s}, \tag{2}$$

where the V_i is the vector at pixel $\vec{p}_i$, and δ is Dirac delta function.

What structure do the image values have with respect to $\mathbb{R}^N$, the data domain? To answer this question, consider what happens to the interior of a segmented object A_i. In the data domain, points that are not near the object's boundary map to a single point in $\mathbb{R}^N$, i.e. they all have the same value because the samples within the support of the kernel are all equal. Points near the boundary between two objects, however, take on values that are linear combinations of the values of the two objects, due to the blurring effect of convolution-based interpolation. Therefore, these values map somewhere on the line connecting

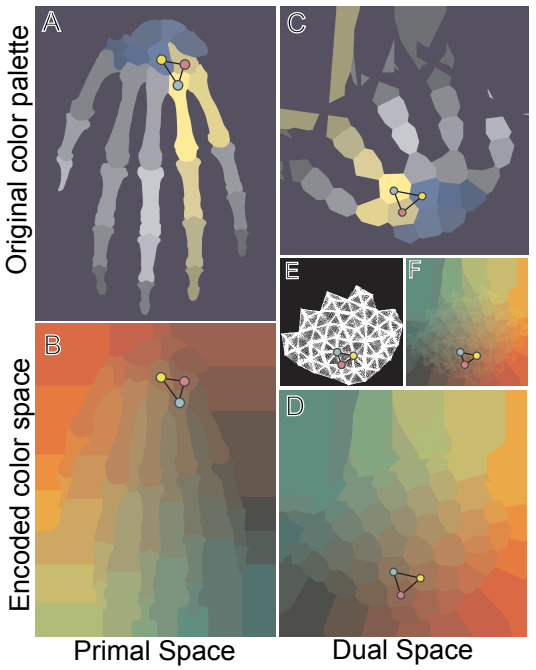

Fig. 3. A hand dataset comparing image structure in primal and dual spaces. The left column shows the image in the primal domain. The right column shows the image in the dual space. [A] shows the hand with an arbitrary color palette. [B] shows the encoded image. [C] shows the "cut regions" in the dual space with the orignal palette and [D] shows the cut regions with the encoded color palette. [E] shows the histogram of the encoded image, which follows the dual-complex structure. [F] shows the histogram overlayed with the dual-domain cut regions. Source: Grays Anatomy.

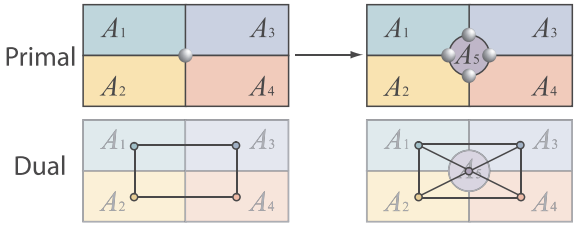

Fig. 4. Blowing up a non-generic boundary in 2D in order to embed it.

the values of the objects in the value domain. Points near the region formed by the intersection of three objects will have values that are linear combinations of the three object values, therefore mapping to the triangle with vertices at the values for each object. Similarly, locations near the region formed by the intersection of four objects will have values located within a tetrahedron in the value domain. This structure (points, lines, triangles, tetrahedrons) is the *dual complex*, χ^*, of the primal, and is defined as $\chi^* = \{\chi^{*q} \,|\, q = 0, 1, 2, 3\}$ where $\chi^{*q} = \chi^{3-q}$. In other words, 3D cells in the primal skeleton become points in the dual, 2D faces become lines, 1D lines become 2D faces, and points become 3D cells. Figure 2 (bottom row) shows χ^* embedded in 3D. We can now see how the dual graph structure described in Section 2 is the dual complex of our primal image. Because our structure encodes the dual of an image I, we refer to it as I^* or IStar. Figure 3 illustrates the image structre in both the primal and dual spaces. A small portion of the dual complex is shown in each image for reference. Notice that the histogram in Figure 3[E] shows the structure of the embedded dual complex.

4.4 Embeddings of χ and χ^*

By definition, the primal complex χ embeds in D, $\chi \hookrightarrow D$. A complex embeds in a space if the uniqueness of *open sets* (elements of the skeleton minus their boundary) is preserved. For example, any point in D is covered by a unique object A_i or is shared by multiple A_i's only

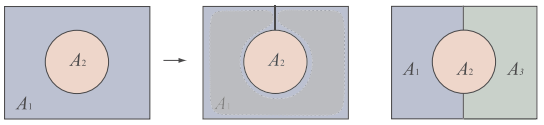

Fig. 5. Making a region simply connected. The center image shows the object intersecting with itself, region A_1 is contracted in grey to show that it is simply connected. Regions can also be made simply connected by dividing them into 2 regions, seen on the right.

if it is in the boundary formed by them, as described by Equation 1. χ^* embeds in $\mathbb{R}^N$ when each of the N nodes are set on unique axes of the space. In fact, χ^* embeds in an $N-1$ dimensional subset of $\mathbb{R}^N$.

While χ^* embeds trivially in $\mathbb{R}^N$, it can be shown that with a few additional constraints, χ^* is embeddable in a compact subset of $\mathbb{R}^d$, where d is the dimension of the image [9]. The constraints are:

(i) either $\bigcap_{j=1}^{m} A_{i_j} = \varnothing$ or $\operatorname{codim}\left(\bigcap_{j=1}^{m} A_{i_j}\right) = m-1$

(ii) A_i is simply connected

(iii) A_i can intersect itself, but only along its boundary

Condition (i) guarantees that the subjects will only intersect in stable, or generic, configurations. For a 2D image, the generic intersection types are 2-way, where two objects meet at a line, and 3-way, where three objects meet at a point. For a 3D image, the generic intersections are 2-way, where two objects meet at a plane, 3-way, where three objects meet at a line, and 4-way, where four objects meet at a point. Generic intersections are those that cannot be destroyed when a subject is moved some infinitesimal amount. Non-generic intersections, greater than N+1-way, can be fixed by "blowing-up" the intersection, which introduces a new subject as seen in Figure 4. Condition (ii) requires each subject to be strictly simply connected, i.e. disconnected regions and holes are not allowed. When a subject is composed of multiple disconnected regions, we can make each simply-connected region a separate subject. Condition (iii) allows subjects with holes to be fixed by linking disconnected boundary segments, as seen in Figure 5. Combined, (i-iii) are sufficient conditions for defining a topological CW-complex which uniquely represents the subjects and their boundaries and its dual. Later in this paper, we will discuss how constraints (ii) and (iii) may not always be necessary when we embed χ^* in a higher dimensional space.

5 Algorithm

In this section we describe the algorithms for embedding and decoding of IStar images. Together, the embedded dual complex and reparametrized values form an encoding of the primal complex that preserves the uniqueness of objects and their boundaries. Discontinuous boundaries are implicitly represented in the encoded image and uniquely identified by their parametrized value. We will use this fact to reconstruct the original image with its discontinuities at any resolution from a raster image. We will also show that the encoded image size can be reduced substantially without significantly impacting the quality of the reconstruction.

5.1 Encoding

The input to the IStar encoding algorithm is an image function, I_v, which can be obtained from a data set, for example an image in 2D or a volume in 3D. For 2D data sets, I_v is a high resolution sampled image of at least 2000×2000 pixels. While it is possible to use geometric representations as input, it is easier to work with high resolution raster images. Note that this does not compromise generality since vector representations can be always be rasterized. For 3D data sets, we use a geometric representation and simply evaluate the indicator vector at any point in the spatial domain. To ensure dual embedability, disconnected objects in the original image may need to subdivided into individual objects, which can be done by re-tagging the image using a flood fill. For ease of explanation, the discussions in the paper are restricted to 2D and 3D, regularly sampled data sets; however, the algorithm and math are general with respect to dimension, and can be used with irregularly sampled data.

The following steps are needed to encode:

1) **Identify and refine primal complex.** Make each connected component of each object an independent object, and create "ghost" 0-cell elements for all n-way intersections greater than $d+1$ where d is the dimension of image.

2) **Identify dual complex.** Each object becomes a 0-cell, any pair of objects co-located within a differential volume become a 1-cell, and so on.

3) **Embed dual in a compact space.** Use a graph layout solver to embed all 0-cell positions in the data space.

4) **Initialize encoded image.** For each pixel in the encoded image, assign its value to the embedded dual 0-cell position associated with the object at that spatial location in the original image.

5) **Optimize.** Using variational minimization, make the encoding reproduce the original image with minimal error.

The first step in encoding an image is to analyze the high resolution input to determine the primal complex. Objects and intersections are found and classified. When (d+1)-way intersections are found, they are preserved by inserting a "ghost" 0-skeleton element in the dual; however, a new region is not inserted into the primal complex. Rather, the ghost nodes are effectively ignored during decoding and the attribute assignment goes to the next closest 0-skeleton element. This ensures that these non-generic intersections are not lost or degenerate. An example of inserting a ghost element in 2D can be seen in Figure 4.

The dual complex is found by iterating over locations in the primal image, inspecting neighborhoods around each sample for differences in objects id's. N-way intersections are recorded by introducing the appropriate skeleton element into a simple data structure which maintains lists for object id's, embedded position $\vec{\mathrm{X}}_i$, indices into other skeleton lists, and associated attributes such as color.

Reparametrizing the data into a lower dimensional value space involves embedding the dual complex in a space $\mathbb{R}^E$, where E is the dimension of the target value space greater than or equal to the image dimension d. In particular, we are interested in a *straight-line* embedding in which the elements of the q-skeleton are parametrized as linear combinations of the elements in the 0-skeleton. For example, the elements of the 1-skeleton are lines (not curves). Although we can show that the dual complex is embeddable in a space of dimension $E = d$, this embedding may require curved q-skeleton elements. With an additional dimension, however, most skeletons can be straight-line embedded.

Embedding can be accomplished using a modified, force-directed graph solver which is adapted to handle CW-complex embeddings by introducing charged-particle forces for faces (dual 2-skeleton elements) and volumes (dual 3-skeleton elements). This is done by adding extra nodes at faces and volumes. These nodes are constrained to the center of their respective faces and volumes and are there only to force 0-skeleton nodes ($\vec{\mathrm{X}}_i$) away from these regions.

For 2D images it can be shown with a straightforward proof that a 3D target space (equivalent to the common color image value space of RGB) will always be sufficient for embedding. Ignoring elements in the 2-skeleton, the 0 and 1-skeletons form a planar 2D graph, which is straight-line embeddable in 2D [3]. The straight-line graph can be mapped to a plane in $\mathbb{R}^3$. Now, move all 0-skeleton elements that intersect the interior of a 2-skeleton element some distance normal to the embedding plane. Continue moving those who still intersect 2-skeleton interiors until all intersections are removed. We know that intersecting sub-complexes must be nested; if they are not, they would have overlapping edges, which have already been resolved by straight-line embedding the graph. Therefore, each lifting step will free at least one 2-cell. So, for finite complexes, this process will terminate and no intersections will remain.

For volumetric data, we default our embedding to a 4D value space (i.e. RGBA). It is easy to detect a false embedding by traversing the embedded elements of the skeleton, decoding their attribute vectors, and identifying any incorrectly assigned attributes. In practice we have never needed more than a 4D embedding for volumetric data, however it is possible to construct pathological cases where each object shares a boundary with every other object in the spatial domain. However, even these situations can be remedied by subdividing the objects.

Now that we have computed and laid out our dual complex, we are

ready to start creating the encoded image. We initialize the encoded image by assigning values from the indicator vector. If we let $\vec{X}_i$ be the embedded position of the 0-skeleton element χ_i^{*0} in the target space $\mathbb{R}^E$ and $\vec{v}$ be a pixel value, then the parametrized value $\vec{v}'$ is

$$\vec{v}' = \sum_i^N \vec{X}_i v_i, \tag{3}$$

where v_i is the i^{th} element of the value vector $\vec{v}$. When the indicator vector has a single 1 at position k and all other elements 0, we get $\vec{v}' = \vec{X}_k$, which basically says that we relabel our encoded image using the embedded position of k^{th} 0-skeleton element χ_k^{*0}. For 2D images, the initial encoded image is the same size as the original raster image and we initialize every pixel of it with the position of its corresponding dual-embedded 0-skeleton element. The downsample process that follows will produce a smoothed, or band-limited, encoding. In a similar way, we initialize the encoded image for 3D volume data by assigning to each sample the position of the embedded dual 0-skeleton element associated with the object at that location, except that in this case we do not get this information from a raster data set but by directly evaluating the function using the 3D geometry. It is also possible to blur the resulting encoded image to enforce smoothness. When the attribute is smooth (C^0 continuous or better), for example with distance transforms or posterior probabilities, the initial values can be weighted sums of the the embedded dual 0-skeleton elements.

Once the encoded image has been downsampled, we can choose to tweak the sample values in order to yield a more accurate reconstruction. This process can be described as a more general optimization problem. If we define a "decode" function $\gamma : \mathbb{R}^E \rightarrow \mathbb{R}^N$ that takes the encoded values and returns them to the discrete colors of the original image, then the encode step described by Equation 3 can be refined using a variational minimization of

$$\varepsilon(\{\vec{v}_i' | i = 1 \ldots \mathrm{N}\}) = \int_D |I_{\mathrm{v}}(\vec{x}) - \gamma(I_{\mathrm{e}}(\vec{x}))|_2 \, d\vec{x}, \tag{4}$$

where I_{v} is the original image function, I_{e} is the encoded image function, and $|\ldots|_2$ is the L2 norm. The characteristics of the encoding depend entirely on the choice of the original image function I_{v}, decode function γ, and interpolation kernel k. The goal of this optimization is to reproduce the original image function with minimal error. Note that neither I_{v} nor $\gamma(I_{\mathrm{e}})$ are C^0 continuous functions. Therefore, we cannot take derivatives of Equation 4 and solve it as a linear system. However, we can minimize Equation 4 with direct search methods [11], which can robustly and efficiently minimize discontinuous functions.

When the original image function is represented as a high resolution rasterized image, the integral in Equation 4 becomes a summation,

$$\varepsilon(\{\vec{v}_i' | i = 1 \ldots \mathrm{N}\}) = \sum_{j=1}^{M} \left| V_j - \gamma(I_{\mathrm{e}}(P_j)) \right|_2, \tag{5}$$

where V_j is a pixel value (an indicator vector) for object j at pixel location P_j in the original image, $\vec{v}_i$ is a pixel value in the encoded image, and M is the number of pixels in the rasterized original image. The direct search optimizer, also known as "pattern search" [11], works by modifying the unknowns $\vec{v}_i$ incrementally, accepting changes that lower the error ε in a greedy manner. Since we typically use interpolation kernels with finite support, we only need to compute the error summation for the region of the original image covered by the kernel support for that sample's location in the encoded image.

One important consideration for optimization is the precision of the target image. Although solvers typically operate on real-valued types (float or double), quantization to fixed precision types (such as 8-bit char) can affect the quality of the encoding. We have found it valuable to include knowledge of the target precision in the optimization. There are two ways to do this. In the direct search optimizer, one can restrict the increments to the size of the target precision's epsilons. Alternatively, one can periodically quantize the solution to the target precision during the optimization, while shrinking the step size. Orlin *et al.* provide a more thorough treatment of fixed precision optimization [19].

5.2 Decoding

Once encoded, the image can be "rendered" or decoded at any resolution. This involves the following steps:

1) **Resample encoded image** to desired resolution.
2) **Apply decoding function** γ for color assignment at each pixel.
3) **Antialias image**, an optional step which blends colors near object boundaries.

Decoding an encoded image requires knowledge of the embedded dual skeleton, which is encoded in the IStar structure. The process basically consists of resampling the image and evaluating γ for each sample. Throughout this paper, the γ function is a general attribute assignment function, which associates regions of value space to an indicator vector, color, position in value space, etc., depending on the context. When an image is encoded, the γ function captures the boundary discontinuities by partitioning (or cutting) the encoded value space. We discuss specific examples of γ functions in Section 6.

Once identified, the sample value is replaced by the attribute associated with object i. This attribute is usually a color, but can also be a more complicated function or shader. For a given γ function that returns an N dimensional indicator vector ($\vec{v} = \gamma(I_{\mathrm{e}}(\vec{x}))$), color assignment can be expressed as

$$\vec{c} = \sum_{i=1}^{N} v_i \vec{c}_i, \tag{6}$$

where c_i is the color associated with the i^{th} object. While both the original image I_{v} and "decoded" image $\gamma(I_{\mathrm{e}})$ functions are discontinuous, the encoded image function I_{e} is a continuous function, where the degree of continuity depends on the interpolation kernel. We can leverage this fact to perform analytic antialiasing. Given a γ function we can derive a signed distance function $g_i : \mathbb{R}^E \rightarrow \mathbb{R}$ that returns the distance to the partition in the encoded value space for object i.

The distance d_i to the boundary in the spatial domain, is

$$d_i(\vec{x}) = \frac{g_i(\vec{x})}{|\nabla g_i(\vec{x})|_2}. \tag{7}$$

Antialiasing can be achieved by blending based on the distance to the boundary, $\vec{c} = w\vec{c}_i + (1-w)\vec{c}_j$ where $w = \mathrm{clamp}(d_i(\vec{x})/2 + .5)$, $\vec{c}_i$ is the color associated with the X_i closest to $I_{\mathrm{e}}(\vec{x})$, and $\vec{c}_j$ is the color associated with the second closest X_j.

6 Discussion

6.1 The Decoding Function γ: Plane vs. Voronoi

One way to partition the value space is with a Voronoi tessellation. While the Voronoi-based γ function is simple to implement, it places a hard constraint on the embedding procedure for the dual skeleton: not only must the dual embed in the target value space, but it must also be arranged so that the Voronoi tessellation produces the correct results. For even moderately complicated images, it may be difficult to satisfy the Voronoi constraint. The use of Voronoi γ functions also has a subtle side-effect in the reconstructed images. Triple intersections in the primal domain tend to be shaped like the intersections in the value domain. Voronoi intersections have a "Y" shape in the value domain, so the intersections in the spatial domain are also slightly "Y" shaped, even when the image is aggressively optimized. This effect is caused by the smoothness of the interpolation kernel. We can address both these issues with a more flexible γ function that defines the tessellation using planes in the value space and allows us to create "T" junctions. In this case, each embedded 0-skeleton element X_i is bounded by a number of planes,

$$p_{ij}(\vec{x}) = \vec{w}_{ij}^T \vec{x} + w_{0,ij}, \tag{8}$$

where $p_{i,j}(\vec{x}) = 0$ is the plane separating X_i and X_j, $\vec{w}$ and w_0 are the plane coefficients. The γ function can be expressed as

$$\gamma(I_{\mathrm{e}}(\vec{x})) = \vec{a}_i \quad \text{where} \quad p_{ij}(I_{\mathrm{e}}(\vec{x})) > 0 \;\; \forall j, \tag{9}$$

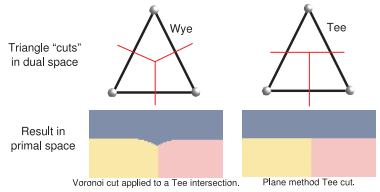

Fig. 6. A comparison of "Y" versus "T" intersections.

where $\vec{a}_i$ is the attribute such as color associated with object i. The value space distance functions for this γ are

$$g_i(\vec{x}) = \min_j \left(\frac{p_{ij}(\vec{x})}{|\vec{w}_{ij}|_2} \right). \tag{10}$$

This function simply returns the distance to the closest boundary plane $(p_{i,j})$ in the value space, and is used for antialiasing in Equation 7.

One advantage of a plane basis for γ is the fact that the planes can be arranged to preserve the structure of object boundaries in the value domain for any valid embedding. The second advantage is the ability to arrange the n-way object intersections in the value domain so that they minimize the decode error. Figure 6 illustrates the difference between "Y" intersections produced by Voronoi tessellations and "T" intersections produced using planes.

Antialiasing object boundaries requires us to capture the second closest X_j, i.e. the closest X_j that shares a 1-skeleton element with the selected X_i. This distinction is important since a plane may be necessary to separate two X_i's that happen to embed closely but do not share a boundary in the spatial domain. Since Equation 10 requires a division by the plane coefficients, we pre-normalize the coefficients so that this does not need to be done for each sample. This makes Equation 10 identical to Equation 8. The second closest X_j will be the one with the minimum g_{ij} that also shares a 1-skeleton element with X_i.

6.2 Considerations for Volumetric Data

The dual-parametrization method can also be used for 3D volume and higher dimensional images. The utility of dual-parametrization for volumetric data is less about compressing the image size (extents), and more about compactly representing object attributes. Consider the segmentation of a data set. Many segmentation and classification algorithms are capable of producing sub-pixel accurate results, for instance level-set and random walker methods [4, 28]. These methods require a scalar image for each segmented object to preserve this sub-pixel accuracy. Therefore, 10 segmented regions would require 10 scalar volumes. For this reason, many discard the continuous representation for a binary tagged data set, which identifies classified features only at data samples. Dual-parametrization can be directly applied, as described in the previous section, where I_v produces an indicator vector based on the segmentation's scalar fields. More general classification methods produce posterior probabilities or volume fractions for each object in the data [14, 26]. In this case, the dual-parametrization can be used to encode these continuously varying probabilities as seen in [10]. The only modification necessary to make that method a dual-parametrization is the connected-component subdivision of objects (condition (ii) in Section 4). In this case, connected objects and the dual structure can be identified based on maximum a-posteriori class assignments.

Level-set segmentations can easily be transformed into signed distance functions with respect to the segmentation boundary [28]. A dual encoding can capture this information as well by adapting the minimization from Equation 4 to

$$\varepsilon(\{\vec{v}_i' | i = 1 \ldots \mathrm{N}\}) = \sum_{j=1}^{N} \int_{D_j} \left(d_{lj}(\vec{x}) - d_{ej}(\vec{x}) \right)^2 d\vec{x} \tag{11}$$

where D_j is the region for which the level-set distance d_{lj} for object j is positive, and d_{ej} is the distance function from Equation 7. Unlike the discrete object representation discussed earlier, this functional has continuous derivatives.

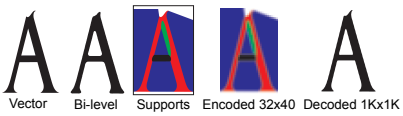

Fig. 7. Supporting cusps by introducing hidden support objects.

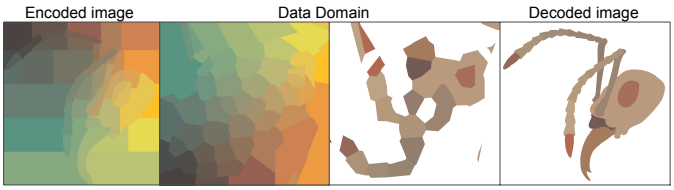

Fig. 8. An ant image. Left: encoded primal image. Left-center: cuts in the dual domain. Right-center: regions in dual domain assigned colors from the original palette. Right: decoded primal image.

We also implement Equation 11 as a summation,

$$\varepsilon(\{\vec{v}_i' | i = 1 \ldots \mathrm{N}\}) = \sum_{j=1}^{N} \sum_{k=1}^{M} \left(d_{lj}(P_k) - d_{ej}(P_k) \right)^2 h\left(d_{lj}(P_k) + \varepsilon \right), \tag{12}$$

where P_k is a pixel location in the image domain, and h is the Heaviside step function, which ensures that we only optimize regions within a distance ε from object i's boundary. We minimize this functional using an iterative Gauss-Seidel solver with successive over-relaxation.

6.3 Enforcing Stable Singularities

One of the advantages of IStar images over traditional raster images is that infinitely thin objects can be represented, such as lines and cusps. As noted earlier, a signed boundary distance function can be used to outline subjects. Therefore, to represent lines, we implicitly represent them as a boundary. A line can be forced to exist in a dual encoded image by adding additional objects so that their boundary is the line. These new objects are hidden when the illustration is rendered by making their color identical to the original object that they were cut from, with the exception of their boundary, which is rendered as a partial outline, the line itself. Cusps, or sharp edges, can be captured by 3-way object intersection in the dual encoded image, and be made an intrinsic property of a boundary by forcing a triple intersection at the cusp's point. Just as with lines, this is done by adding an extra hidden subject. Figure 7 shows the letter A with cusps supported by hidden subjects.

7 Results

In this section, we evaluate the IStar framework by first implementing the reconstruction algorithm on graphics hardware in order to texture map surfaces in real-time applications. Next, the quality of the reconstruction is compared against existing techniques for efficiently representing 2-D images, both for real-time and for offline applications. Finally, we demonstrate the preservation of semantic meaning in the IStar representation and give an example of how it can fix classification errors.

7.1 Graphics Hardware Implementation

The Voronoi decoding of IStar images was implemented as a fragment program on programmable graphics hardware. To do this, the compressed IStar bitmaps are bound to the appropriate surfaces in the scene and its texture samples are bilinearly interpolated on the hardware. The position and color of the nodes on the dual complex are provided as two additional non-interpolated textures. To compute the correct color, we must find the closest node to the upsampled value at each pixel. This can be done by comparing each node in turn, resulting in an algorithm with linear complexity with respect to the number of nodes. Instead, we use a constant-time algorithm that first divides the original image into a uniform grid of $m \times n$ cells in the spatial domain during encode. For each cell, we determine which regions (i.e. 0-cell elements in the dual skeleton) it contains, and the maximum number of regions in any single cell is computed. This information is then

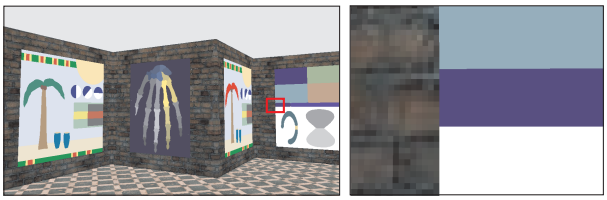

Fig. 9. Objects textured with IStar images for an interactive walkthrough. The scene is a "museum" of IStar images with a wide view on the left, and a detailed view on the right where the user has walked up to one of the artworks. The standard texture on the wall appears pixelated, the IStar image is not. This scene renders at about 480 fps.

output to a constant-sized list of nodes for every cell, where the size of the lists is determined by the cell with the most nodes. For cells with less than the maximum node count, the excess nodes are padded out with large values which will never be accessed during the nearest-node determination. During decode, we first determine which cell the point to be shaded is in, and compute the square-distance in histogram space from the upsampled value at that point to each of the nodes in that cell. A series of conditional move statements allows us to determine the node with the smallest square distance, thereby mapping the sample point to its Voronoi cell in the dual complex. This algorithm has constant complexity since it is limited by the maximum number of nodes in each cell. In our experiments, this number was less than 10, even for complex images. The rendering algorithm runs at real-time rates ($>$ 100 fps) at 1024×1024 resolution on an NVIDIA 7900GTX with 512MB of video memory. Figure 9 demonstrates how we can apply IStar images to general texture mapping by using them to texture map the walls of a walkthrough environment.

7.2 Quality of Reconstruction

To measure the quality of the IStar algorithm, we must compare the original primal image with the reconstructed result. However, the quality of the reconstruction varies with the amount of downsampling in the encoding process, making this method a form of lossy image compression. In other words, the algorithm is replacing a high-resolution bitmap with a smaller representation by trading memory space for algorithmic complexity.

Therefore, we also compare our algorithm to existing raster-based compression schemes. However, unlike our IStar representation, these do not take discontinuities into account. In particular, we compare against the three most common approaches used in real-time applications: downsampling with (1) nearest-neighbor and (2) linear upsampling, and (3) S3TC image compression. For completeness, we also compare against high-end offline image compression techniques such as jpeg and jpeg2000. We present our results in Figures 10 and 11, where the Peak-Signal to Noise Ratio (PSNR) is plotted against compression ratio for the different approaches (higher PSNR means better quality). The size of the entire IStar structure (compressed bitmap, dual skeleton, and color palette) is taken into account when doing our calculations. While the size of the skeleton and color information are constant, we can vary the size of the IStar structure by changing the amount of downsampling for the bitmap. While jpeg and jpeg2000 outperform our approach, these compression schemes are not compatible with real-time applications as their decompression algorithms cannot exploit the regular sampling and locality that traditional raster images can. In addition, jpeg and jpeg2000 compress the image at a fixed resolution, which means that magnification of the image will not preserve the boundary discontinuities. If desired, IStar images can be combined and optimized with jpeg or jpeg2000 and enjoy both aggressive compression and resolution independence. Figures 11 and 12 compare detail regions of a test image for each compression method.

The artifacts introduced by our approach at high compression ratios are very different in nature than those of other compression schemes. Rather than the ringing, blurring, or block artifacts of other compression schemes, high compression of IStar images introduces artifacts that are simplifications of the geometry of the dual encoded image, which are less perceptually objectionable.

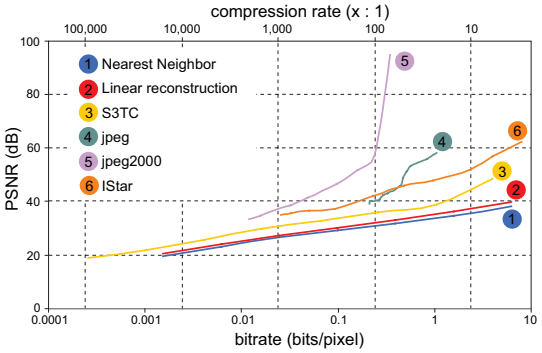

Fig. 10. Rate-distortion comparisons of PSNR versus compression rate for compression algorithms operating on the test image of Figure 12.

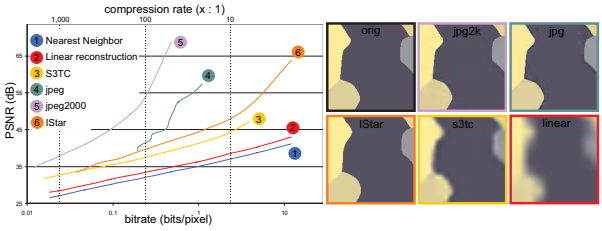

Fig. 11. Rate-distortion comparisons of PSNR versus compression rate for compression algorithms operating on the hand image of Figure 3.

7.3 Modification of Regional Attributes

The goal of most visualization applications is to produce images that highlight features of interest. IStar images are well suited for such applications because their structure stores distinct regions of the data as nodes in a dual graph, which can be used to add semantic information to each desired region. The assignment of node colors can be changed from that of the original color palette to produce false-color images or create special effects such as transparency. In addition, the antialiasing method presented in Section 5.2 can be altered to generate outlines, which can help highlight objects during visualization. Several of these effects are applied to a 2D test image in Figure 13.

These approaches can also be used to get around the limitation that our input primal image must have regions of constant color. In order to process an image with arbitrary gradients, we first add a preprocessing step to our pipeline where we segment the image into separate regions. This creates a constant-color image that can be used by the rest of the system. During rendering, the decoded regions can then be used as "region masks" while another source of data (a low-resolution texture, fragment shader, etc.) is used to reconstruct the gradient within those regions. As shown in Figure 13[e], this is a simple way to add gradients or high frequency texture detail to IStar images.

7.4 Fixing Classification Errors with Topological Information

An interesting application of the IStar structure is using the stored topological information to fix errors in the data set. Figure 14 (left image), shows a classified MRI of a human head that is known to contain errors, in this case incorrect classification that places white matter tissue in direct contact with both skull and cerebro-spinal fluid (CSF). We can enforce the anatomically-correct configuration in which CSF completely separates the skull and white matter by eliminating the edge between the white matter and skull nodes in the dual skeleton and then reparametrizing so that the white matter-skull interpolated values follow the white matter-CSF-skull boundary path in the value domain. The result of this reparametrization is the corrected data set shown on the right.

8 Future Work and Conclusions

Our continuing efforts will focus on developing the IStar framework for more general image classes, including arbitrarily varying attributes, such as color gradients and texture. We believe this approach

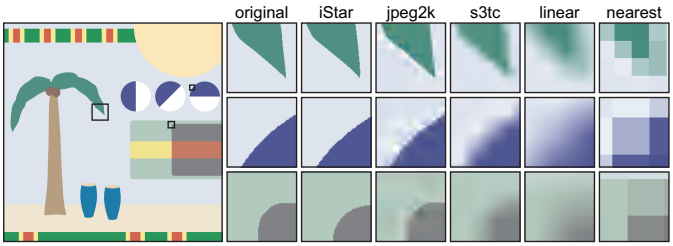

Fig. 12. Comparison of compression artifacts. The image on the left was compressed 1000 : 1 with various algorithms. The insets show the artifacts introduced by the algorithms for the regions specified. "Linear" and "nearest" refer to a bilinear interpolation and nearest-neighbor upsampling of the downsampled data, respectively. Our algorithm does very well against the real-time approaches and even compares favorably against jpeg2000. Note that IStar artifacts are geometric simplification.

will benefit multi-class classification and segmentation methods as an acceleration mechanism. The compact data representation may be able to eliminate the need for computing independent scalar fields for each class and allow all class probabilities to be solved simultaneously. We are also interested in automating the cusp support algorithm and providing a postscript decode implementation.

We have presented a new topological description for continuous images in both 2D and 3D. This topological description allows images to be reparametrized so that discontinuities are preserved at arbitrary resolutions. By applying the machinery of topology, lower bounds can be found for the dimension of a value space that is capable of preserving the uniqueness of n-way object intersections. We have demonstrated that this preservation of semantic structure can enhance 3D visualizations and 2D images.

9 Acknowledgements

We would like to thank Mariana Ruiz for the ant image of Figure 8. We also thank Ross Whitaker for seeding some of these ideas and Bobby Hanson for invaluable help with the math. This work was supported in part by the DOE High Performance Computing Graduate Fellowship and NSF Grant CCF-0702787.

References

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

[2] T. Dey, H. Edelsbrunner, and S. Guha. Computational topology. *Advances in Discrete and Computational Geometry (Contemporary mathematics 223), American Mathematical Society*, pages 109–143, 1999.

[3] I. Fary. On straight line representation of planar graphs. *Acta Univ. Szeged. Sect. Sci. Math.*, 11:229–233, 1948.

[4] L. Grady. Multilabel random walker image segmentation using prior models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '05)*, volume 1, pages 763–770, 2005.

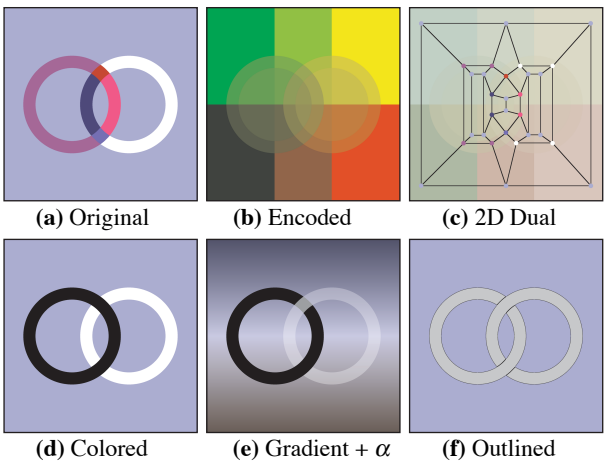

Fig. 13. A comparison of different attribute assignments.

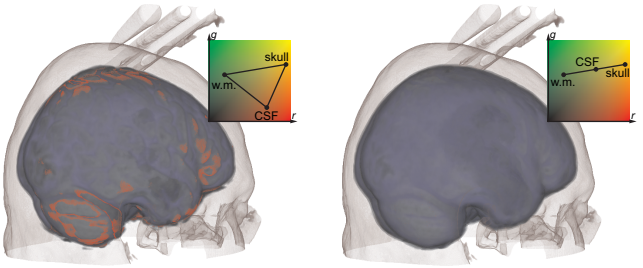

Fig. 14. Visualizing errors in classification and reparametrizing to enforce a correct anatomical model. (Left) red regions outline incorrect boundaries between skull and white matter. The dual skeleton (inset) shows connectivity between all three tissues. (Right) rendering from reparametrized data which disallows skull-white matter boundaries. The dual skeleton has been modified and laid out so that values interpolated by the white tissue-skull boundary are mapped to the CSF. Data source: BrainWeb Phantom.

[5] A. Gyulassy, V. Natarajan, V. Pascucci, P.-T. Bremer, and B. Hamann. Topology-based simplification for feature extraction from 3d scalar fields. In *IEEE Visualization 2005*, pages 275–280, 2005.

[6] J. Hart. Computational topology for shape modeling. *Shape Modeling International '99*, pages 36–45, 1999.

[7] A. Hatcher. *Algebraic Topology*. University Press, Cambridge, 2002.

[8] J. Helman and L. Hesselink. Representation and display of vector field topology in fluid flow data sets. *IEEE Computer*, pages 27–36, Aug. 1989.

[9] J. Kniss and R. Hanson. Images and dual dataspaces. Technical report, University of Utah, 2006.

[10] J. Kniss, R. V. Uitert, A. Stephens, G. Li, T. Tasdizen, and C. Hansen. Statictically quantitative volume visualization. In *IEEE Visualization 2005*, pages 287–294, 2005.

[11] T. Kolda, R. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45(3):385–482, 2004.

[12] T. Kong, A. Roscoe, and A. Rosenfeld. Concepts of digital topology. *Topology and its Applications*, 46:219–262, 1992.

[13] T. Kong and A. Rosenfeld. Digital topology: Introduction and survey. *Computer Vision, Graphics, and Image Proc.*, 48(3):357–393, Dec. 1989.

[14] D. Laidlaw. *Geometric Model Extraction from Magnetic Resonance Volume Data*. PhD thesis, California Institute of Technology, 1995.

[15] C. Loop and J. Blinn. Resolution independent curve rendering using programmable graphics hardware. *ACM Transactions on Graphics*, 24(3):1000–1009, 2005.

[16] J. Loviscach and H. Bremen. Efficient magnification of bilevel textures. In *ACM SIGGRAPH 2005 Conference Abstracts and Applications*, 2005.

[17] J. Matas, R. Marik, and J. Kittler. The color adjacency graph representation of multi-coloured objects. Technical Report VSSP-TR-1/95, University of Surrey, 1995.

[18] J. Munkres. *Topology*. Prentice-Hall, Englewood Cliffs, NJ, 1975.

[19] J. B. Orlin, A. S. Schulz, and S. Sengupta. ε-optimization schemes and L-bit precision: alternative perspectives in combinatorial optimization. In *ACM Symposium on Theory of Computing*, pages 565–572, 2000.

[20] V. Pascucci. *Topology Diagram of Scalar Fields in Scientific Visualization*, chapter Chapter 8 in Topological Data Structures for Surfaces. 2004.

[21] N. Ray, X. Cavin, and B. Levy. Vector texture maps on the gpu. Technical Report ALICE-TR-05-003, 2005.

[22] G. Scheuermann and X. Tricoche. *Visualization Handbook*, chapter Topological Methods in Flow Visualization, pages 341–356. Elsevier, 2004.

[23] P. Sen. Silhouette maps for improved texture magnification. In *Proceedings of Graphics Hardware*, pages 65–73, 2004.

[24] P. Sen, M. Cammarano, and P. Hanrahan. Shadow silhouette maps. *ACM Transactions on Graphics*, 22(3):521–526, 2003.

[25] A. Systems. *Postscript Language Reference Manual*. 1985.

[26] T. Tasdizen, S. Awate, R. Whitaker, and N. Foster. Mri tissue classification with neighborhood statistics: A nonparametric, entropy-minimizing approach. In *Proceedings of (MICCAI)*, volume 2, pages 517–525, 2005.

[27] J. Tumblin and P. Choudhury. Bixels: Picture samples with sharp embedded boundaries. In *Proceedings of the Eurographics Symposium on Rendering*, pages 186–196, 2004.

[28] R. Whitaker, D. Breen, K. Museth, and N. Soni. A framework for level set segmentation of volume datasets. In *Proceedings of ACM Workshop on Volume Graphics*, pages 159–168, June 2001.