Statistically Quantitative Volume Visualization

Joe M. Kniss, University of Utah
Robert Van Uitert, National Institutes of Health
Abraham Stephens, University of Utah
Guo-Shi Li, University of Utah
Tolga Tasdizen, University of Utah
Charles Hansen, University of Utah

Abstract

Visualization users are increasingly in need of techniques for assessing quantitative uncertainty and error in the images produced. Statistical segmentation algorithms compute these quantitative results, yet volume rendering tools typically produce only qualitative imagery via transfer function-based classification. This paper presents a visualization technique that allows users to interactively explore the uncertainty, risk, and probabilistic decision of surface boundaries. Our approach makes it possible to directly visualize the combined "fuzzy" classification results from multiple segmentations by combining these data into a unified probabilistic data space. We represent this unified space, the combination of scalar volumes from numerous segmentations, using a novel graph-based dimensionality reduction scheme. The scheme both dramatically reduces the dataset size and is suitable for efficient, high quality, quantitative visualization. Lastly, we show that the statistical risk arising from overlapping segmentations is a robust measure for visualizing features and assigning optical properties.

Keywords: volume visualization, uncertainty, classification, risk analysis

1 Introduction

Volume visualization endeavors to provide meaningful images of features "embedded" in data. There has been a significant amount of research over the past 17 years on providing visualization of volume data [4, 6, 17, 19]. Interactive volume visualization strives to allow the user to highlight features of interest in the volume data, such as material boundaries or different tissue types. Such features are dependent on a number of factors: the kind of data, domain specific knowledge, and the user's semantics. Simultaneously, there has been progress towards classifying features from volumetric data [5, 7, 21]. While segmentation is not considered to be a solved problem, there exist many different methods for segmenting volume data [10].

The demand for more quantitative measures in visualization has grown both within the visualization community and with the users of visualization tools. In volume rendering applications, transfer functions have typically been used for both classification and assignment of optical properties. However, using transfer functions for classification limits the user's ability to change the type of classification that occurs and does not provide any quantifiable measure of uncertainty. Transfer functions also tend to be unintuitive to use and do not provide the user with a clear concept of how classification is being performed on the data.

Statistical classification and segmentation methods incorporate a probabilistic model of the data and feature behaviors, a sophisticated notion of spatial locality, as well as the ability for the user to input their expertise. Interaction with this kind of probabilistic data and decision rules can provide each user the ability to define what information is important to his/her particular task as part of the visualization.

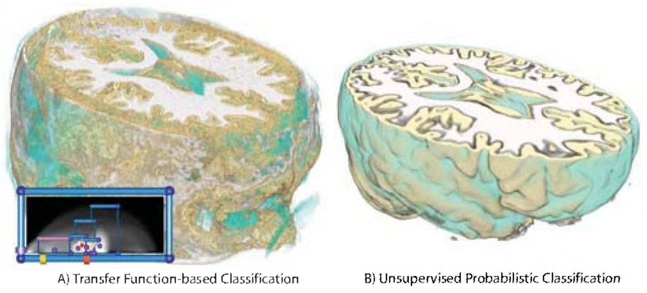
A) Transfer Function-based Classification
B) Unsupervised Probabilistic Classification

Figure 1: A comparison of transfer function-based classification versus data-specific probabilistic classification. Both images are based on T1 MRI scans of a human head and show fuzzy classified white-matter, gray-matter, and cerebro-spinal fluid. Subfigure A shows the results of classification using a carefully designed 2D transfer function based on data value and gradient magnitude. Subfigure B shows a visualization of the data classified using a fully automatic, atlas-based method that infers class statistics using minimum entropy, non-parametric density estimation [21].

In this paper, we propose a system that provides the user access to the quantitative information computed during fuzzy segmentation. The decision making step of classification is deferred until render time, allowing the user finer control of the "importance" of each class. Unfortunately, postponing this decision result comes at the cost of increased memory consumption. To accomodate this memory use, we propose a data dimensionality reduction (DDR) scheme that is designed to accurately represent pre-classified or segmented data for visualization. This approach allows data to be classified using the most appropriate fuzzy segmentation method, while utilizing existing volume visualization techniques. We also show that statistical risk is a robust measure for visualizing features and assigning optical properties.

2 Previous Work

There has been an enormous number of publications on both volume visualization and classification/segmentation. A comprehensive overview is outside the scope of this paper. For an extensive overview of volume rendering, the reader is refered to an excellent survey by Kaufman and Mueller [13]. The book by Duda *et al.* provides a solid introduction to the topic of statistical classification and segmentation [5]. Stalling *et al.* demonstrate the utility of fuzzy probabilistic classification for creating smooth, sub-voxel accurate models and visualization [20].

Using transfer functions for volume rendering involves

IEEE Visualization 2005
October 23-28, Minneapolis, MN, USA
0-7803-9462-3/05/$20.00 ©2005 IEEE.

mapping data values to optical properties such as color and opacity [17, 4]. Transfer function design is a difficult process, especially when features are indistinguishable based on data value alone. As such, researchers have investigated augmenting the domain of the transfer function with derivative information to better disambiguate homogeneous materials and the boundaries between them [14, 15].

Laidlaw demonstrated the effectiveness of classification techniques that define an explicit material mixture model and incorporate a feature space that includes spatial neighborhood information [16]. Recently, a number of visualization research efforts have begun to leverage high quality classification techniques to enhance the expressiveness of transfer function design. Bajaj *et al.* show how statistical analysis can drive the creation of transfer function lookup tables [1]. Tzeng *et al.* demonstrate two kinds of binary discriminant classifiers for transfer function specification using artificial neural networks and support vector machines [24]. Their approach illustrates the benefits of a robust feature space including local spatial information. In later work, Tzeng *et al.* utilize a cluster-based discriminant and discuss the importance of fuzzy classification with respect to material boundaries [25].

Others take a different approach in dealing with the difficulties of transfer function-based classification and color mapping by separating classification from the transfer function entirely. Tiede *et al.* describe a technique for volume rendering *attributed* or tagged data that smoothes feature boundaries by analyzing the relationship between the tags and original scalar data [23]. Hadwigger *et al.* and Viola *et al.* describe techniques for rendering tagged data that extends the approach of Tiede using a sophisticated hardware accelerated system and novel rendering modalities for pre-classified or segmented data [9, 26]. Bonnell *et al.* describe a method for the geometric extraction of features represented in data with volume-fraction information [2].

There has been a recent call from within the visualization community for visualization techniques that provide a rigorous treatment of uncertainty and error in the images they produce [12]. Grigoryan and Rheihgens present a point-based approach for representing spatial uncertainty in segmented data [8]. Whittenbrink *et al.* describe how geometric glyphs can be used to express uncertainty in vector valued data fields [27].

A number of dimensionality reduction techniques have been developed to either detect low-dimensional feature manifolds in a high dimensional data-space or reduce the dimensionality of the data-space while preserving relationships between data samples. Principal component analysis and independent component analysis are examples of linear dimensionality reduction [5]. ISOMAP and Local Linear Embedding are examples of non-linear manifold learning techniques that attempt to "flatten out" a sparsely sampled manifold embedded in a higher dimensional space while preserving the geodesic distance between points [18, 22].

3 General Statistical Classification

Classification of image data in either 2D or 3D is a special case of a more general data classification. Since nearly all image data share the characteristic that samples are spatially correlated, the "feature space" for image data classification includes not only data values, but also spatial relationships.

Rather than discussing a specific classification scheme for image data, we would like to focus on the more general statistical classification process and its application to visualization. There are five basic elements of the statistical classification process that need to be considered when designing a classifier: feature selection, classifier selection, parameter estimation, class conditional probability estimation, and decision and risk analysis. The remainder of this section covers a statistical framework for image data classification for use in visualization applications and describes each of these steps in further detail.

3.1 Feature Selection

The first step in classifying data is to decide what features should be identified and subsequently visualized. The features are the different classes that exist in the data, which we will identify as ω_i, representing physical items such as white matter and gray matter in MRI brain data, or more abstract phenomena like warm and cold air-masses in numerical weather simulation.

3.2 Classifier Selection

Before features can be classified, it is necessary to understand how they are represented by the raw image data. For scanned image data, *e.g.* MRI or CT, there are several assumptions commonly made with respect to features in the acquired signal. These assumptions can help guide the construction of a statistical feature model. One common assumption is that discrete materials tend to generate nearly constant scanned values, *i.e.* if two samples come from the same material, their signal intensities should be the same. It is assumed that data sample values are degraded or perturbed by an independent noise source due to thermal variation and electro-magnetic interference. If the noise model can be adequately characterized as a probabilistic distribution, it dictates the expected variation of data value for a locally homogeneous material. Because data is only available in a discrete form, it is also assumed that the signal is band limited and that the sample values are mixtures of discrete materials near that sample. This assumption of partial volume effects allows one to predict, or model, how data values for multiple classes mix near boundaries. If for no other reason, partial volume effects alone suggest that *a-priori*, discrete class assignment of data samples is a poor choice for representing classified data. That is, partial volume effects indicate that the classification of data samples near feature boundaries is inherently fuzzy.

3.3 Parameter Estimation

If the feature model is parametric the next step is the estimation of the model parameters. For instance, if materials are represented as Gaussian distributions, it is necessary to identify the mean and standard deviation for each of the materials to be classified.

Not all feature models are parametric however, *i.e.* there may not be an explicit, *a-priori* model for which to estimate parameters. For instance, consider an artificial neural network as a classifier. With this type of classifier, the model and its parameters are implicit, and must be inferred from a training set. The training set is a set of samples and the associated class memberships identified by a user, which are used to "teach" the classifier the relationships between feature vectors (data values) and classes. A training set might also be used as segmentation seed points.

3.4 Class Probability Estimation

Once an appropriate feature model has been developed and parameters identified for each feature of interest, it is possible to compute the class conditional probabilities for each sample in the dataset. When these probabilities are calculated using only the global feature model, with respect to individual samples or feature vectors ($\vec{x}$), we call this the probabilistic likelihood $P(\vec{x}|\omega_i)$. What is wanted, however, is the posterior distribution $P(\omega_i|\vec{x})$, which weighs the likelihood against observed evidence and prior information. Bayes Rule provides the relationship between the posterior distribution and likelihood,

$$P(\omega_i|\vec{x}) = \frac{P(\vec{x}|\omega_i)P(\omega_i)}{\sum_{i=1}^{\mathbf{C}} P(\vec{x}|\omega_i)P(\omega_i)}$$

where $\mathbf{C}$ is the number of classes, $P(\omega_i)$ is the prior probability of class ω_i, and the denominator is a normalization factor that insures that $\sum_{i=1}^{\mathbf{C}} P(\omega_i|\vec{x}) = 1$.

3.5 Decision and Risk Analysis

Conditional risk $R(\omega_i, x)$ describes the loss incurred for deciding that a sample x belongs to class ω_i based on multiple class conditional probabilities,

$$R(\omega_i, \vec{x}) = \sum_{j=1}^{\mathbf{C}} \lambda(\omega_i, \omega_j)P(\omega_j|\vec{x}) \tag{2}$$

where $\mathbf{C}$ is the number of classes, $\lambda(\omega_i, \omega_j)$ is the risk weight, which expresses the cost associated with deciding ω_i when the true state of nature is ω_j. The optimal, discrete class assignment rule for some feature sample $\vec{x}$ is the ω_i that minimizes $R(\omega_i, \vec{x})$, and is commonly known as the Bayes Risk or Bayes Decision Rule.

The *maximum a-posteriori* discriminant, also known as the "0-1 risk" decision rule, is commonly used when building a tag volume from class conditional probabilities. It is a special case in which the minimum risk class decision is simply the class with the maximum conditional probability. The risk weights in this case are,

$$\lambda(\omega_i, \omega_j) = \begin{cases} 1 & i \neq j \\ 0 & i = j \end{cases}$$

The risk function becomes simply $R(\omega_i, \vec{x}) = 1 - P(\omega_i|\vec{x})$.

Another constructive way of reasoning about risk is to consider what minimum value of $\lambda(\omega_i, \omega_j)$ with respect to $\vec{x}$ would be required to make class ω_i the minimum risk class. This can be expressed as

$$\lambda(\omega_i, \vec{x}) = \max_{j, i \neq j} \left(\frac{P(\omega_j|\vec{x})}{P(\omega_i|\vec{x})} \right) \tag{4}$$

We call this the "risk-ratio". Figure 2 shows the relationship between probabilities and their ratios for a 1D feature space (x) and two classes. For compactness we denote $P_1(x) \equiv P(\omega_1|x)$ and $P_2(x) \equiv P(\omega_2|x)$. Because it is often useful to work in "log-probability" space, Figure 2 also plots the log probabilities and log probability ratios (log risk ratios). Figure 2A shows plots for a pair of non-normalized distributions. Figure 2B shows the plots for the normalized distributions based on $P_1(x)$ and $P_2(x)$ from Figure 2A. Notice that neither the probability ratio nor log probability ratio is changed by normalization. Also notice that the log risk-ratios have a zero-crossing at the "0-1" risk boundary, denoted by the vertical line labeled **B**. In the following section, we will leverage the behavior of the log risk ratio to design a continuous discriminant function suitable for visualizing the relationships between multiple class probabilities.

While specifying or manipulating the $\lambda(\omega_i, \omega_j)$ risk weight for each pair of classes is extremely useful for exploring uncertainty in the classification, we have found that in practice it is often tedious and cumbersome. Instead, it is preferable to specify a weight that describes the "importance" of a class. From this weight it is possible to derive the risk weight as

$$\lambda(\omega_i, \omega_j) = \begin{cases} \mu_j/\mu_i & i \neq j \\ 0 & i = j \end{cases} \tag{5}$$

where μ_i is a user specified importance weight for class ω_i.

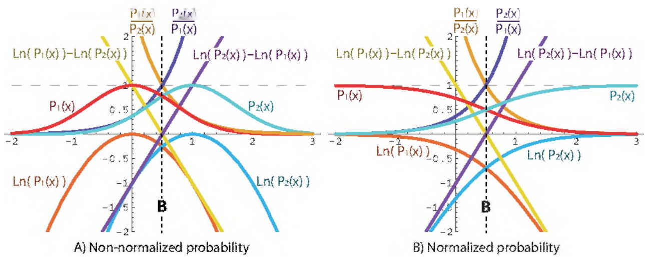

Figure 2: A 1D example of probabilistic boundary behavior. The graphs plot the relationship between probability, log-probability, probability ratio, and log-probability ratio in multi-class uncertainty analysis. Figure B shows these quantities for normalized probabilities.

4 Visualizing Classified Data

Our approach for visualizing classified data advocates decoupling the first four primary stages of classification from the transfer function and deferring the final step, the decision, until a sample is rendered. This requires the fuzzy class probabilities to be included with the data used for rendering. The advantage of using the fuzzy probabilities is that they interpolate, unlike discrete class assignments, and allow the transfer function design to be greatly simplified.

4.1 Color Mapping Multi-Class Probabilities

Figure 3 shows a simple, synthetic 2D example that illustrates various approaches for color mapping based on class probabilities from a realistic classifier, iso-surfaces, and transfer function-based classification. The simulated raw data (Figure 3B) was created by assigning a unique intensity value to each of the generated materials (Figure 3A), rasterizing the materials into a 256^2 image, blurring the image, and finally adding three percent normally distributed noise. Figure 3C shows four relevant iso-value thresholds (Taken at intervals between the class means) as subimages. The posterior class conditional probabilities were estimated using the known parameters (Figure 3 D); mean data value, noise distribution, and a neighborhood size proportional to the blur kernel. Figure 3E shows the image color mapped based on the class with the maximum probability (0-1 risk decision), as is often done when generating "tagged data". Figure 3F shows a color mapping based on class probabilities greater than a threshold of 0.5 for all classes; all data values containing a probability less than 0.5 are shown as black. Figure 3G shows the image with colors weighted by the minimum reciprocal-risk-ratio, $w_i = 1/\lambda(\omega_i, \vec{x})$). Notice that the boundaries are crisper than in the probability weighted example and that the variation in thickness for the loop (material e) is easier to see. Figure 3H shows a color mapping based on the 0-1 risk decision, with the addition of two importance weighted risk decisions for material e, where $\mu_e = 1.15$ and $\mu_e = 1.5$. The additional max risk-ratios were blended over the color map weighted by $1/\mu_e$. Finally, Figure 3I shows a color mapping made using a carefully designed 2D transfer function, based on data value and gradient magnitude. Because gradient estimation is highly sensitive to noise, the 2D transfer function performed quite poorly with the raw data (top-right subfigure), even though the gradient was estimated using the derivative of a cubic b-spline kernel, which implicitly blurs the data. To accommodate for the noise, the data was pre-processed using a median filter with a width of five pixels before gradient computation (Figure 3I, bottom-right subfigure).

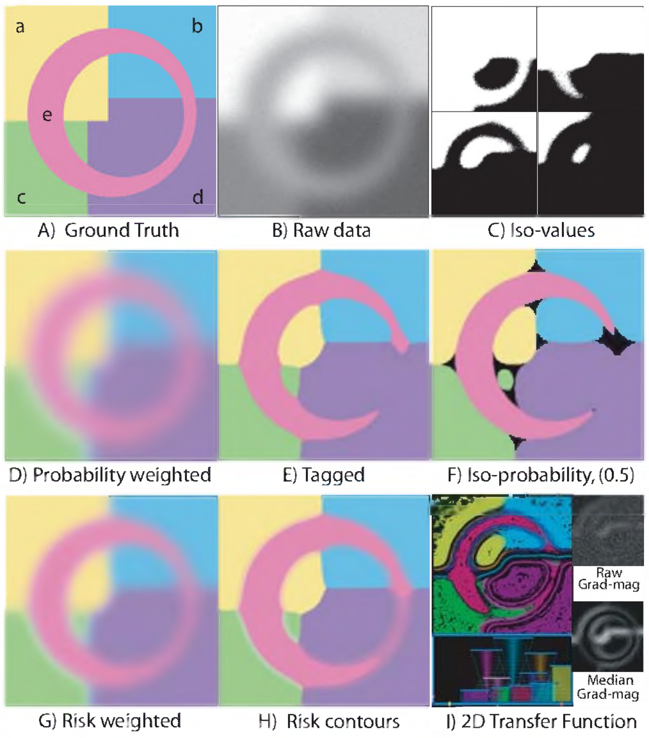

Figure 3: A 2D example of probabilistic boundary behavior. **A)** The synthetic dataset, consisting of five materials. **B)** The raw dataset constructed from a blurred monochrome version of the synthetic dataset with noise added. **C)** The four most relevant iso-value thresholds of the raw data as subimages. **D)** An image colored based on the class conditional probabilities of the classified raw data. **E)** A "max-probability" tagged image. **F)** The data set color mapped based on a probability threshold of 0.5. **G)** An image colored based the probability ratios (risk curves). **H)** An image showing several risk contours for material "e". **I)** Data color mapped using a carefully hand tuned 2D transfer function, based on raw data value and the gradient magnitude of the median filtered raw data.

4.2 Risk-centric Transfer Functions

Instead of taking a **D** dimensional vector of raw-data values as input, a transfer function based on class conditional probabilities transforms a **C** dimensional vector into the optical properties needed for rendering, where **C** is the number of classes. While it may seem appropriate to use the class conditional probabilities as input to the transfer function, as described in Section 3.5, the relationships between the individual posterior probabilities are best expressed in terms of risk. A reasonable choice is the **C** dimensional risk vector, $\vec{\Lambda}(\vec{x}) = [\lambda_0(\vec{x}) \ldots \lambda_{\mathbf{C}}(\vec{x})]^T$, where $\lambda_i(\vec{x}) = R(\omega_i, \vec{x})$ from Equation 2.

Unfortunately, this expression of risk does not provide much more information than the raw probabilities. Instead, it is preferable to use a discriminant function that we call the minimum *decision boundary distance*:

$$\lambda_i(\vec{x}) = \max_{j, j \neq i} \left(\log \left(\lambda(\omega_j, \omega_i) \frac{P_j(\vec{x})}{P_i(\vec{x})} \right) \right) \quad (6)$$

where we are using the short-hand $P_i(\vec{x}) \equiv P(\omega_i|\vec{x})$. This can be rewritten as

$$\lambda_i(\vec{x}) = \max_{j, j \neq i} \left(\log(\lambda(\omega_j, \omega_i)) + \log(P_j(\vec{x})) - \log(P_i(\vec{x})) \right)$$

In terms of importance weights μ_i, the minimum decision boundary distance is the maximum over all $j \neq i$

$$\lambda_i(\vec{x}) = \log(\mu_j) + \log(P_j(\vec{x})) - \log(\mu_i) - \log(P_i(\vec{x})) \quad (8)$$

The benefit of this expression is that it places the decision boundary, with respect to class ω_i, at $\lambda_i(\vec{x}) = 0$, with negative values indicating that class ω_i is the minimum risk class, and positive values indicating that it is not. It also has a more linear behavior than the probability ratio, and is invariant with respect to normalization (or any other uniform scaling) of the class conditional probabilities. For Gaussian distributions with the same standard deviation, this term is exactly the minimum decision boundary distance (in the feature space) scaled by $2\|\vec{c}_i - \vec{c}_j\|$, two times the distance between their means or centers. Figure 4 illustrates the behavior of this term for three different class distributions in a 1D feature space (x), with a varying importance term for class 2. Notice that in Figure 4C a small increase in μ_2 was able to make class 2 the minimum risk class, even though it would not have been using the maximum *a-posteriori* decision rule, used in Figure 4A. The arrows below the plots indicate the range over which each class is the minimum risk decision. In Figure 4B all three classes are the minimum risk at the origin.

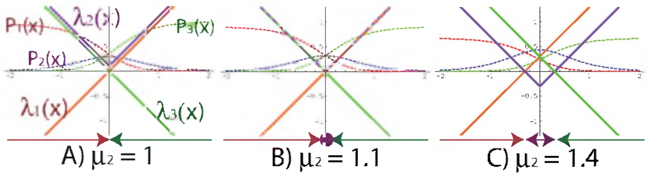

Figure 4: Behavior of the decision boundary distance function with respect to a varying importance term (mu).

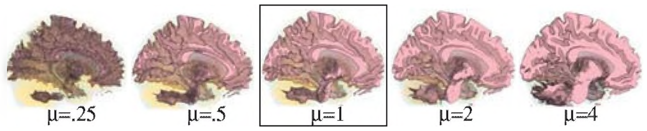

Figure 5: Effect of varying the importance term for white matter in a classified brain dataset visualization.

Like iso-surfaces, risk surfaces, *i.e.* spatial decision boundaries, have a number of desirable properties; water tight, easy geometric extraction. Unlike iso-surfaces, risk surfaces can support interesting boundary configurations, non-manifold 3-way and 4-way intersections, whereas iso-surfaces only support manifold 2-way interfaces.

5 Reparameterization

The increased storage size required for representing multiple fuzzy classified features is an important issue for interactive rendering. Most hardware based rendering platforms place hard restrictions on dataset size. Increased data size also has a dramatic effect on data access bandwidth, a prime concern for rendering efficiency, which is arguably a more pressing issue than memory capacity limitations.

To address the problems associated with increased data set size, we need a data-space transformation $T(\vec{c} \in \Re^{\mathbf{C}}) \rightarrow \Re^{\mathbf{P}}$ with the following properties:

1) Reduces the dimensionality of the dataspace; $\mathbf{P} \ll \mathbf{C}$.
2) Invertible with minimal error; $\|\vec{c} - T^{-1}(T(\vec{c}))\| < \epsilon$.
3) Encoded values can be interpolated prior to decode;
4) Decoding has minimal algorithmic complexity.

The criteria above describe a transformation, or encoding, of the data that effectively compresses the data, while allowing the conditional probabilities to be reconstructed after the data has been resampled during rendering. While dimensionality reduction (criterion 1) helps us solve the problem

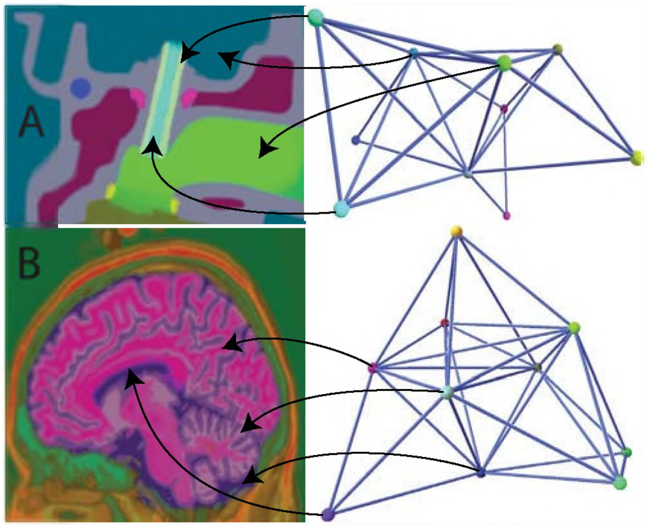

Figure 6: Slices of classified datasets reparameterized into a 3D data-space, and their associated graphs. The color is generated by mapping the 3D data coordinates directly to RGB colors. Subfigure A shows a classified engine dataset, and Subfigure B shows the BrainWeb Phantom fuzzy classifed data [3].

of increased storage and bandwidth, the criterion of interpolation prior to decode (3) helps eliminate redundant computation during the resampling and gradient estimation stages of the rendering pipeline.

5.1 Graph-based Dimensionality Reduction GDR

Our approach models the transformation ($T(\vec{c})$) as a graph layout problem, and is similar to work done independently by Iwata *et al.* [11]. Nodes represent pure classes, $P(\omega_i|\vec{x}) = 1$, and edges represent mixtures of multiple classes. Once connectivity of the graph is known, it is laid out in a space with a dimension $\mathbf{P}$ of our choosing, optimizing the spacing between nodes so there is no overlap of edges. The node locations are then used as a sparse data interpolation system, which serves as the inverse mapping $T^{-1}(\vec{p})$. All data samples $\vec{c}$ are then mapped to this parameterization space by finding the position $\vec{p}$ that minimizes the difference between $T^{-1}(\vec{p})$ and $\vec{c}$. Figure 6 illustrates this method applied to 2 classified datasets using a 3D reparameterization.

5.1.1 Graph Construction

Graph construction begins with identifying edge weights e_{ij} for each pair of class nodes $\mathcal{N}_i$ and $\mathcal{N}_j$. Edge weight is the covariance of the class probabilities assuming a mean of 1 for each class,

$$e_{ij} = \sum_{k=1}^{N} P(\omega_i|x_k)P(\omega_j|x_k)$$

where N is the number of samples in the dataset. In addition to the pair-wise class variance, we are interested in identifying higher order mixtures. To do this, an additional node is added to the system for any significant higher order mixtures. These nodes have an associated weight, n, that is the higher order variance for the mixture type it represents. For instance the three way mixture variance is

$$n_{ijk} = \sum_{l=1}^{N} P(\omega_i|x_l)P(\omega_j|x_l)P(\omega_k|x_l)$$

Note that for volume data these variance weights represent fuzzy boundaries between classified features. In general, most edge and higher order node weights are zero, *i.e.* the features/classes do not touch in the spatial domain; this is a property that our data parameterization method exploits.

5.1.2 Graph Layout

Once the edge and higher order node weights are determined, they are normalized based on the maximum weight. These weights are then used to compute potentials for a force directed graph layout. Our solver treats edges as springs with a unit natural length, and nodes as charged particles, which repel one another. The solver seeks to minimize an energy function with respect to the class nodes, which are positions in a $\mathbf{P}$ dimensional space;

$$\mathcal{E}(\vec{\mathcal{N}_2},\ldots,\vec{\mathcal{N}_\mathbf{C}}) = \sum_{i=1}^{M}\sum_{j=i+1}^{M} \|\varphi\left(\vec{\mathcal{N}_i},\vec{\mathcal{N}_j}\right)\|$$

where $\mathbf{C}$ is the number of classes, M is the total number of nodes in the system including the higher order variance nodes, and $\varphi(\vec{\mathcal{N}_i},\vec{\mathcal{N}_j})$ is the force function. The first node, $\mathcal{N}_1$, is constrained to the origin of the ambient space. The nodes representing higher order variance, $\mathcal{N}_{\mathbf{C}+1}\ldots\mathcal{N}_M$, are constrained to the average position of the class nodes whose variance they represent. The role of these nodes is to insure that class node placement does not interfere with spaces that represent important feature mixtures.

The force function, $\varphi(\vec{\mathcal{N}_i},\vec{\mathcal{N}_j})$, for the charged particle and spring edge model is,

$$\varphi(\vec{\mathcal{N}_i},\vec{\mathcal{N}_j}) = \left(e_{ij}(\|\vec{d}\|-1) + n_i n_j \exp(-\|\vec{d}\|^2)\right)\vec{d}$$

where $\vec{d} = \vec{\mathcal{N}_i} - \vec{\mathcal{N}_j}$, and n_i is a higher order variance node weight or 1 if $\vec{\mathcal{N}_i}$ is a class node. If either of the nodes represent a higher order variance, the edge weight e_{ij} is 0. This function returns the force vector with respect to node $\vec{\mathcal{N}_i}$. It is anti-symmetric with respect to the order of its parameters, *i.e.* $\varphi(\vec{\mathcal{N}_i},\vec{\mathcal{N}_j}) = -\varphi(\vec{\mathcal{N}_j},\vec{\mathcal{N}_i})$.

5.1.3 Sparse Data Interpolation and Encoding

Once we have laid out the graph in our target space, the class nodes serve as fiducials for a sparse data interpolation scheme. For this we choose Gaussian radial basis functions. This sparse data interpolation scheme defines a mapping, $T^{-1}(\vec{p})$, from our encoding space back to the $\mathbf{C}$ dimensional probability space. For some point $\vec{p}$ in the encoding space, the corresponding vector of conditional probabilities, $\vec{c}$, is given by

$$\vec{c} = \frac{\sum_{i=1}^{\mathbf{C}} \vec{c_i}\,\exp(-\|\vec{p}-\vec{\mathcal{N}_i}\|^2)}{\sum_{i=1}^{\mathbf{C}} \exp(-\|\vec{p}-\vec{\mathcal{N}_i}\|^2)}$$

where $\vec{c_i}$ is the conditional probability vector associated with node $\mathcal{N}_i$, and the denominator expresses the fact that this equation is a "sum of unity" sparse data interpolation scheme. Since each node represents a pure class probability, all of the elements of its associated $\vec{c_i}$ are 0 except the ith entry, which is 1. Therefore, the conditional probability for each element of $\vec{c}$ reduces to

$$\vec{c}[i] = \frac{\exp(-\|\vec{p}-\vec{\mathcal{N}_i}\|^2)}{\sum_{j=1}^{\mathbf{C}} \exp(-\|\vec{p}-\vec{\mathcal{N}_j}\|^2)} \quad (14)$$

That is, the ith element of $\vec{c}$ is simply the normalized interpolation kernel weight associated with $\vec{\mathcal{N}_i}$. As suggested in Section 4.2, the minimum decision boundary distance discriminant (Equation 8) is perhaps a better quantity for transfer function color mapping. In this case, the denominator in Equation 14 cancels and the expression for the elements of $\Lambda(\vec{x})$ becomes the maximum over all $j \neq i$,

$$\lambda_i(\vec{x}) = \log(\mu_j) - \|\vec{p}-\vec{\mathcal{N}_j}\|^2 - \log(\mu_i) + \|\vec{p}-\vec{\mathcal{N}_i}\|^2 \quad (15)$$

where $\vec{p} = T\left([P(\omega_1|\vec{x}),\ldots,P(\omega_\mathbf{C}|\vec{x})]^T\right)$, and note that $-\|\vec{p}-\vec{\mathcal{N}_j}\|^2 \equiv \log(P(\omega_j|\vec{x}))$.

For each sample in our dataset, identified by its feature vector $\vec{x}_i$, the associated vector of class conditional probabilities, $\vec{c}_i = [P(\omega_1|\vec{x}_i), \ldots, P(\omega_{\mathbf{C}}|\vec{x}_i)]^T$, is parameterized, or mapped under $T(\vec{c})$, into our new space as the point, $\vec{p}_i$, that minimizes

$$\mathcal{E}(\vec{p}_i) = \|\vec{c}_i - T^{-1}(\vec{p}_i)\|$$

Unfortunately, since we are using non-compact basis functions, when a class probability approaches 1, the $\vec{p}$ vectors tend to infinity. This is due the fact that the Gaussian basis functions are never actually zero. To accommodate this we apply an affine transformation to the elements of $T^{-1}(\vec{p}_i)$ that ramps smoothly zero as the values approach some threshold ϵ. This epsilon value is the reciprocal of the maximum importance weight μ_i that our system allows; empirically, a $\mu_{\max} = 200$ is sufficient to make the minimization well behaved. The affine transformation has no effect on the placement of the decision boundaries, and tends to push error in the transformation out to the extremely low class probabilities, *i.e.* $P(\omega_i|\vec{x}) < \epsilon \rightarrow 0$.

This mapping can alternatively be thought of a reparameterization of the data-space that allows us to trivially classify the data using normalized Gaussian distributions with means equal to the class node centers. That is, the data samples are arranged in the new space so they are, by construction, normally distributed based on the feature classes.

6 Implementation

Our implementation of this work found that decoupling classification and transfer function color mapping not only improved the flexibility of our visualization system, but also dramatically simplified its construction. Our system naturally breaks up into several components: slicing and probing, classification and segmentation, GDR encoding, and visualization. Whenever possible, we leveraged existing tools and libraries to speed the development and prototyping of application specific variants of our system.

6.1 Slicing and Probing

The first step in the visualization of data using our system is the inspection of the raw data on a slice by slice basis. Our slicing tool's interface is modeled after a user interface commonly used for medical data. There are three slice views, one for each axis of the 3D data, and mouse clicks in one window automatically update the slice positions in the other two. This tool also provides window and level contrast settings as well as simplistic coloring of multi-variate data. The main function of this tool is to provide the ability to do feature/class selection and training set generation. This is done by probing locations in the data, on the slice views, where a class is present. These probe locations can then be used to estimate classification parameters, or as training data for non-parametric classifiers, or as seed points for segmentation.

6.2 Classification and Segmentation

While we have developed several specialized classification algorithms of our own, we rely heavily on a collaborative project aimed at the development of open source algorithms for image registration, classification, and segmentation called the Insight Toolkit (ITK) [10]. The designers of this toolkit were careful to make a strong distinction between class conditional probability estimation and decision rules with respect to the statistical classifiers it supports, which makes the specialization of classification and segmentation algorithms for the purpose of visualizing class conditional probabilities very convenient.

6.3 GDR Encoding

The implementation of the Graph-based Dimensionality Reduction scheme represented the bulk of our development effort. Even so, the library only consists of approximately 300 lines of code. We developed the library to be generic with respect to dimension, so it naturally supports encodings with any target dimension. A force directed graph solver naturally lends itself to least-squared, implicit solutions. However, given the relatively low number of nodes that we need to layout (typically between ten and one hundred), we found that a time dependent explicit solver performs quite well. The advantage of using an explicit solver is in the simplicity of its implementation. The disadvantage is that explicit solvers can tend to get stuck in local minima, which can be resolved using simulated annealing randomization. Our solver is iterative, with an adaptive timestep proportional to the maximum force over all nodes. Our solver begins by initializing all class nodes to random positions. At each timestep the class node positions are updated by

$$\vec{\mathcal{N}_i}' = \vec{\mathcal{N}_i} + \frac{\Delta t}{1 + \alpha \max_k \sum_{j=1}^{N} \varphi(\vec{\mathcal{N}_k}, \vec{\mathcal{N}_j})} \sum_{j=1}^{N} \varphi(\vec{\mathcal{N}_i}, \vec{\mathcal{N}_j})$$

where α is a scale term, in our system we choose $\Delta t = 1$ and $\alpha = 10$. All higher-order variance nodes are constrained to the average position of the class nodes whose variance they represent, therefore after each time step, we update the positions of these nodes accordingly. The iteration proceeds until the energy function $\mathcal{E}(\mathcal{N}_2, \ldots, \mathcal{N}_C)$ is no longer decreasing. We then record the graph configuration and the value of the energy function, and randomize several of the class node positions and minimize the new configuration. We perform this process of minimization and randomization several times, generally 10, and return the configuration with minimal energy.

The encoding step, $\vec{p} = T(\vec{c})$, is also expressed as a minimization. This too, can be implemented as an iterative solver. We accomplish this by first selecting an initial $\vec{p}$ as

$$\vec{p} = \sum_{i=1}^{\mathbf{C}} \vec{\mathcal{N}_i}\, \vec{c}[i]$$

The update step is

$$\vec{p}\,' = \vec{p} + \sum_{i=1}^{\mathbf{C}} \vec{\mathcal{N}_i} \left(\frac{\vec{c}[i] - \mathcal{A}(T^{-1}(\vec{p})[i])}{s} \right)$$

where $\mathcal{A}()$ is an affine transformation mapping the range $[\epsilon, 1] \rightarrow [0, 1]$, and s is a scale term proportional to the iteration number. Empirically, we have found that $s = (1+n)/2$ gives excellent results, where n is the iteration number. Because our inverse mapping, T^{-1}, is smooth, this optimization converges quickly. Five iterations is generally enough to achieve an acceptable RMS error, ideally this error should be approximately ϵ. For instance, the 4D GDR encoding of the BrainWeb Phantom classified data achieves an RMS error of .0004, with an $\epsilon = .0003$.

6.4 Visualization

Our rendering system is a simple single pass hardware ray caster. When the number of classes being visualized is five or less, we do not require a GDR encoding of the posterior probabilities. Since we know that $P(\omega_c|\vec{x}) = 1 - \sum_{i=1}^{\mathbf{C}-1} P(\omega_i|\vec{x})$, we can simply use a 4D dataspace (RGBA texture) for four classes, and easily derive the fifth class's probability. When using GDR encoded data, the first step in rendering a sample is decoding. Recall that since we want the minimum decision boundary distance, $\vec{\Lambda}(\vec{x})$, we do not need to actually decode the class conditional probabilities (see Equation 15). The algorithm for computing $\vec{\Lambda}$ from GDR encoded data can be computed efficiently by looping over the classes twice. The first loop computes $\log(\mu_i) + \log(P(\omega_i|\vec{x}))$ and saves off the maximum and second maximum of these values. The second loop completes the computation of $\vec{\Lambda} \equiv \mathrm{L}$ by subtracting the respective $\log(\mu_i)+\log(P(\omega_i|\vec{x}))$ from the maximum of these values, unless this term is already the maximum, in which case we use the second maximum. This small optimization converts the computation of $\vec{\Lambda}$ from an $\mathcal{O}(\mathbf{C}^2)$ to an $\mathcal{O}(\mathbf{C})$ algorithm.

Once the $\vec{\Lambda}(\vec{x})$ vector has been computed, the transfer function can be evaluated as $\mathbf{C}$ separable 1D opacity functions, or lookups. The domain of these 1D transfer functions is $[-\log(\mu_{\max}), \log(\mu_{\max})]$, where negative values indicate that the class is the minimum risk class and 0 is the decision boundary.

7 Results and Discussion

Color mapping based on the decision boundary distance term has a number of advantages. Unlike the raw class conditional probabilities, this term takes into account the relationships between the class probabilities for each sample and a user defined importance for each class. This means that the transfer function can be evaluated for each class independently, *i.e.* we need only design a simple 1D transfer function for each class. Furthermore, thanks to its well-defined behavior, we can define transfer functions based on decision boundary distance in advance, and apply them to classes as effects or profiles. For instance, if we desire a surface-like rendering, the opacity function for a particular class is simply a dirac delta centered at 0. This can be implemented robustly using a single preintegrated isosurface lookup table, which can be used for all classes that are to be rendered in this way. Alternatively, this can also be done by detecting a $\lambda_i(\vec{x})$ zero-crossing between two adjacent samples along the viewing ray, an example of risk-surfaces rendered using this method can be seen in Figure 7. For two sided risk-surfaces, we need only move the delta function to a slightly negative $\lambda_i(\vec{x})$ value, so that each class' risk-surface appears at a slightly different position than those who share that boundary. If we desire a more traditional fuzzy rendering, a suitable opacity function can be any monotonically decreasing function based on $-\lambda_i(\vec{x})$.

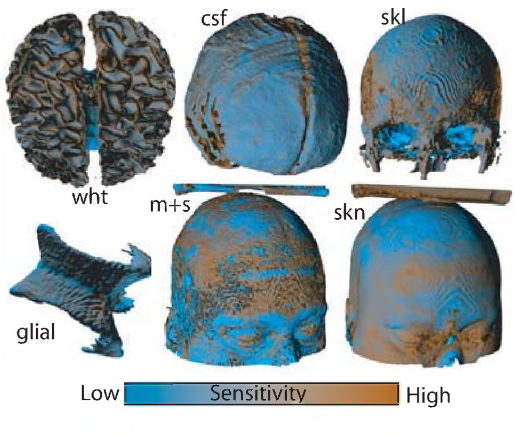

Figure 7: Selected risk-surfaces from the classified BrainWeb Phantom "fuzzy data", color mapped based on sensitivity (a measure of uncertainty in the decision boundary position). The data used for the renderings is a 4D GDR encoding of ten material classes.

Often, when probabilistic data is presented graphically, it is also shown with error bars indicating some confidence interval or sensitivity. We can create a kind of 3D analogue by displaying multiple risk-surfaces for a class at once, by varying μ_i. These concentric risk-surfaces provide a visual indication of the *sensitivity* of the boundary test. That is, it shows how the boundary would change if some class ω_i was μ_i times more likely. When the contours are packed closely together, we see that the decision boundary is well defined. When the contours are spread out, we see that the boundary is sensitive to small changes in class likelihoods, indicating that the exact placement of the decision boundary is less reliable. We can also derive a local measure of sensitivity that can be used for coloring the risk surfaces to highlight regions where the boundary placement is less certain. The measure we use is $s(\vec{x}) = 1/\|\nabla\lambda_i(\vec{x})\|$, larger values of $s(\vec{x})$ indicate a higher sensitivity in the spatial boundary location with respect to small changes in μ_i. Confidence intervals are another way of generating risk-boundary error bars. Confidence intervals can be measured in terms of the volume enclosed by a decision boundary with respect to μ_i. We compute confidence intervals by generating a histogram for the range of μ_i from $[1/\mu_{\max}, \mu_{\max}]$, where each bin is simply the number of samples in the dataset with a negative $\lambda_i(\vec{x})$ values given the corresponding μ_i. The 95% confidence interval is the decision boundary for the μ_i whose volume histogram count is 95% of the $\mu_{\max}$ histogram bin count. Figure 8 shows examples of each of these methods as well as a more traditional approach to color and opacity specification. Notice that the fuzzy method ramps color to black as λ approaches 0. This gives us yet another visual indication of uncertainty, when the black boundary is thick, the placement of the decision surface in this region is less certain.

Because the reparameterized dataspace is an *encoding*, transformations to the data, such as scaling and bias or quantizing, must also be applied to the node centers. We have found this to be quite easy if we simply encode the centers as part of the dataset, for instance appended to the end of the file or setting the first few samples to be the (parameterized) node centers. Of course, this technique is fragile with respect to spatial transformations, such as resampling and cropping.

8 Conclusion and Future Work

This paper describes a key way in which domain specific classification and segmentation can be integrated with state of the art volume visualization techniques. By decoupling classification and color mapping, classification can be accomplished independently of color mapping, allowing application specific solutions to evolve without concern for the current limitations of transfer function-based volume rendering. The transfer function interface also is dramatically simplified, not only is the feature space broken down into independent components, but these components have semantic meaning to the user. We show that deferring the decision step of the classification pipeline until render-time can provide the user with the ability to manipulate the decision making process and investigate uncertainty in the classification. We also address the increase in data set size, due to the need to store each class' probabilities independently, by developing a data dimensionality reduction technique specifically designed to accurately encode probabilistic image data and efficiently decode after resampling during the rendering phase of the visualization pipeline.

9 Acknowledgements

We wish to acknowledge Randal Frank, Mark Duchaineau, and Valerio Pascucci for providing substantial support and

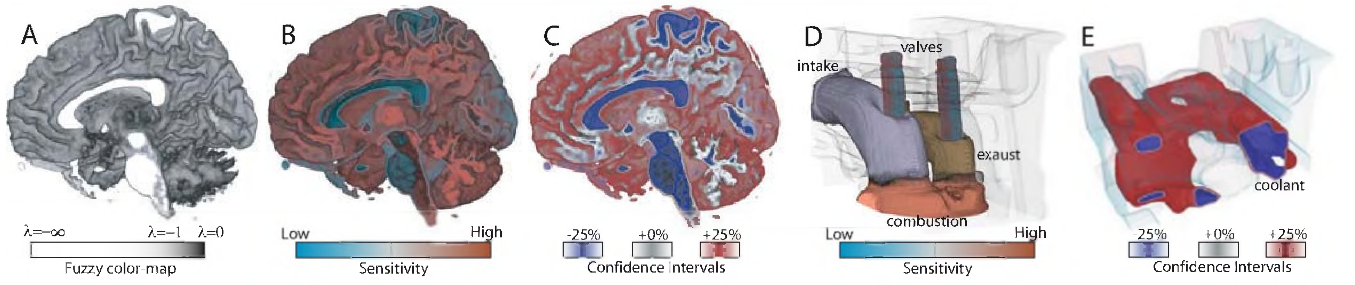

Figure 8: A comparison of rendering profiles. Subfigure A shows a "fuzzy" volume rendering color-mapped based on lambda for white matter. Subfigure B shows the risk-surface (lambda equal 0) for white matter color-mapped based on sensitivity (change in boundary position per unit change in importance). Subfigure C shows confidence intervals based on percent change in importance. Subfigures D and E show various features classified/segmented in the engine dataset. D shows sensitivity color mapping applied to the valve guides. E shows the coolant chamber of the engine rendered using confidence intervals.

insight in the development of this research, and Aaron Lefohn for early feedback and last minute writing. We would also like to thank David Laidlaw and Ross Whitaker for many helpful discussions relating to this work over the last several years, and Ryan White for help navigating the world of computer vision. This work has been supported by the DOE HPCS Graduate Fellowship, DOE-CSAFE, DOE-VIEWS, and DOE-ASC ASAP.

References

[1] C. Bajaj, V. Pascucci, and D. Schikore. The Contour Spectrum. In *IEEE Visualization 1997*, pages 167–173, 1997.

[2] 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.

[3] C. Cocosco, V. Kollokian, R. Kwan, and A. Evens. Brainweb: Online interface to a 3d mri brain database. In *NeuroImage*, http://www.bic.mni.mcgill.ca/brainweb/, 1997.

[4] R. A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. *Computer Graphics (Proceedings of SIGGRAPH 88)*, 22(4):65–74, August 1988.

[5] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, 2000.

[6] K. Engel, M. Kraus, and T. Ertl. High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading. In *Eurographics / SIGGRAPH Workshop on Graphics Hardware '01*, Annual Conference Series, pages 9–16. Addison-Wesley Publishing Company, Inc., 2001.

[7] L. Grady. Multilabel random walker image segmentation using prior models. In *CVPR 2005*, 2005 (Accepted).

[8] G. Grigoryan and P. Rheingans. Point-based probabilistic surfaces to show surface uncertainty. *TVCG*, 5(10):564–573, 2004.

[9] M. Hadwiger, C. Berger, and H. Hauser. High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. In *IEEE Visualization 2003*, 2003.

[10] L. Ibanez, W. Schroeder, L. Ng, and J. Cates. *The ITK Software Guide: The Insight Segmentation and Registration Toolkit*. Kitware, Inc, 2003.

[11] T. Iwata, K. Saito, N. Ueda, S. Stromsten, T. L. Griffiths, and J. B. Tenenbaum. Parametric embedding for class visualization. In *Advances in Neural Information Processing Systems 17*, pages 617–624. 2005.

[12] C. Johnson. Top scientific visualization research problems. *IEEE Computer Graphics and Applications*, pages 13–17, 2004.

[13] A. Kaufman and K. Mueller. *The Visualization Handbook*, chapter Overview of Volume Rendering. Elsevier, 2004.

[14] G. Kindlmann. Semi-automatic generation of transfer functions for direct volume rendering. Master's thesis, Cornell University, 1999.

[15] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE TVCG*, 8(3):270–285, jul - sep 2002.

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

[17] M. Levoy. A hybrid ray tracer for rendering polygon and volume data. *IEEE Computer Graphics & Applications*, 10(2):33–40, March 1990.

[18] S. Roweis and L. Saul. Nonlinear dimensionality reduction by local linear embedding. *Science*, 290:2323–2326, Dec 2000.

[19] L. Sobierajski and A. Kaufman. Volumetric ray tracing. In *Symposium on Volume Visualization*, pages 11–18, October 1994.

[20] D. Stalling, M. Zoeckler, and H.C. Hege. Segmentation of 3d medical images with subvoxel accuracy. In *Computer Assisted Radiology and Surgery*, pages 137–142, 1998.

[21] T. Tasdizen, S. Awate, R. Whitaker, and N. Foster. Mri tissue classification with neighborhood statistics: A non-parametric, entropy-minimizing approach. In *MICCAI 2005*, 2005 (Accepted).

[22] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, Dec 2000.

[23] U. Tiede, T. Schiemann, and K. H. Hohne. High quality rendering of attributed volume data. In *IEEE Visualization 1999*, pages 255–262, 1999.

[24] F. Tzeng, E. Lum, and K. Ma. An intelligent system approach to higher dimensional classification of volume data. *IEEE TVCG*, 11(3):273–284, may - jun 2005.

[25] F. Tzeng and K. Ma. A cluster-space visual interface for arbitrary dimensional classification of volume data. In *IEEE TCVG Symposium on Visualization*, 2004.

[26] I. Viola, A. Kanitsar, and M. E. Gröller. Importance-driven volume rendering. In *Proceedings of IEEE Visualization'04*, pages 139–145, 2004.

[27] C. M. Wittenbrink, A. Pang, and S. Lodha. Glyphs for visualizing uncertainty in vector fields. *IEEE TVCG*, 2(3):266–279, September 1996.