

# An Intelligent System Approach to Higher-Dimensional Classification of Volume Data

Fan-Yin Tzeng, Eric B. Lum, *Member, IEEE*, and Kwan-Liu Ma, *Senior Member, IEEE*

**Abstract**—In volume data visualization, the classification step is used to determine voxel visibility and is usually carried out through the interactive editing of a transfer function that defines a mapping between voxel value and color/opacity. This approach is limited by the difficulties in working effectively in the transfer function space beyond two dimensions. We present a new approach to the volume classification problem which couples machine learning and a painting metaphor to allow more sophisticated classification in an intuitive manner. The user works in the volume data space by directly painting on sample slices of the volume and the painted voxels are used in an iterative training process. The trained system can then classify the entire volume. Both classification and rendering can be hardware accelerated, providing immediate visual feedback as painting progresses. Such an intelligent system approach enables the user to perform classification in a much higher dimensional space without explicitly specifying the mapping for every dimension used. Furthermore, the trained system for one data set may be reused to classify other data sets with similar characteristics.

**Index Terms**—User interface design, classification, transfer functions, graphics hardware, visualization, volume rendering, machine learning.

## 1 INTRODUCTION

VOLUME rendering has become an important tool for many visualization applications. Visualizing volume data consists of two key steps: classification and rendering. In the classification step, the task is to define a mapping between data values and the corresponding colors and opacities, usually through the specification of a transfer function. The rendering step assigns color and opacity to each voxel according to this mapping and then projects the voxels according to the view setting to produce an image that appropriately displays the features of interest in the data.

While constructing a 1D (one input) or 2D (two input) transfer function is a manageable task for average users, the interface can be unintuitive since the user must work in the derived transfer function space, shown in the bottom left and bottom middle of Fig. 1.

In addition, the traditional transfer function is of limited effectiveness in performing the actual classification. For example, for the MRI head data set used in Fig. 1, it is difficult to use a 1D transfer function to differentiate the brain and the region near the skull since the scalar values of the two materials are similar. As such, both materials must be shown together, in which case, the outer layer might obscure the brain material of interest when rendered in 3D. The user could reduce the opacity of the outer layer to make the brain more visible, but the brain would simultaneously become more transparent and difficult to see. A transfer function that is a compromise between the two may be found through an iterative process of trial and error. An example is the upper left image in Fig. 1, which has some

material obscuring the brain as well as a slight transparency of the brain.

Higher-dimensional classification can lead to better results since it uses more properties for each voxel, such as gradient information, its location, and local texture. With previous methods, however, the user specifies the classification in transfer function space directly, limiting the number of data properties that can be used simultaneously. An example of a visualization of the brain using a 2D transfer function and its interface are shown in the center of Fig. 1. The additional gradient information used in constructing the transfer function allows more refined classification and works well when the user intends to visualize the boundaries between different materials. In many cases, classification can be improved even further by using transfer functions that extend beyond 2D or 3D. Finding a method for specifying these transfer functions in an intuitive and efficient manner remains an important challenge in making volume visualization a more attractive tool for understanding data.

We present a new approach to the volume classification problem, relying on an intelligent system to abstract high-dimensional mapping functions from the user. The user's interaction consists of specifying regions of interest by painting on a small number of slices from the volume data. The user is given full control of what materials they want to classify by applying one color of paint to parts of the volume they are interested in and another color of paint to regions they do not want to include. The intelligent system employs a machine learning method using the painted regions as training data to "learn" a classification function. We have conducted a study of using such an approach to higher-dimensional volume classification by experimenting with using a variety of data properties as inputs, such as scalar value, scalar gradient magnitude, location, and neighboring values. We compare two different machine

• The authors are with the Institute for Data Analysis and Visualization (IDAV), Department of Computer Science, University of California, Davis, CA 95616-8562. E-mail: {tzeng, lume, ma}@cs.ucdavis.edu.

Manuscript received 2 Oct. 2004; revised 13 Dec. 2004; accepted 23 Dec. 2004; published online 10 Mar. 2005.

For information on obtaining reprints of this article, please send e-mail to: [tcvg@computer.org](mailto:tcvg@computer.org), and reference IEEECS Log Number TVCG-0122-1004.

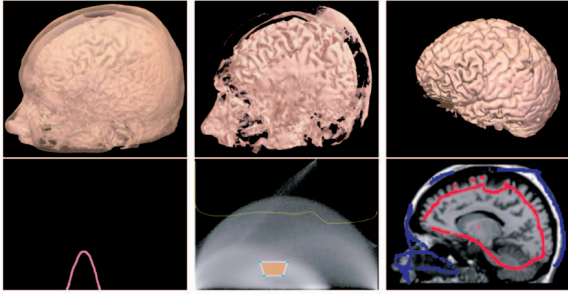


Fig. 1. Left: Volume rendering of an MRI head data set with a 1D transfer function. Middle: Using a 2D transfer function. Right: Using a 10D classification function.

learning methods: neural network and support vector machines, and demonstrate them with several data sets. We show that high-dimensional classification functions can better differentiate volumetric materials for visualization, achieving sophisticated classifications that would otherwise be impossible in traditional lower-dimensional transfer function spaces.

We also exploit the programmability of modern graphics hardware to accelerate all stages of the volume visualization process, including the implementation of a hardware-accelerated, machine-learning-driven volume renderer. As a result, the user is able to paint on the volume, immediately see the results of the classifier's training, and continue to paint to provide additional training data to steer the machine learning model toward achieving a classification function that meets the user's visualization objective. Finally, we show that a trained system is potentially reusable. For example, in the context of medical imaging, such a system can be trained to classify data sets from different scans of the same patient or even different patients. This capability is very desirable since the training can capture an expert's knowledge about the specific type of data as well as the classification task.

## 2 RELATED WORK

There has been a great deal of research devoted to the generation of transfer functions for volume visualization [23]. Fujishiro et al. [7] use topological information from a hyper-Reed graph to derive transfer functions. Bajaj et al. [1] present techniques for capturing isosurfaces of interest. He et al. [11] use genetic algorithms to breed trial transfer functions. The user can either select functions from generated images or allow the system to be fully automated. Marks et al. [19] address the parameter selection problem in general by rendering a multidimensional space of those parameters. The user then navigates this space, in the context of volume visualization, to choose appropriate transfer functions. Jankun-Kelly and Ma [14] present automated methods for generating transfer functions to visualize time-varying volume data.

Levoy [18] shows how to use gradient magnitude to enhance material boundaries in volume data. König and Gröller [17] introduce a user-interface paradigm with a set of specification tools assisted with real-time volume rendering to make it easier for the user to select transfer

functions. Kindlmann and Durkin [15] suggest that, by looking at a two-dimensional scatter plot of data values and gradient magnitudes, opacity transfer functions can be easily defined. Such transfer functions can effectively capture boundary features between materials of relatively constant data value. Kniss et al. [16] extend this work by introducing a set of direct manipulation widgets as the interface for defining multidimensional transfer functions for volume visualization. The concept of dual-domain (i.e., the volume data space and the transfer function space) interaction they describe allows the user to specify regions in volume space and immediately see the resulting regions in transfer function space. This interaction helps incorporate the user's spatial understanding of the volume data into the transfer function space. Huang and Ma [13] present a technique that can suggest a 2D transfer function by using the results of partial region growing from a point selected in volume space. Our technique eliminates the transfer function space entirely, replacing it with a volume space painting interface.

We employ machine learning techniques coupled with an interactive user interface and a hardware-accelerated volume renderer for classification and visualization of biomedical volume data. Artificial neural networks have received a great deal of attention in the field of biomedical imaging, especially to assist in image segmentation tasks [9], [10], [22]. Support vector machines [3], [5] have been used in the applications of pattern recognition including object recognition [2], text categorization [26], and face detection [21].

The work presented in this paper extends our previous research [27] in the following ways. First, to demonstrate that our method is flexible and can be used with machine learning classifiers other than neural networks, we have implemented a support vector machines (SVMs) classifier. We have compared its performance to that of a neural network and have found that using SVMs typically leads to better classification results. Second, we have performed a study of this intelligent system approach for a variety of volume data classification tasks. Third, we have also investigated information reuse by training a neural network or SVMs model with one data set and applying it to a new data set with similar characteristics. Fourth, we have extended the description of our hardware neural network implementation. Finally, we conclude our work with a discussion of the pros and cons of such an intelligent system approach and provide pointers for future work directions. This paper thus presents a more complete treatment of the intelligent system approach to the volume classification problem.

## 3 AN INTELLIGENT SYSTEM APPROACH

In traditional volume rendering interfaces, the user requires a certain level of understanding of the intensity distribution of a data set to make an acceptable transfer function. When a new data set is used, a great deal of time might be spent experimenting with different mapping functions for that data set.

The use of an intelligent system approach allows for the application of high-dimensional classification functions

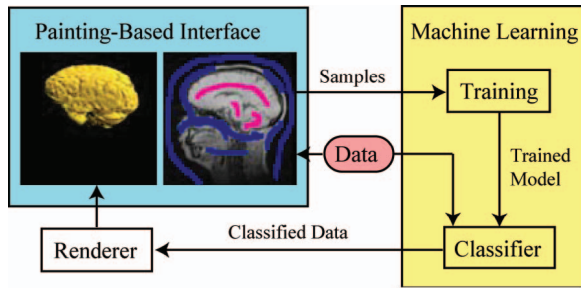


Fig. 2. The visualization process. The painting user interface gives the user direct access to the volume data and allows the user to indicate the region of interest. “Samples” are the points in painted regions and are the inputs to the machine learning technique. By applying the trained machine learning model to a volume, the user can classify and then render the volume. The resulting visualization is shown to the user so that paint can be added or removed to refine the classification until a satisfactory result is obtained.

while freeing the user from operating the visualization system in a derived transfer function domain. In our study, we found that a painting user interface provides a simple and intuitive means of specifying regions of interest. Fig. 2 shows the overall process of our interactive classification and visualization method. The portion most visible to the user is the painting-based interface used to collect sample points of the region of interest. The user starts by painting sample points on a slice of the volume. These painted sample points are fed to the intelligent system, where training begins to produce the classifier.

Training is an iterative process, with the user able to interactively view the classification results by applying the current AI system for classifying individual slices or the entire volume in real-time. The user can use this feedback to further revise the painting and add additional training data so that the resulting system leads to a more desirable classification.

#### 4 A PAINTING-BASED INTELLIGENT USER INTERFACE

The goal of our user interface is to provide a means for the user to partition a data set into different material classes. The user specifies membership into a material class by painting on slicing planes using two different colors. One color is applied to indicate example regions that are part of the material class, while the other color is used to indicate regions that are not part of the material class. For example, the user might apply red paint to the brain region of a slice of an MRI scan to indicate it is a material of interest and blue to other regions to indicate otherwise. The user is also provided with a set of familiar painting user-interface tools which include brushes and erasers of varying sizes. There are x, y, and z axis-aligned slicers for the user to view and paint on the volume, as shown in Fig. 3.

The user does not need to paint on all slices of the volume, but instead, must paint on some areas of several slices to classify the entire volume. The required number of painted voxels varies depending on the data set. When using position information for the classification, the painted voxels should be spread out across different slices in order to provide general samples for training. Thus, real-time visual feedback

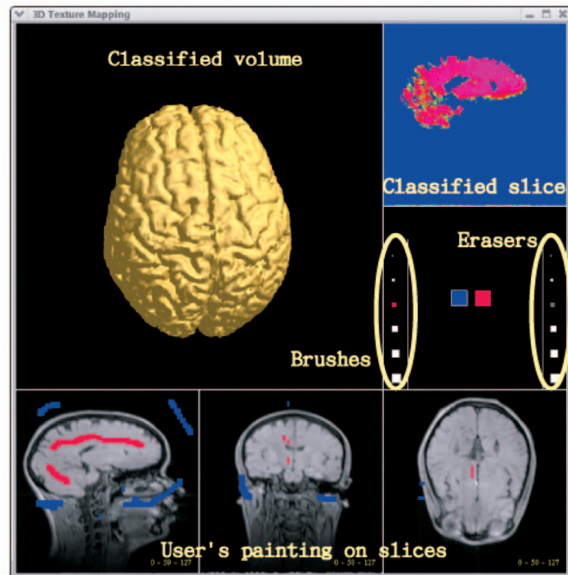


Fig. 3. The painting-based intelligent user interface.

must be provided such that the user can look at the resulting slices or volume to find the regions that are not well-classified and quickly go to the corresponding slices to paint more sample points to improve the classification.

When painting on a slice, the corresponding painted regions are also shown in the other two slicers, with all painted sample points recorded in a table that is used to train the neural network. For scalar data, the sample point data includes the scalar value of the voxel, gradient magnitude, the scalar values of its neighbors (up, down, left, right, front, and back), the position of this voxel, and an output value that indicates membership into a material class. For color data, the sample point data includes the R, G, B values of the voxel, values of its neighbors, and position. Therefore, when a user chooses to visualize a color data considering six neighbors, the dimensionality of its classification function is  $3 + (6 \times 3) + 3 = 24$ .

In Fig. 4, the progression of a session of a user employing our technique is shown. The left image of the top row shows a slice painted with pink representing the area the user wants to see and blue representing the area the user does not want to see. The middle image shows the result of the color-coded classification by a machine learning model. As indicated by the color bar, if a pixel’s color is closer to blue, the pixel is less likely to be part of the material of interest. If it is closer to pink, the pixel is more likely to be part of the material of interest. When the user only paints on the empty region and the brain, the classified slice shows the brain in pink, which indicates that the voxels in this region have very similar characteristics to the regions the user painted. The other regions of the head are shown in red to green since the user has not given input to classify them. The right image is a volume rendering for this classification.

When the user obtains a result containing areas that are not well-classified, more painting is required. The middle row of Fig. 4 shows a painted slice, the classified slice, and the classified volume rendering after more paint has been applied. Most materials, except the top of the head and



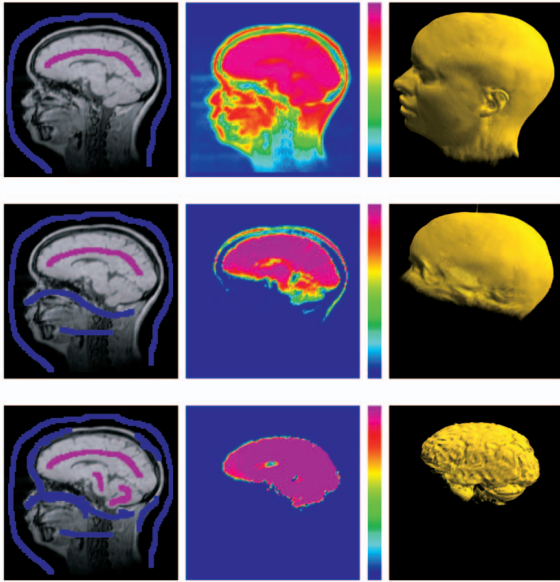


Fig. 4. Top: A slice painted by a user where pink represents the material the user would like to see and blue represents undesirable materials, the result of classification with a color bar to its right, and the rendered result of the classified volume. Middle: The results after more painting information has been added. From left-to-right: A painted slice, the classified slice, and the classified volume. Most materials except the top of the head and brain have been removed. Bottom: Additional paint is added to remove all regions except the brain. From left-to-right: A painted slice, the classified slice, and the classified volume.

brain, have been removed. By continuing to paint on the top of the head, as shown in the bottom row of Fig. 4, the brain is the only material remaining, with all other regions removed.

## 5 INTELLIGENT-SYSTEM-ASSISTED CLASSIFICATION

With our method, the user interacts with materials in a volume directly. The underlying classification is hidden from the user through the use of machine learning.

Any supervised machine learning technique, such as artificial neural networks [28], support vector machines [12], Bayesian networks [6], or hidden Markov models [20], can fit into this framework as long as the technique is able to perform classification after learning from the user's high-dimensional inputs.

When using a machine learning classifier, there are two steps involved. First, the classifier is trained with pairs of inputs and desired outputs. The input in our design is a set of vectors, where each vector contains information about a painted voxel such as the voxel value, gradient magnitude, position, and neighboring voxel values. The desired output is the corresponding class information, which is assigned to one if it is part of the material class and zero if it is not.

After training, a classifier is created to determine which regions in the volume are members of a particular material class. For each voxel, an input vector containing the same information used for training is then passed to the classifier to obtain an output value between zero and one, which indicates the likelihood that the voxel belongs to the material class.

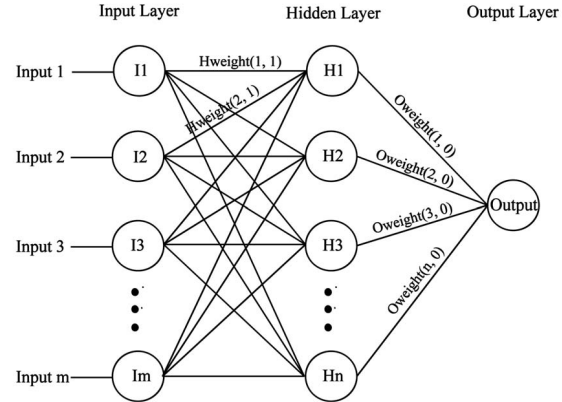


Fig. 5. Structure of an artificial neural network with  $m$  inputs,  $n$  hidden nodes, and one output.

The size of the input vectors has a strong influence on the time required for classification to occur. Therefore, the user is given the option to either use all the input vectors or subsequently remove those dimensions which do not contribute to the classification results.

The various machine learning algorithms that can be used for classification have different trade-offs in performance, accuracy, and complexity. We employ neural networks and support vector machines as the machine learning classifiers. Neural networks are a well-established technique that have been shown to be well-suited for a wide variety of tasks, while support vector machines are a newer method that has become increasingly popular in recent years.

### 5.1 Artificial Neural Network

Fig. 5 shows the structure of an artificial neural network. Each connection between neurons has a weight, with the weights modulating the value across the connection. Training is the process of modifying the weights until the network implements a desired function. To train a network, a set of training inputs and desired outputs are required. At the beginning, the weights are set at random and are iteratively modified to obtain a network which minimizes the error at the output for the training data. Once training has occurred, the network can be applied to data that was not part of the training set.

The neural network topology we use is three-layer perceptron and it is trained with the Feed-Forward Back-Propagation Network (BPN) algorithm. The back-propagation algorithm, which is designed for supervised training, was introduced by Werbos [28] and has been widely used since the work of Rumelhart and McClelland [24].

A back-propagation neural network consists of at least three layers: an input layer, at least one hidden layer, and an output layer. The structure of a neural network is shown in Fig. 5. In this example, there are  $m$  inputs in the input layer,  $n$  hidden nodes in the hidden layer, and one output in the output layer. Neurons are connected in a feed-forward fashion and every neuron in the input layer is connected to all neurons in the hidden layer. Similarly, hidden nodes are fully connected to the nodes in the output layer.

The input vector is propagated forward through the network, influenced by the weights of each connection

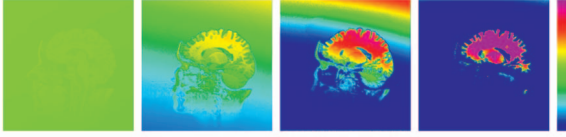


Fig. 6. The user can immediately see the result of the iterative refinement of the neural network. This image sequence shows the progression of training results over a three second period.

between neurons from different layers. The output of the back-propagation network is compared with the desired output and an error value is calculated. The error is back-propagated to the inputs and used to adjust the weights to better match the desired output. In order for a neural network to model nonlinear relationships between inputs and outputs, a nonlinear transformation is required. In our work, we use the standard sigmoid function, which can be expressed as  $f(x) = 1/(1 + e^{-x})$ .

In order to perform as much training as possible while maintaining interactivity, the incremental training process occurs in the system idle loop. After each training iteration, a classified slice is shown so the user can determine whether the network performs satisfactorily in that slice region or if more paint should be applied to supply additional training data.

In our work, the user can see training progress in real-time. Fig. 6 shows an example of the iterative refinement of neural network weights over a period of three seconds in one second increments when applied to the MRI Head data set. The image on the left shows the result of the network without any training, while the figure on the far right shows the solution after three seconds of training to isolate the brain region. If additional paint samples were added, the training algorithm would use this data to refine the previous network.

Classification of an entire volume can be performed in hardware during rendering, as described in Section 6, or in software. Applying the network in software consists of feeding each voxel and its neighboring properties into the network to get an uncertainty value between zero and one. This uncertainty value can then be stored in a new classified volume that can be rendered using traditional volume rendering methods in either software or hardware. The advantage of using software for classification is that it does not require any graphics hardware, the size of the network can be arbitrarily large, and classification is a preprocessing step that occurs once, rather than for each frame during rendering. The disadvantage of software is performance, where it can take up to 9 seconds to apply the network with 11 input nodes for 11-dimensional classification, 8 hidden nodes for classifying a  $256 \times 256 \times 256$  volume using a 2.8 GHz Pentium 4 computer, making it unsuitable for applications where the user must see the result of the trained network during training.

## 5.2 Support Vector Machines

In recent years, the support vector machines (SVMs) method has become increasingly popular for efficiently solving classification problems. SVMs is a newer classification technique that not only separates data into different

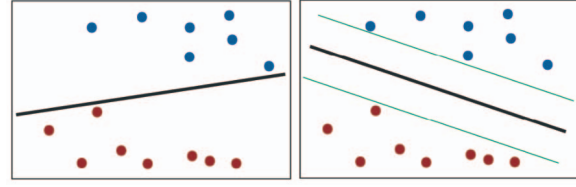


Fig. 7. This figure illustrates the maximal margin property of SVMs. If there are two classes to be classified, most machine learning algorithms find a separation to well classify the training data, as shown in the left image. SVMs in the right image, provide the maximal margin property that not only separates the training data, but has the potential to better classify the data which are not shown in the training data set.

classes, but leaves a maximal margin between those classes. Fig. 7 illustrates the maximal margin property. In the left image, an infinite number of lines could be used to separate the two classes shown in red and blue. The black line shown in the right image illustrates a separation with the maximum margin property, with the green lines illustrating the margins. Because of their maximum margin separation, SVMs can obtain better generalized classification results, which minimizes the risk of misclassification for data that is not shown in the training set.

When the given classes cannot be linearly separated in the original data space, SVMs map the data from the lower-dimensional data space to a higher-dimensional feature space where the classes can be linearly separated. This linear classification is then performed using hyperplanes, called “optimal separating hyperplanes,” that are placed at equal distances between the separated classes.

The nonlinear mapping to the higher-dimensional feature space is accomplished using kernel functions. Several different kernel functions have been developed, including linear, polynomial, radial basis function, and sigmoid kernels. A radial basis function kernel is widely used for general-purpose classification and is the one we use in our work. It typically requires fewer hyperparameters than other kernel functions, which improves runtime performance since the complexity of applying SVMs scales linearly with the number of hyperplanes used.

The linear classification performed by SVMs can be defined as follows: Given a set of  $N$ -dimensional training data,  $x_i \in R^n$ , and the corresponding class labels,  $y_i \in \{+1, -1\}$ , we can use a simple linear classifier

$$S = \{x | \langle w, x \rangle + b = 0\}$$

to separate two classes. The decision function becomes

$$f(x_{new}) = \text{sign}(\langle w, x_{new} \rangle + b).$$

Maximizing the margin between different classes is a problem of constrained optimization and the Lagrange method is used to solve the constrained optimization problem. With the Lagrange method, training data  $x_i$  are used as support vectors and determine the hyperplane when the Lagrange multiplier  $\alpha_i > 0$ . The decision function with support vectors (sv) becomes

$$f(x_{new}) = \text{sign} \left( \sum_{i=1}^{\#sv} \alpha_i y_i \langle x_i^{sv}, x_{new} \rangle + b \right).$$

TABLE 1

Using the Training Set Obtained Based on an NN Classifier





| Machine Learning Algorithm | NNs   | SVMs  |
|----------------------------|---|---|
| Number of Training Samples | 7988  | 7988  |
| User's Input Time          | 2-3 mins  | -   |
| Training Time              | 1.3 secs  | 0.74 secs   |
| Classifying the Volume     | 6.28 secs   | 55.47 secs  |
| Rendered Classified Volume |  |  |

TABLE 2

Using the Training Set Obtained Based on an SVMs Classifier

| Machine Learning Algorithm | NNs   | SVMs  |
|----------------------------|---|---|
| Number of Training Samples | 818   | 818   |
| User's Input Time          | -   | < 30 secs   |
| Training Time              | 1.41 secs   | 0.06 secs   |
| Classifying the Volume     | 6.47 secs   | 48.78 secs  |
| Rendered Classified Volume |  |  |

When a kernel function is applied, this becomes

$$f(x_{new}) = \text{sign} \left( \sum_{i=1}^{\#sv} \alpha_i y_i K(x_i, x_{new}) + b \right),$$

where, for the radial basis function kernel used in our work,

$$K(x_1, x_2) = \exp(-\gamma \|x_1 - x_2\|^2).$$

The error from classifying training data is used as the stopping criteria. When the error is higher than a pre-selected threshold, the hyperplane is refined to reduce the error iteratively until the threshold is reached.

### 5.3 Comparison between NNs and SVMs

#### 5.3.1 NNs versus SVMs in Theory

Both NNs and SVMs have the same goal of finding the hypothesis function to separate data into different classes that minimize errors in classification. Most machine learning methods, such as neural networks, are designed to minimize empirical risk, that is, to reduce the probability of misclassification within the training data. On the other hand, support vector machines are designed to minimize the structural risk so that the probability of misclassifying the unseen data is minimized to better approximate the classification function needed.

With NNs, training is done repeatedly to obtain better and better results for classification. In most cases, the neural network is continuously improved during the training process. It is difficult to determine when to stop training or when the best network is generated. SVMs handle the entire training data set simultaneously and provide the optimal solution when the training is complete.

A neural network trains and modifies itself by changing the weights on the connected edges. During this updating process, the solution can get stuck at local minima since the training data are fed into the network one by one and a subset of the training data can have stronger influence on the weights than the other subsets. SVMs, on the other hand, are guaranteed to a global minima solution for a classification task.

#### 5.3.2 NNs versus SVMs with Examples

Tables 1 and 2 show a comparison between using NNs and SVMs with our method when performing the same task of classifying the brain. For the results shown in Table 1, we added training data through the application of paint until the neural network provided the satisfactory classification

shown in the lower left. Table 2 shows the result of providing training data to perform the same task, but with the interactive feedback of the performance of a support vector machines classifier. The result of this classification is shown in the lower right. Notice that, in order to obtain similar classification results, we needed to provide far more training data for the NN compared to that for the SVMs since SVMs provide maximal margins between classes to better predict the unseen data. As shown in Table 1, 7,988 training sample points were required for the NN compared to the 818 training samples for the SVMs as shown in Table 2.

In these tables, "User's training time" refers to the time we spent training the program to learn the satisfying classification. It took two to three minutes to provide training data for the NN compared to less than 30 seconds for the SVMs.

"Training time" is the time used by each machine learning technique for training. It is difficult to compare the two methods since the choice of training parameters can lead to significant variations in performance. In this work, training is halted when the results look satisfactory for the NN and when the SVMs find the optimal solution. Under these circumstances, SVMs technique is noticeably faster than NN.

"Classifying the volume" shows the time needed to classify the entire volume after training. For NN, it depends on the network size. Therefore, the times shown for two tables with differing numbers of training samples are very close. For SVMs, the classifying time depends on the number of training samples, so the two tables show times that differ significantly.

## 6 HARDWARE ACCELERATION

An advantage of using a neural network classifier is that the structure of a neural network is more easily implemented in graphics hardware than that of support vector machines. The added performance of using graphics hardware allows for classification to occur fast enough that the user can evaluate the result of neural networks training on the entire volume during the application of paint. With this feedback, the user can steer the network to implement the desired classification function. The application of the network to a volume in software, on the other hand, limits this interactivity because of the sheer number of voxels typically involved.

The neural-network volume renderer is implemented using a fragment program, which permits the execution of



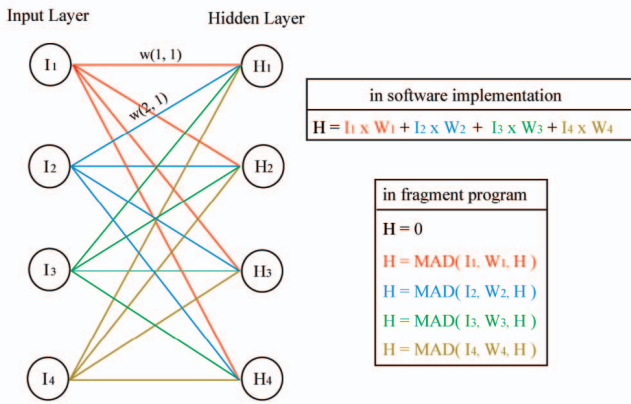


Fig. 8. Illustration of the hardware implementation for the neural network.

assembly language instructions that are run on a per-pixel basis during rasterization. Rendering uses the standard technique of drawing a stack of view-aligned polygons that sample a 3D texture [8], with the enhancement that a fragment program is used that implements the neural network as these polygons are rendered. We tested our implementation on an ATI Radeon 9700 Pro graphics card.

Our fragment program requires seven texture lookups for retrieving the center voxel values and its neighbors as inputs for the neural network. The position of a sample in 3D volume space is simply derived from the texture coordinates of the pixel. The weights of a neural network are a set of floating-point numbers and are passed to the fragment program as input parameters. The number of input parameters is limited; thus, for larger neural networks, the weights would need to be passed to the fragment program using a different mechanism. They could, for example, be stored in textures, which would add overhead from the redundant texture reads needed to fetch these weights each time the network is applied.

The same nearest neighbors used for neural network calculation are also used for estimating gradient for lighting. Thus, normal map textures are not required for lighting, which permits the storage of significantly larger volumes on the graphics card. The neural network itself can be implemented with a series of vectorized MAD (multiply add) instructions, with additional instructions to compute the sigmoid excitation function. Fig. 8 illustrates this process where the network has only four input nodes and four hidden nodes for simplicity. The hidden nodes  $H_m$  can be represented as

$$H_m = \sum_{n=1}^4 I_n \times w(n, m), m \in \{1, 2, 3, 4\},$$

where  $w(n, m)$  is the weight between input node  $n$  and hidden node  $m$ . The hidden nodes and weights are stored in vectors

$$H = (H_1, H_2, H_3, H_4), \text{ and}$$

$$W_n = (w(n, 1), w(n, 2), w(n, 3), w(n, 4)),$$

respectively. The value of  $H$  is computed with a series of MAD instructions, as shown in Fig. 8. After these operations, the sigmoid function is then applied to  $H$ , completing

one layer of computation. By repeating this procedure, larger networks and additional layers can be implemented in a fragment program.

Rendering performance is limited by the time required to rasterize the textured polygons to the screen, which is hampered by the necessary seven texture lookups and the length of the fragment program. The amount of data sent to the renderer for each frame consists only of the newest neural network weights. The rendering of a  $256 \times 256 \times 256$  volume to a  $512 \times 512$  window occurs at approximately 1.5 frames per second using one slice per voxel on the graphics card mentioned previously. Since the amount of data that must be sent to the renderer is low, rendering can be easily done asynchronously on a separate computer. The rendering PC queries the PC where painting and neural network training occurs for the most recent set of weights (under one kilobyte of data) for each frame.

## 7 CASE STUDIES

We present three case studies using the painting user interface combined with a machine learning approach. The data sets we used to test our system include a knee, chest, and cryosection color brain data set each resized to  $256^3$ , a  $128^3$  MRI head data set, a  $256 \times 256 \times 156$  MRI brain data set, and three  $256 \times 256 \times 128$  tumor data sets.

### 7.1 High-Dimensional Classification

For color data sets, the classification function is only used to assign opacity since each voxel has an RGB color associated with it. Assigning opacity by specifying a three-dimensional transfer function for this type of data is made difficult by the traditional 2D interface and display. Our method, however, is able to handle color data sets in exactly the same manner as scalar value data sets, where the user paints on 2D slices that pass through the volume. The information used in classification includes the RGB color values of a voxel, the colors found in the voxel's six neighbors, and its position. This gives up to 24 dimensions for our classification function.

An example of classifying the cryosection color data set is shown in Fig. 9. The left image is one of the photographic slices of the original data set. Each slice consists of the brain in the middle with the surrounding white ice and the table the brain was placed on. Some regions in each slice show gaps in the brain which reveal deeper regions of the data set that are not part of the current slice. To visualize the brain properly, it is necessary to remove the ice and surrounding regions, as well as the gaps in the brain, which often have a very similar color as the brain itself. Takanashi et al. [25] developed a method for specifying three-dimensional transfer functions that consists of transforming the RGB color values using independent component analysis into a derived ICA space that allows a transfer function to be specified as a series of 1D transfer functions aligned to each of the ICA axes. Their method simplifies the task of specifying RGB color transfer functions, but requires the user to work in a derived data space that is further from the original data. With our method, the brain can also be classified from its surrounding material, but the user is able

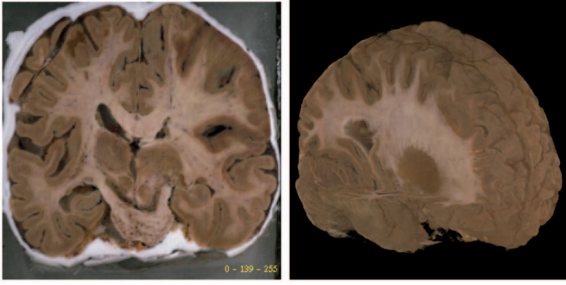


Fig. 9. The left image is a slice of the original cryosection data set and the right image is the classified result of the brain with a region cut away to reveal the inner structure.

to work directly with the data, avoiding the specification of a three-dimensional transfer function.

The right image is the result of the classified brain generated using the values of a voxel, six neighbors, and the position with 20 hidden nodes. Two cutting planes are applied to the volume so that the user can look at the inner structure. For color data sets, the voxel's luminance is used to calculate the gradient for shading.

## 7.2 Classifying Multiple Materials

The method described so far can be used to classify a single material in the volume. Often, it is desirable to specify more than one class in a volume. For example, a user might want to show more than one material at a time or a certain organ with high opacity value and other regions with low opacity to provide context. Our method can be easily extended to work with multiple material classes.

To classify more than one material at a time, we use multiple classifiers. First, we classify a material by the method described in the previous sections. When adding another material class, a new classifier is created and used. For the second material, we also follow the same procedure used for classifying the first material. Two classifiers are generated separately by two different groups of sample points. The results of all classifiers are used to render the classified volume.

Training is only performed on the newest material class. Therefore, when classifying a volume into multiple materials, the training time is the same as for single material classification. Rendering is accomplished using multiple passes, one for each material class.

We applied this technique to the first MRI head data set, which has brain material in the middle surrounded by materials of similar intensity near the skull. As shown in Fig. 10, the two slices on the left are the user's specification for this classification process. The user first paints on the brain to classify it and then paints on the lower half of the head to provide context. The right image shows the result of the classified brain and the rest of the head without the upper half of the skull and other materials covering the brain. With traditional transfer functions, it is difficult to achieve a similar classification since the brain and the materials near the skull have similar scalar values. But, when texture and position are taken into account, as in our approach, the two materials can be separated. Showing only the lower half of the skull and skin is achieved mainly by considering the position information.

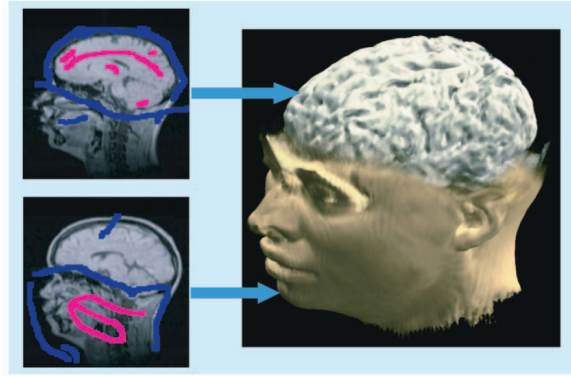


Fig. 10. The result of the classification of multiple materials. The slices shown on the left are the user's specification for classifying the brain and the bottom of the head. The right image is the rendering result of this classification.

Next, we tested the classification of multiple materials with a knee data set containing several bones. Since the bones are composed of the same materials, it is difficult to separate them using only properties from the material itself such as data value and gradient magnitude. Position information plays an important role in this classification task. With our system, a user is able to put different pieces of bones into different classes. Fig. 11 shows the knee data set and the classification result of different pieces of bones. In the left image, a volume rendering result using 1D transfer function is shown. The right image is created with our system where the three bones are separated and rendered with different colors.

Fig. 12 is the result of classifying multiple materials from a chest data set. In this case, the bones and lungs have been classified. The lung has similar data value to other tissue and has been separated using gradient and position information.

## 7.3 Information Reuse

Using the painting metaphor with immediate visual feedback, the process of training a classifier is straightforward, but it can take a few minutes to complete a classification.

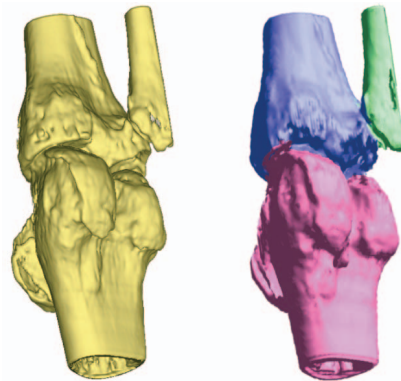


Fig. 11. The left image is a rendered result of the knee data set with a 1D TF where all the bones have the same data properties. The right image shows the classification result of three different pieces of bones using our system, which takes into account position information for classification.





Fig. 12. The result of classifying the bone and lung from the chest data set.

Therefore, it can be advantageous to save these intelligent classifiers for reuse on data sets with similar characteristics. The capability of sharing and reusing a trained system can be very powerful, especially for a comparative study such as monitoring a patient's condition over time. In addition, one might want to save the classifier created by a person with a high level of expertise so that the knowledge may be shared and reused.

When reusing a trained classifier, it is necessary that the two data sets share similar characteristics. For example, if data value is used for classification, the same range of data values in two data sets should represent the same material. If position is taken into account, the materials in the different data sets should be at similar locations.

To reuse a classifier, we save the neural network or support vector machines configuration after training. The amount of data to be saved is small, under 30 kilobytes. Fig. 13 shows the results of reusing a classifier on two anatomical MRI scans of the same person taken several days apart. Fig. 13a shows the first data set rendered with a 1D transfer function. In Fig. 13b, the same data set is shown using an SVMs classifier to extract the brain. In Fig. 13c, the SVMs classifier is reused on the second data set generated eight days later. The reused classifier performs reasonably well, with some misclassification in the frontal lobe. The

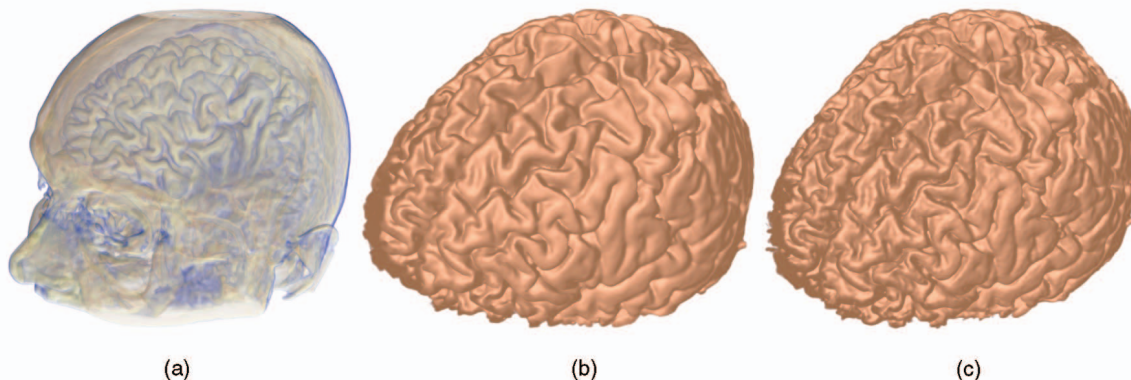


Fig. 13. (a) A volume rendering result of a MRI brain data set with a 1D transfer function. (b) Classification result using an SVMs classifier to classify the same brain data set. (c) Result of applying the previous classifier to a different scan the same person taken several days apart.

user, however, can still use this classifier as a starting point for refinement by supplying additional training data.

In medical practice or research, doctors often have to look at a large number of data sets in a short period of time for similar types of diagnoses or studies. The reuse of our high-dimensional classifier can help increase their performance in a significant way since classifying each data set from scratch would be very time consuming. With our method, all the data sets could first be automatically classified using a trained system and the results are then examined by the doctor to determine if any revision is needed. Fig. 14 shows the results of reusing a classifier for identifying tumors in different MRI brain data sets. Fig. 14a shows a volume rendered image of a tumor data set using a 1D transfer function. Next, a neural network classifier was created for this data set to isolate the tumor. The inputs used are the voxel's scalar value, gradient magnitude, and its six neighbors to define the dense and high scalar value regions. Fig. 14b shows the result of the neural network classifier capturing a tumor at the lower right of the brain. The result contains the tumor, some blood vessels, and small pieces of the skull.

The trained classifier was then applied to a different brain data set. The tumor in this data set has similar properties and is captured by the classifier, as shown in Fig. 14c. Notice that this tumor is not at the same location as the tumor in the first data set. We also applied the same classifier to a scan of a healthy brain; as expected, there is nothing captured except small pieces of the skull and some blood vessels, as shown in Fig. 14d.

We have demonstrated information reuse is feasible, but there are limitations. For example, when position information is used for classification, in many cases, better results can be obtained since location is an important attribute in differentiating materials with similar properties. However, position information is often specific to a given data set and can be undesirable when classifier reusability is intended. Position information was not used for the results in Fig. 14 since tumor position is not a learnable attribute, but rather, is something that changes between data sets. As a result, the classified results show some undesired materials, such as blood vessels and small pieces of skull. By taking into account location, a better classifier

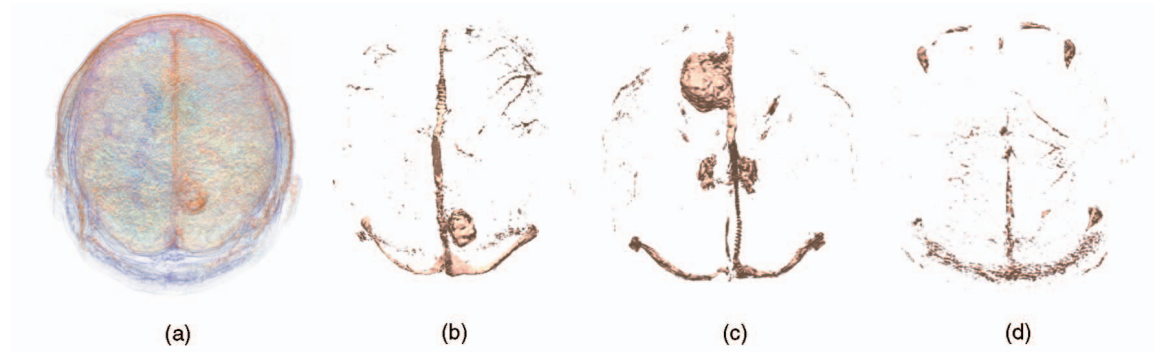


Fig. 14. (a) The data set used for training a neural network classifier. (b) The result of classifying a tumor from the data set. (c) The trained classifier was applied to a different brain data set and captures a tumor at different location. (d) The same classifier was then applied to a data set of a normal brain. No anomaly is shown after the classification.

could be constructed for isolating a tumor in a data set, but the reusability of the classifier would be low. We believe that the practical value of classifier reusability in the context of comparative study justifies this trade-off.

## 8 DISCUSSION AND FUTURE WORK

The use of higher-dimensional inputs often allows for classification to occur with improved accuracy since additional attributes can be taken into account when making classification decisions. Some care must be taken when considering these inputs since using unnecessary attributes can have detrimental effects on the resulting classifier. First, the use of unnecessary inputs increases the complexity of the resulting classifier, which reduces runtime performance for both training and classification. Although a machine learning classifier can learn to ignore these inputs, learning this behavior can often require more training data than would normally be necessary. In our implementation, the user can specify which inputs are used by a classifier at runtime based on their intuition as to which inputs should be beneficial in classification. It would be desirable to have an automatic mechanism for removing unnecessary inputs from a classifier.

When a voxel's neighboring values are used as inputs into the machine learning classifier, including the voxel's gradient magnitude as input provides seemingly redundant information. Specifically, since a voxel's gradient magnitude can be computed from its neighbors using central differencing, a classifier should be able to "learn" this relationship during training. Including redundant but relevant input attributes has been shown to improve learning performance [4], so we include gradient magnitude since it is an important property in classification. In our experience, incorporating the gradient information directly can reduce the amount of training time and data required to achieve a given classification result.

In our implementation, the volume rendered results can be slightly different when using hardware and software neural networks. Our hardware neural network renderer performs classification in the fragment program using interpolated data as input. Thus, classification occurs after filtering (postclassification). For our software implementation, each voxel in the volume is sent to a neural network to

obtain a classified volume that is then rendered in hardware using linear interpolation. Thus, classification occurs before interpolation (preclassification). Unlike traditional transfer function methods, however, in the preclassified case, interpolation is performed on a scalar valued classified uncertainty volume and not on an RGBA color volume.

Our method is entirely based on the assumption that the user has an understanding of the data they wish to visualize and can look at a slice and identify how they would like regions on that slice to be classified. In some cases, it might be unclear to the user how some regions should be classified. In the future, it would be beneficial to have a mechanism for the user to communicate their uncertainty over how different regions in a volume should be classified.

With a traditional transfer function user interface, the resulting transfer function can often provide some insight into the data properties of features in a volume. The resulting classification functions generated with our method could similarly provide meaningful information to scientists about the higher-dimensional nature of the materials they are studying. One of the difficulties of machine learning classifiers and neural networks, in particular, is that the structure and values of the resulting classifiers do not provide direct user understandable information about the nature of the resulting classification function. Some research has been performed in visualizing neural networks to try to gain understanding of their underlying classification. We believe that similar analysis of the classification functions used in our work could prove helpful to scientists.

Additional future work includes determining how our painting interface could be combined with the traditional transfer function paradigm. The technique described in our paper could be adapted to generate traditional 1D or 2D transfer functions by using artificial intelligent systems with one or two inputs. However, much of the power of our method comes from higher-dimensional classification functions that would otherwise be lost. Thus, the challenge in this type of integration would be to maintain the high-quality classification that comes from higher-dimensional classification with some of the interface characteristics of traditional transfer function interface.

## 9 CONCLUSION

An effective visualization is able to communicate information about those specific spatial structures that are of interest to the viewer without the distraction of materials that are not of interest. Since the types of structures of interest vary widely depending on the user of a system, user interfaces for classification must be powerful enough to provide high-quality classification, yet intuitive enough to be accessible to a wide range of scientists.

We have described a new type of volume visualization user interface that allows the user to specify which regions in a volume they would like to visualize by simply painting on a few slices from that volume. Abstracted from the user is a higher-dimensional classification function implemented using artificial intelligence systems that makes effective use of a voxel's value, gradient magnitude, individual nearest neighbors, and spatial position. Thus, the new user interface is not only more intuitive than specifying a one-dimensional transfer function curve, it is also more powerful because it uses far more information for classification. In addition, a classification function can also be reused to perform similar classification tasks on different data sets.

The rendering of the classified volume can be implemented in graphics hardware, providing maximum interactivity, which is critical for giving users the ability to control which aspects of the volume they visualize. We believe more intuitive interfaces, like the one we describe, will make volume visualization more accessible to a wider range of scientists.

## ACKNOWLEDGMENTS

This work has been sponsored in part by the US National Science Foundation under contracts ACI 9983641 (PECASE award), ACI 0325934 (ITR), ACI 0222991, and CMS-9980063 and the US Department of Energy under Memorandum Agreements No. DE-FC02-01ER41202 (SciDAC) and No. B523578 (ASCI VIEWS). The cryosection data set was provided by Dr. Arthur Toga of the UCLA Brain Research Institute. The "Bruce Gooch's Brain" data sets were provided by Bruce Gooch at Northwestern University. The authors would also like to thank the members of the University of California Davis visualization and graphics group for the valuable discussion and providing some of the test data sets.

## REFERENCES

- [1] C.L. Bajaj, V. Pascucci, and D.R. Shikore, "The Contour Spectrum," *Proc. IEEE Visualization '97 Conf.*, pp. 167-175, 1997.
- [2] V. Blanz, B. Schölkopf, H. Bülthoff, C. Burges, V. Vapnik, and T. Vetter, "Comparison of View-Based Object Recognition Algorithms Using Realistic 3D Models," *Proc. Int'l Conf. Artificial Neural Networks*, pp. 251-256, 1996.
- [3] B.E. Boser, I. Guyon, and V. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," *Proc. Fifth Ann. Workshop Computational Learning Theory*, pp. 144-152, 1992.
- [4] K.J. Cherkauer and J.W. Shavlik, "Rapid Quality Estimation of Neural Network Input Representations," *Advances in Neural Information Processing Systems*, vol. 8, pp. 45-51, MIT Press, 1996.
- [5] C. Cortes and V. Vapnik, "Support Vector Network," *Machine Learning*, vol. 20, no. 3, pp. 273-297, 1995.

- [6] N. Friedman and D. Geiger, and M. Goldszmidt, "Bayesian Network Classifiers," *Machine Learning*, vol. 29, nos. 2-3, pp. 131-163, 1997.
- [7] I. Fujishiro, T. Azuma, and Y. Takeshima, "Automating Transfer Function Design for Comprehensible Volume Rendering Based on 3D Field Topology Analysis," *Proc. IEEE Visualization '99 Conf.*, pp. 467-470, 1999.
- [8] A. VanGelder and U. Hoffman, "Direct Volume Rendering with Shading via Three-Dimensional Textures," *Proc. ACM Symp. Volume Visualization '96 Conf. Proc.*, pp. 23-30, 1996.
- [9] E. Gelenbe, Y. Feng, K. Ranga, and R. Krishnan, "Neural Networks for Volumetric MR Imaging of the Brain," *Proc. Int'l Workshop Neural Networks for Identification, Control, Robotics, and Signal/Image Processing*, pp. 194-202, 1996.
- [10] L.O. Hall, A.M. Bensaid, L.P. Clarke, R.P. Velthuizen, M.S. Silbiger, and J.C. Bezdek, "A Comparison of Neural Network and Fuzzy Clustering Techniques in Segmenting Magnetic Resonance Images of the Brain," *IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 672-682, 1992.
- [11] T. He, L. Hong, A. Kaufman, and H. Pfister, "Generation of Transfer Functions with Stochastic Search Techniques," *Proc. IEEE Visualization '96 Conf.*, pp. 227-234, 1996.
- [12] M.A. Hearst, "Trends and Controversies: Support Vector Machines," *IEEE Intelligent Systems*, vol. 13, no. 4, pp. 18-28, 1998.
- [13] R. Huang and K.-L. Ma, "RGVis: Region Growing Based Techniques for Volume Visualization," *Proc. Pacific Graphics '03 Conf.*, pp. 355-363, 2003.
- [14] T.J. Jankun-Kelly and K.-L. Ma, "A Study of Transfer Function Generation for Time-Varying Volume Data," *Proc. Joint IEEE TCVG and Eurographics Workshop*, pp. 51-68, 2001.
- [15] G. Kindlmann and J.W. Durkin, "Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering," *Proc. '98 IEEE Symp. Volume Visualization*, pp. 79-86, 1998.
- [16] J. Kniss, G. Kindlmann, and C. Hansen, "Interactive Volume Rendering Using Multi-Dimensional Transfer Functions and Direct Manipulation Widgets," *Proc. IEEE Visualization '01 Conf.*, pp. 255-262, 2001.
- [17] A. König and E. Gröller, "Mastering Transfer Function Specification by Using VolumePro Technology," *Proc. Spring Conf. Computer Graphics*, vol. 17, pp. 279-286, 2001.
- [18] M. Levoy, "Display of Surfaces from Volume Data," *IEEE Computer Graphics and Applications*, vol. 8, no. 3, pp. 29-37, 1988.
- [19] J. Marks, B. Andalman, P. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber, "Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation," *Proc. SIGGRAPH '97*, pp. 389-400, 1997.
- [20] A. McCallum, D. Freitag, and F. Pereira, "Maximum Entropy Markov Models for Information Extraction and Segmentation," *Proc. 17th Int'l Conf. Machine Learning*, pp. 591-598, 2000.
- [21] E. Osuna, R. Freund, and F. Girosi, "Training Support Vector Machines: An Application to Face Detection," *Proc. '97 Conf. Computer Vision and Pattern Recognition*, pp. 130-137, 1997.
- [22] L.I. Perlovsky, *Neural Networks and Intellect: Using Model-Based Concepts*. Oxford Univ. Press, 2000.
- [23] H. Pfister, B. Lorensen, C. Bajaj, G. Kindlmann, W. Schroeder, L.S. Avila, K. Martin, R. Machiraju, and J. Lee, "The Transfer Function Bake-Off," *IEEE Computer Graphics and Applications*, vol. 21, no. 3, pp. 16-22, May/June 2001.
- [24] D. Rumelhart and J. McClelland, *Parallel and Distributed Processing: Explorations in the Microstructure of Cognition*. The MIT Press, 1986.
- [25] I. Takanashi, E.B. Lum, K.-L. Ma, and S. Muraki, "ISpace: Interactive Volume Data Classification Techniques Using Independent Component Analysis," *Proc. Pacific Graphics '02 Conf.*, pp. 366-374, 2002.
- [26] J. Thorsten, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," *Proc. 10th European Conf. Machine Learning*, pp. 137-142, 1998.
- [27] F.-Y. Tzeng, E.B. Lum, and K.-L. Ma, "A Novel Interface for Higher-Dimensional Classification of Volume Data," *Proc. IEEE Visualization '03 Conf.*, pp. 505-512, 2003.
- [28] P. Werbos, "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences," PhD thesis, Dept. of Applied Math., Harvard Univ., 1974.





**Fan-Yin Tzeng** received the bachelor's degree in computer and information science in 2001 from the National Chiao-Tung University in Taiwan. She is a PhD candidate in computer science at the University of California, Davis. Her research interests primarily involve scientific data visualization, centering on volume visualization and intelligent systems-based user interfaces.



**Kwan-Liu Ma** received the PhD degree in computer science from the University of Utah in 1993. He is a professor of computer science at the University of California (UC), Davis. His research spans the fields of visualization, computer graphics, and high-performance computing. During 1993-1999, he was with ICASE/NASA LaRC as a research scientist. In 1999, he joined UC Davis. In the following year, he received the presidential Early Career Award for Scientists and Engineers (PECASE). Presently, he is directing research projects on parallel visualization, volume modeling and visualization, artistically inspired illustrations, visual interface designs, and information visualization. He is the editor of the VisFiles Column of ACM SIGGRAPH's *Computer Graphics Quarterly*. He is a senior member of the IEEE and the IEEE Computer Society. Information about Professor Ma's publications and research projects can be found at: <http://www.cs.ucdavis.edu/~ma>.



visual representation of scientific data. He is a member of the IEEE and the IEEE Computer Society.

**Eric B. Lum** received the PhD degree in computer science from the University of California (UC), Davis in 2004 and the BS and MS degrees in electrical engineering from the University of California Los Angeles in 1997 and 1999, respectively. He is a postdoctoral researcher in computer science at the University of California, Davis. His research interests involve the development of rendering and user interaction methods that facilitate the expressive

▷ **For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/publications/dlib](http://www.computer.org/publications/dlib).**