FINAL REPORT
Colorado Advanced Software Institute

SEMI-AUTOMATED BOUNDARY TRACING OF MEDICAL IMAGES FOR THREE-DIMENSIONAL MODEL DEVELOPMENT

Principal Investigator:	Charles Anderson, Ph.D. Associate Professor Department of Computer Science Colorado State University, Fort Collins
Graduate Student:	Stewart Crawford-Hines Department of Computer Science Colorado State University, Fort Collins
Collaborating Company:	Visible Productions, Inc. Thomas McCracken, Vice President Representative

Project Title:	SEMI-AUTOMATED BOUNDARY TRACING OF MEDICAL IMAGES FOR THREE-DIMENSIONAL MODEL DEVELOPMENT
Principal Investigator:	Charles Anderson, Ph.D.
University:	Colorado State University, Fort Collins
Collaborating Company:	Visible Productions, Inc.
Collaborating Company Representative:	Thomas McCracken, Vice President

As authorized representative of the collaborating company, I have reviewed this report and approve it for release to the Colorado Advanced Software Institute.

Signature ______________________ Date ______________

Contents

SEMI-AUTOMATED BOUNDARY TRACING OF MEDICAL IMAGES FOR THREE-DIMENSIONAL MODEL DEVELOPMENT

Abstract

Segmentation is a critical step in many image processing applications. It is usually performed manually, since automatic techniques yield results which are less than satisfactory. We are experimenting with a semi-automated approach based on neural networks that models the decisions made by users as they trace the boundaries of regions of interest in medical images. At Visible Productions, Inc., a Fort Collins, CO, company, our method is being integrated with the company's software system for constructing three-dimensional models of human anatomy. Their current procedure requires many person-hours of hand-tracing regions in cross-section images. Visible Productions has identified our semi-automated technique as a technology that greatly reduces their production time and cost.

1 Problem

The need for creating accurate anatomical models from a variety of input sources is becoming central for the broad-based population of researchers, educators, and medical professionals. Virtually all of the common input sources are 2-D representations of parallel slices of 3-D anatomical structures. These 2-D images are typically CT, MRI, PET, confocal imagery, EM, Visible Human Project (1996) cryosections, or other sources of cross-sectional images. Images display some or all of the tissue on a certain level, and do not explicitly distinguish between separate tissue types. A difficult problem is segmenting the separate tissue types on a given level and organizing the resulting data on the 2-D slices into 3-D polygonal surface models of the anatomical structures. An alternative to the assembling of 2-D slice information into 3-D surface models is volume-based modeling, but it typically requires 10-20 times more computing resources (memory and CPU speed).

Currently there are no systems for developing high-fidelity polygonal models at a reasonable cost and in a timely fashion from this type of 2-D slice data. Model construction currently involves either many skilled person-hours (to manually trace the boundaries of the desired structures on each level) or use of automatic methods which take massive amounts of memory and computational power, and produce inadequate results. These methods are woefully inadequate for satisfying the needs of researchers, teachers, and medical diagnosis.

Visible Productions, Inc. is bridging the gap between the time-consuming manual tracing (which produces accurate results) and the faster automatic methods (which produce poor results) by developing an automated polygonal model generator which is both fast and accurate. To speed the tracing task, we are using neural networks to model the decisions of human tracers, then using the trained neural networks to automate much of the human tracer's job.

The remainder of this report consists of a statement of our objectives followed by summaries of our research approach, results, and the method of evaluation. Then the transferred technology is described. We conclude with summaries of networking, publications, and follow-on funding.

2 Objective

The intended outcome of this work was two-fold. First, we developed an initial implementation of our semi-automated approach to segmentation. This required research into ways of representing the information needed to automatically segment an image. The second outcome was a software prototype of our semi-automated tracing system, which was delivered to Visible Productions. Our initial studies resulted in a number of refinements to the method and the user interface. The follow-on funding objective was met with a successful Phase I, SBIR proposal to the National Science Foundation.

3 Background

Advanced medical imaging modalities combined with improved 3-D rendering capabilities and fast graphics hardware promise incredibly rich visualizations for medical research, practice, and instruction. After preliminary analysis of the raw imaging data, for example, active contour models can follow the beating of a heart, and sets of control points on the surface of an organ along with texture maps can be used to generate photo-realistic illustrations and biomedical virtual realities(Kerr et al., 1996). Before these are possible, though, the basic raw image data must be analyzed and the tissues and organs of interest delineated; the resultant outlines are then used as a basis for higher-level models and interpretations.

Currently, the reference standard for the task is an expert's delineation of the region. The state-of-the-practice is that this is done manually. There has been much research directed toward the automatic segmentation of images, from which the extraction of an outline would then proceed.

Özkan et al. (1993) note the specific motivation of their work is to relieve physicians of the manual task of tumor tracing when planning a patient's radio-surgery. In certain domains, such as tumor identification, there have been some modest successes (Raya, 1990). However, Höhne et al. (1996) note, *"automatic segmentation is not possible with present-day technology [and] interactive segmentation methods are still too time-consuming for broad clinical-application..."*

Figure 1a is derived from the National Library of Medicine's Visible Human Project (). This project has collected 1,800 high-resolution, full color photographs, each a cross section through a human male, taken at 1mm intervals. Figures 1b and c illustrate standard Laplacian edge detection operators applied to this image. Figure 1b is tuned for strong edges: the outline of the head and roughly two-thirds of the skull are well defined, though there is no detail evident within the cortex. Figure 1c is tuned for weaker edges, to pick up the finer cortical distinction between white and grey matter: much of that detail is now visible, but at the expense of cluttering the visualization with all the other edges, from weak to strong, within the image. For this technique to be useful in capturing the finer detail in the image, we must be willing to filter somehow the spurious clutter. Often the task of removing the clutter is more costly than simply manually outlining the region of interest from the start, which is the reason why automatic image processing techniques are seldom used in practice for segmentation in medical applications. Similarly, snakes (Kass et al., 1987) have not made major inroads in automating this task, because they require manual adjustments of parameters, adjustments which were too lengthy or unwieldy to make in practice.

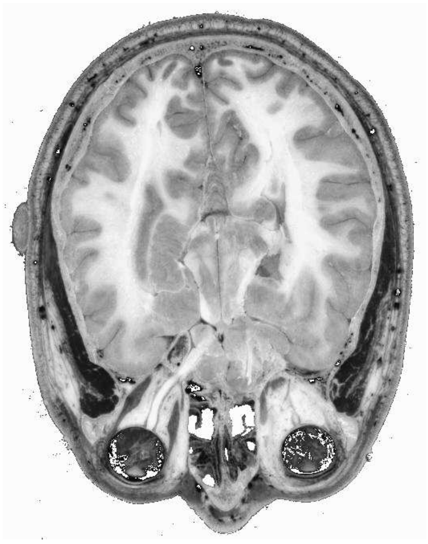
4 Approach

4.1 Introduction

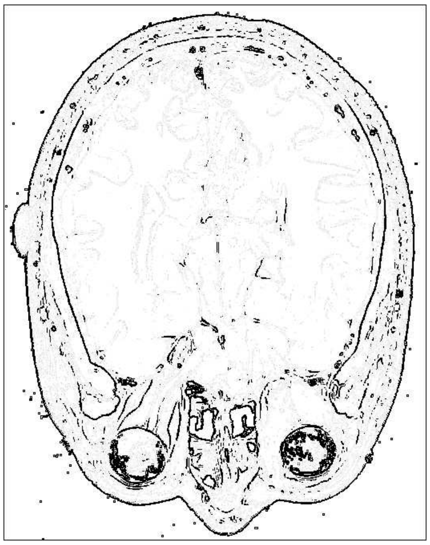
Our approach and results are summarized in this section and in Crawford-Hines and Anderson (1998) and Crawford-Hines and Anderson (1997).

Our work is focused on using neural networks to assist an expert in the region segmentation task. As a person begins tracing a tissue or an organ, this data is used to train a neural network to follow the same contour in the image which the expert is tracking visually. Figure 1d illustrates region definitions learned by, then continued by such a network. The initial segments indicated as "Training Segments" were acquired by tracking the cursor. The network has been trained twice, once to learn the skull boundary distinction, and again independently to learn the white/grey matter distinction. There is a lack of clutter (since from the start the network is focused on one specific task), and the edges defined are crisp, one-pixel wide boundaries. Of the roughly 2,000 pixels on these traced edges shown here, 80 (4%) were traced manually (used to train the network and establish a direction for the trace) and the remainder placed by the network. Training the network to trace took roughly two minutes, and the unassisted extending of the initial trace is on the order of milliseconds (on an HP700 RISC machine).

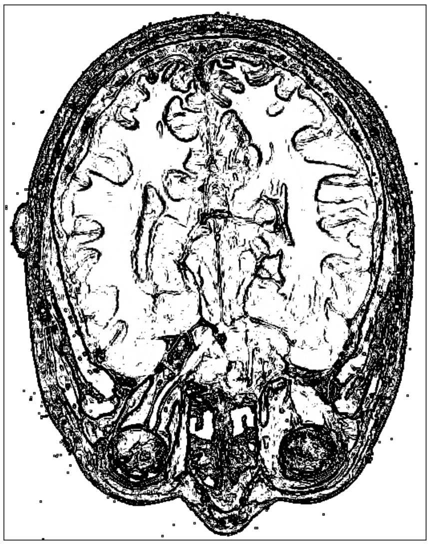
This semi-automated tracing tool is not limited to color photographs. Figure 2 shows a slice of data from an MRI scan, provided by NIMH (National Institute of Mental Health). Researchers at NIMH are investigating possible correlations between the structure of the cerebral cortex and symptoms of schizophrenia. To this end, they have approximately 300 MRI scans of patients from which they must produce accurate 3-D models of the cortex surface. Currently, they painstakingly erase away all non-brain tissue one slice at a time. Obviously, this will take many months. In Figure 2, some preliminary tests of our tracing tool on their MRI images is shown. The relatively long sequence of green points near the top center of the figure were traced by the user and a neural network was trained on just those points. The trace was then continued by the neural network with points shown in red, until it reached a section where it could not confidently determine which way to go. The user then restarted the trace by drawing more points, in green, and the network continued again, without additional training. This shows that our approach has the potential to greatly reduce the time needed to separate brain and non-brain tissue in MRI scans.



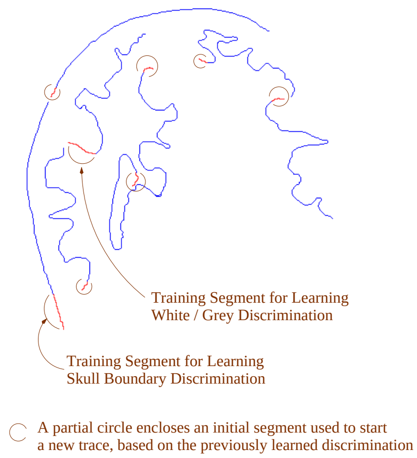
a. Original image obtained from a photograph of a head cross-section, from Visible Human Project ().



b. Laplacian edge detection tuned for strong edges.



c. Laplacian edge detection tuned for weak edges.



d. Short hand-drawn segments and long boundary segments drawn by neural networks.

Figure 1: Comparison of conventional edge-detection and our semi-automated approach.

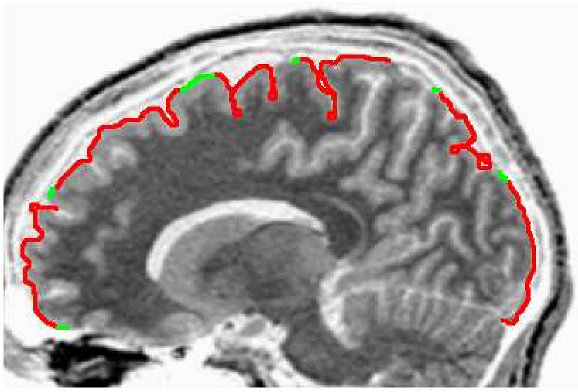
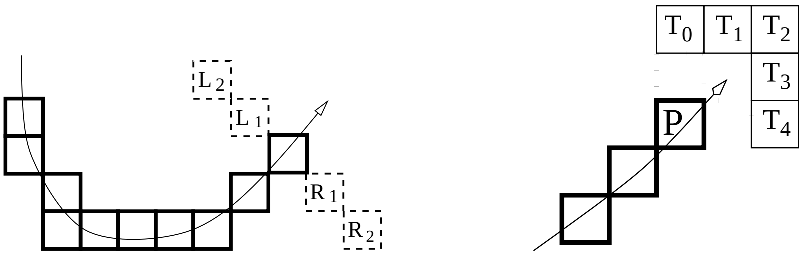


Figure 2: Example of our semi-automated tracing system applied to an MRI scan, using software produced from our prior SBIR Phase I award. Less than 10% of the boundary pixels were traced by hand (colored green).

4.2 Method

A trace along a boundary is a set of points connected in a free-form line. Since this boundary captures the movement of a cursor over an image, it has a direction of travel at each point along the trace. A *path* is the set of points included in the trace at any given time. Figure 3a illustrates a point on a path, and its immediate left, L_i, and right, R_i, neighbors. These neighbors are determined by perpendicular extensions from the given path. As a path along a boundary is traversed, there exists some distinction between the left neighbors, right neighbors, and points along the path. For example, all the left neighbors may be dark pixels and the path and right neighbors are light pixels. The goal is to train a neural network to learn this distinction. While many techniques might be employed to learn the measure, our focus on neural nets stemmed from their relative freedom from assumptions about input data, and our earlier work (Crawford-Hines and Anderson, 1994) exploring the ability to develop nonlinear classifiers in region segmentation tasks.



a. Point T and its neighbors. b. Potential 2-step look-ahead.

Figure 3: Information used in choosing next boundary pixel.

Given that a path of at least several pixels has been traced, Figure 3b illustrates the basic task of extending this trace. Point P is at the end of the path traced so far. Using a straight geometric projection alone, the path would be projected to point T_2 in two steps. By constraining the curvature of the path, we can assume that all possible two-step extensions of the path are covered by T_0 through T_4. Here's the basic path extension algorithm:

1. consider the two-step path extensions T_0, T_2, and T_4;

2. evaluate each of those extensions and its hypothetical neighbors, were it to be a future point on the path, as shown in Figure 4;
3. move one step (to P_0, P_2, or P_4) along the path of the best two-step evaluation;
4. iterate.

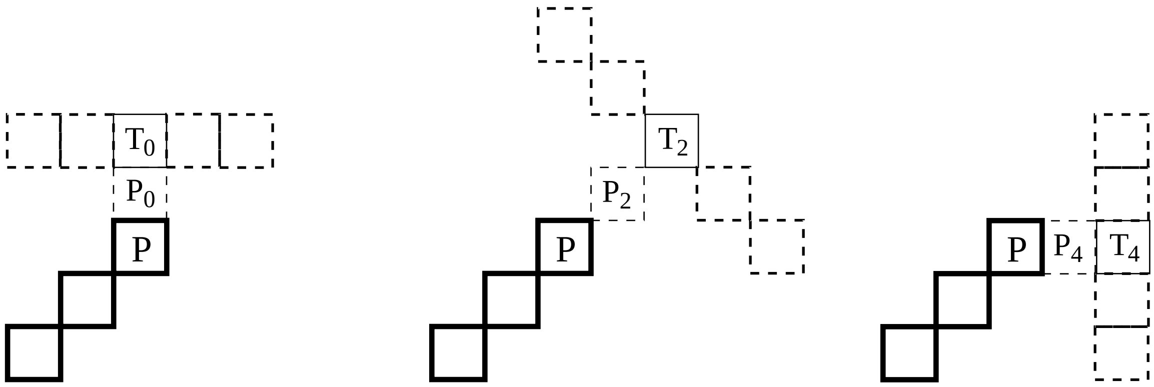


Figure 4: Some possible extensions beyond P, and their neighbors.

Two-step extensions are considered because evaluating only one-step extensions (a pure greedy strategy) resulted in path extensions with excessive jitter on straight segments.

In our initial tests, promising results have come from the simple feed-forward network structures. The neural network used for the tracings in Figure 1d had a 5-N-1 topology. The five inputs represent grey-scale pixel values (normalized to [0,1]) of a point on the path, its two left neighbors, and its two right neighbors. N was set to about one-half the expected amount of training data available (from 10 to 15). The single output unit was trained to respond in the range [0,1], where an output of 0.5 implies the selected pixel is on the boundary, less than 0.5 implies the point is off to the left of the path, and greater than 0.5 implies it is off to the right.

The training set was established by an initial path defined by the user. The two training segments of Figure 1d are indicated by short segments marked by half-circles. Five or six points along this path were selected for training (the actual number is a function of the path's geometry). Each point on the path and its left and right neighbors could be added to the training set with a target value of 0.5, but less obvious is the training set for off-path exemplars. Referring back to Figure 3, five input-vector and target pairings can be associated with a given training point on the path: T_2 is set to be on the actual path extension (target=0.5); T_1 and its neighbors, slightly off to the left, are given a target of 0.25; T_0, yet further left, has 0.10 as a target. Target values for T_3 and T_4 are 0.75 and 0.90, respectively. The goal is to generate a single-valued function which sweeps from 0 to 1, as the path extension options are considered from the left to the right. We call this the smooth-evaluation, or SEV, method. An alternative way to assign target values is to assign boundary pixels target values of 1, and non-boundary pixels target values of 0. We call this the feature detector, or FD, method.

The neural net was trained through standard incremental error backpropagation. Training was stopped when the squared error over the training set leveled off. The several-hundred pixel extensions of the initial path are the test set. After an initial training on the network, the path can be restarted elsewhere in the image to judge the net's discrimination in other areas. In Figure 1d, initial segments are marked by incomplete circles. All the white/grey discriminations made in the cortical areas of Figure 1d were based on the one white/grey training segment shown.

4.3 Input Representation

We performed some initial experiments aimed at verifying the hypothesis that preprocessing of inputs to the neural network speeds learning. The experimental design compared three ways of representing the inputs, two ways of specifying target values, and two tracing tasks. The speed of learning was quantified by the time taken to reach an RMS error of 0.1 from the target values

The three input representations consisted of the raw pixel values and two, neurologically-inspired models, illustrated in Figure 5.

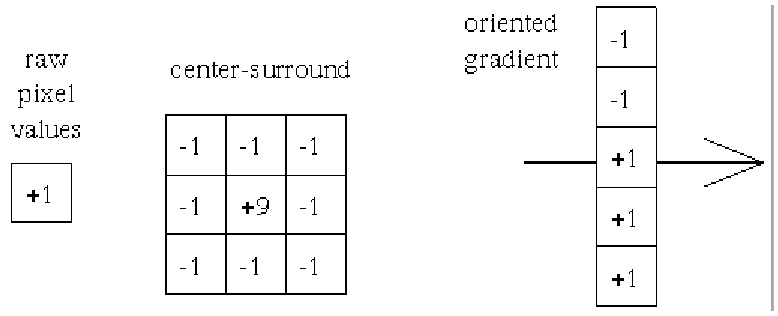


Figure 5: Three ways of representing, or preprocessing, the intensity values of a pixel in the image.

The center-surround filter is based on early retinal processing, and the oriented gradient represents an oriented filter, such as those in the complex cells. These three filters evaluate to +1 when over a constant area. The two ways of assigning target values were those discussed above: the SEV (smooth-evaluation-function), and the FD (feature-detector). The two tasks were edge following and line following. The simple image shown in Figure 6 was used for the experiments to minimize confounding influences of noise and texture in real images.

Figure 6: Simple image used to test the edge and line following abilities of our system.

The neural network topology consisted of one input layer with five inputs, preprocessed as discussed above, 10 hidden units, and one output unit. The initial training segment, the pseudo-random network initialization, and the learning and momentum rates were held constant for all experiments.

Table 1 summarizes the full set of initial results In one specific case, the SEV output with raw pixel inputs on the edge-following task was the fastest learning system, reaching its learning objective in 200 epochs. With the other two input representations, several thousand training epochs were required. A drawback of the SEV output model is that it works only on edges and fails miserably when learning to follow a line.

Target Method	Task	Raw Pixels	Center - Surround	Oriented Gradient
SEV	Edge Task	traces well learns very fast	traces well learns fast	traces well
	Line Task	traces poorly	traces poorly	traces poorly
FD	Edge Task	traces well	learns fast	traces well learns fast
	Line Task	traces ok	traces well	traces well learns fastest

Table 1: Results of filter experiments.

With the FD output model, both tasks could be learned well. For the edge-following task, the center-surround and oriented-gradient filters both reached their learning criteria in 100 epochs versus 800 epochs for raw pixel inputs. Equivalent improvements were evident for the line-following task.

It is interesting to note that the SEV outputs with raw pixel inputs showed the fastest learning, but is the most specific combination tested. In a way, they are custom fit for each other. On an edge, the pixels on one side will all have low values and high values on the other; it should be easy to learn a smooth mapping onto [0,1] in this case.

Using the FD output model, the network could learn both tasks, and in this case the preprocessed inputs yielded faster learning. This neurologically-inspired combination was a more general learning mechanism, even though somewhat slower in learning overall.

4.4 Prototype in Matlab

The feasibility of our approach to semi-automated segmentation was demonstrated by building a prototype of a complete software system for constructing 3-D models from boundary contours drawn on 2-D slices. Matlab was chosen as the programming environment for several reasons:

- neural network training and application software is very easy to implement and modify in the matrix-based Matlab language;
- visualization tools in Matlab are very useful for displaying and interacting with the 2-D slices, for displaying the results of neural network training, and for displaying the 3-D models;
- Matlab's GUI tools simplify the construction of intuitive user interfaces;
- interfaces with code and executables external to Matlab are easy to develop;
- Matlab is available on a wide variety of computing platforms.

This subsection describes the prototype system by summarizing how the user interacts with the system's three windows—the boundary tracing, neural network training, and the volume visualization windows, as shown in Figure 7.

The user first deals with the boundary tracing window, shown in Figure 8. A slider on the left of the window is used to select one of the slices in the volume of data to be used. The slice is displayed as an intensity image in the large central area of the window. Along the right side are buttons that are used to switch the mode of interaction. See Figure 9 for a closer view of the buttons. Clicking on the "Trace" button activates the tracing mode, in which the user can outline the boundary of a region of interest. As the user moves the mouse, the pixels selected are marked in red. If an error is made, the red pixels that were incorrectly marked can be deleted by clicking the "Select to Delete"

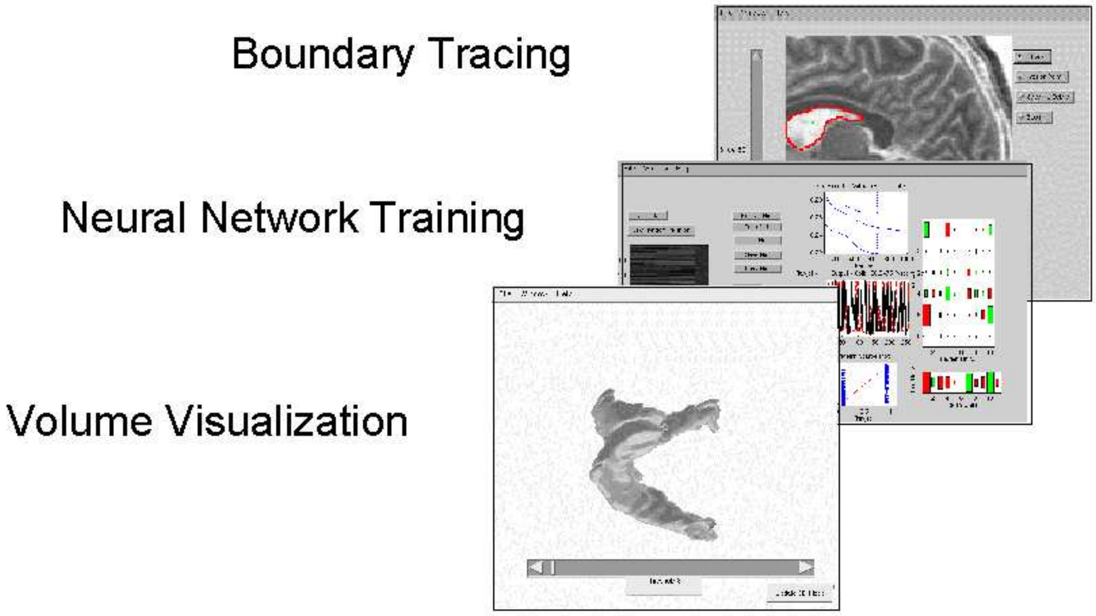


Figure 7: The prototype in Matlab consists of three windows, one for tracing the boundary of regions in slices, one for training the neural network, and one for visualizing the constructed 3-D model.

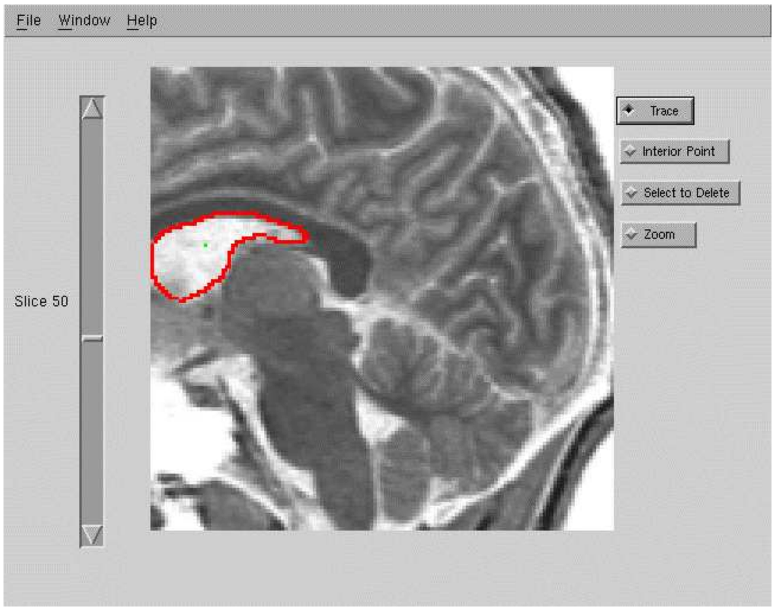


Figure 8: The user has hand-traced the boundary of a ventricle in this MRI scan of a human head. The slider on the left shows that this is slice number 50 from a whole-head scan. The red pixels on the image show the pixels traced by the user.

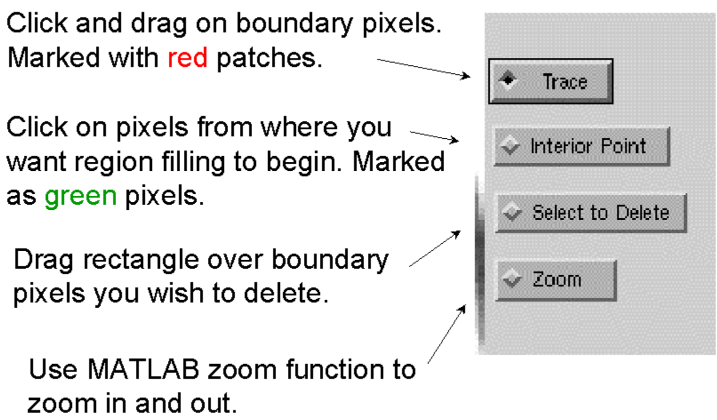


Figure 9: Buttons on the boundary-tracing window are used to edit the boundary and to zoom in and out.

button. To completely specify the region in a slice, an interior point must be selected by clicking the "Interior Point" button. If a more detailed view is needed, the user may click on the "Zoom" button and use Matlab's built-in function for zooming in and out of an image using the mouse.

At any point the user may choose to train the neural network to model the boundary decisions that have been made so far on all of the slices in which boundaries have been drawn. This is done by first selecting the window for neural network training, shown in Figure 10. Along the left side of the window are buttons for obtaining and partitioning the data. These and the data display are shown in detail on the left side of Figure 11. Clicking on the "Get Data" button collects data for each boundary pixel drawn by the user. For each boundary pixel, the data consists of the five intensities along the wing of pixels perpendicular to the boundary, as described above. Also, with each pixel, is a target value that the neural network is trained to approximate that represents whether the pixel is one marked by the user, or is a non-boundary pixel near the boundary. Boundary pixels are associated with a high target value and non-boundary pixels are assigned low target values with the FD method. The display shows a thin line for each pixel. The first five strips of intensities on each line represent the strength of each of the five wing pixels, and the sixth intensity strip shows the target value. Brighter intensities represent higher values. The example shows that approximately 1,000 pixels of data have been extracted.

The result of training a neural network is very dependent on the manner in which the available data is partitioned into training, validation, and test data. Clicking the "New Random Partition" button re-partitions the data in a random fashion. One-half of the data is randomly selected to be in the training set, one-fourth is selected for the validation set, and the remaining fourth constitutes the test set. The displayed data is rearranged to show the training set data at the top of the figure, followed by the validation set and the test set. Within each set, the non-boundary pixels data are shown first, followed by the boundary pixel data.

On the right side of Figure 11, the controls for training the neural network are shown. The "Initialize Net" button sets all of the network's weights to small, random values. The next step is to type the desired values for the following parameters. The number of "hidden units" controls the size of the single hidden layer of units in the network. The "hidden learning" value is the factor that controls the size of the change in the hidden unit weights on each training step. The "output learning" value is analogous factor for the weights in the output unit. The "momentum rate" is the factor controlling how much of the previous weight change is added to the current change. The "maximum epochs" determines how many epochs, or repetitions, through the training data will be

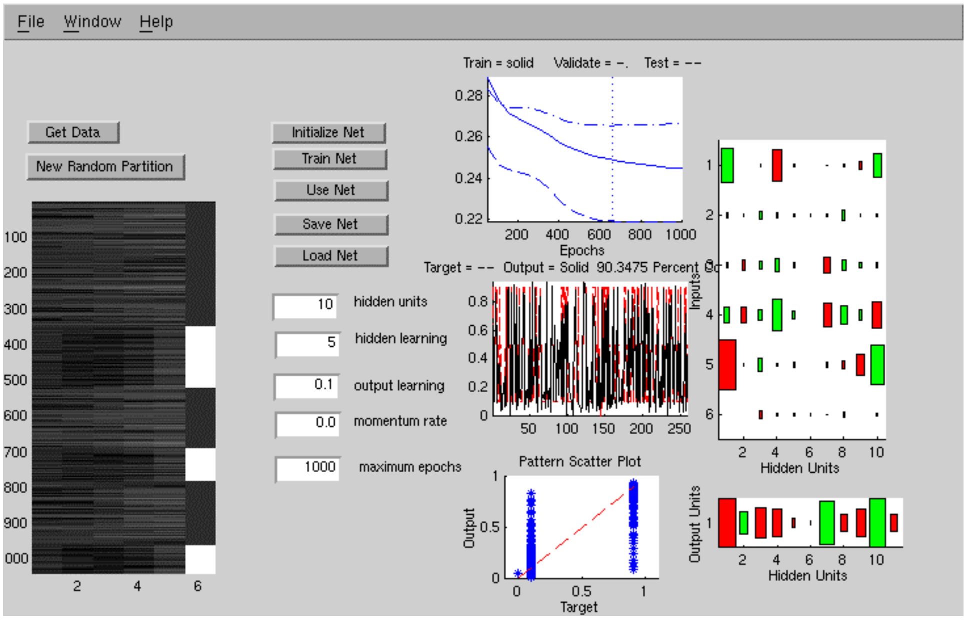


Figure 10: The window for training the neural network. It consists of three sections from left to right, explained in the following figures.

taken while training the network. Once all values are given, the user clicks on the "Train Net" button, at which time training starts and the training results are continuously updated. These are described in the next paragraph. After training, the trained net can be used for boundary tracing by clicking on "Use Net". The net may also be saved to a file by clicking "Save Net", and a previously trained net may be loaded by clicking "Load Net".

While the network is training, the results are continuously updated in the section of the neural network training window shown in detail in Figure 12. In the upper left is a graph of the RMS (root-mean-square) error between the output of the neural network and the target values, graphed with respect to epoch number. The three curves correspond to the error for the training set, the validation set, and the test set. In this case we see that the error in the training and test sets continued to decrease for all 1,000 epochs, but that the error on the validation set was lowest at about Epoch 650. Therefore, the network's state at that epoch becomes the trained network for this session. Below this graph is a graph of the actual output and target values for the test set data. The bottom graph is a scatter-plot of the test data target values versus the actual values. The best results would produce a scatter-plot with all data plotted along the dotted diagonal line. Along the right side of the window is a display of the weight values of the network at the end of training. Green boxes represent positive weight values and red boxes are negative weight values. The size of the box is proportional to the magnitude of the weights. The hidden-layer weights appear in the upper diagram and the output-unit weights are in the lower diagram.

The user may now change the neural network parameters or try a different data partition and train again. When ready to use the network, the user clicks "Use Net" and moves back to the boundary tracing window. Now the user may start tracing a boundary, either on the same slice as before or on a different slice. When the user lifts their finger from the mouse button, the system takes over and continues to trace the boundary by using the neural network to predict which pixels are on the boundary. These pixels are marked in yellow, as shown in Figure 13. In the example shown, the user started tracing the boundary of the ventricle by drawing one short, red segment near the middle of the image, after which the system took over and drew the yellow segment down and

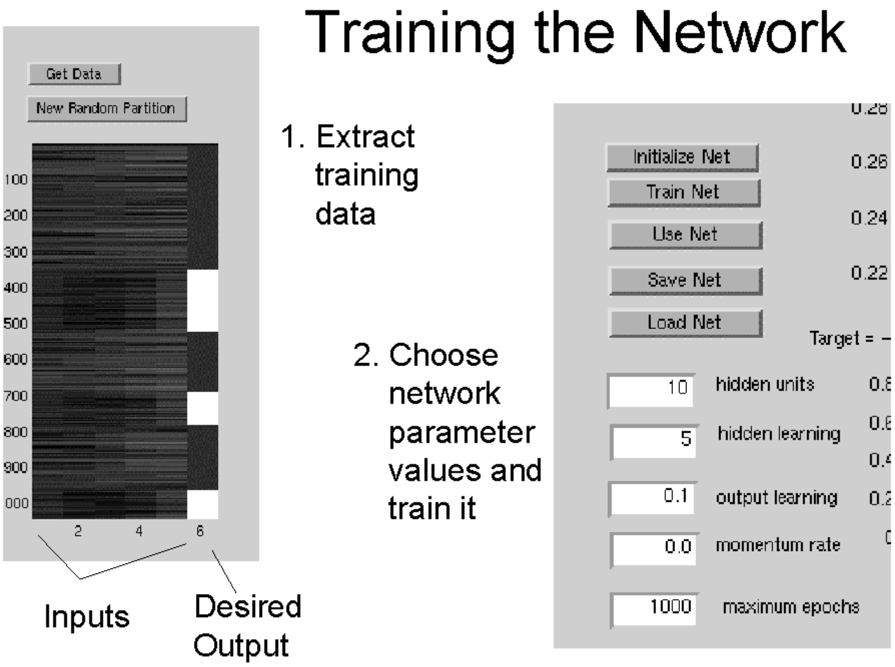


Figure 11: The left half of the figure shows the data used to train the network, displayed as a table of values shown as intensities. Each row of the table is one training example. The first five columns in each row represent the input values to the network, and the last column is the target value. Dark intensities are low values, and bright intensities are high values. The right half of the figure shows the buttons and parameter fields used to control the training of the network.

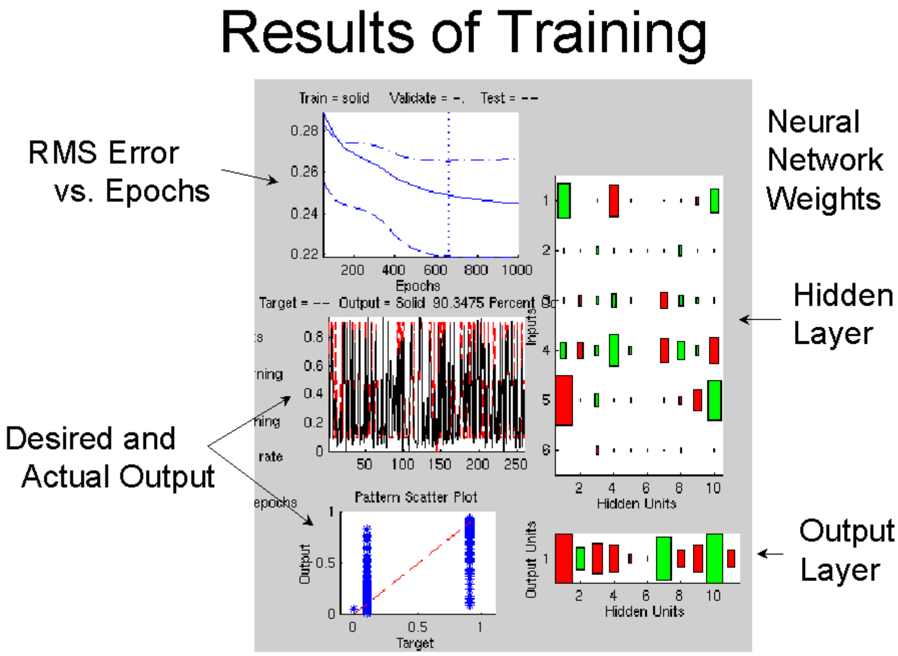


Figure 12: The training results are shown as a graph of the training, validation, and testing error versus the number of training epochs, and graphs of the desired and actual outputs. The weights learned by the neural network are displayed on the right. Green boxes represent positive weights and red ones represent negative weights. The weights are arranged in two diagrams corresponding to the two layers of the neural network.

Figure 13: Using the network. The user traces a short section of the boundary while pixels are marked in red. The neural network then continues the trace by marking pixels in yellow.

to the left. At this point, the neural network was unable to determine with confidence the next best pixel, so it stopped. The user then started drawing again, shown by the red pixels around the left edge of the ventricle. When the user released the mouse, the system took over again, traced along the top of the ventricle, then continued on to the right. The system incorrectly continued drawing beyond the ventricle. This would normally be a problem that must be corrected by the user, but in this case it is not a problem. The interior point marks the interior of the desired region, and the red and yellow pixels collectively define the interior to be just the ventricle. The yellow pixels are drawn with no visible computation lag. The process appears to be instantaneous.

At any point, the user may choose to construct a 3-D surface model of the segmented regions. The boundaries from all traced slices are extracted and passed to a triangulation routine named `polyr` that was written in C by Jensen (1995) who makes his code publicly available. It employs the marching cubes algorithm to build an initial set of polygons, then reduces the number of polygons using several methods, such as merging connected, coplanar polygons. After the user has traced all boundaries of the region of interest, the user moves to the visualization window shown in Figure 14, where the user sets a "Threshold" value using the bottom slider and clicks on "Update 3D Model". The threshold value determines the intensity at which the surface is extracted. The intensities of all pixels exterior to the traced regions are set to zero, but the interior intensities remain at their original values. This provides flexibility to the user to produce surface models that represent structures inside the selected region. In the example shown, the user segmented all of the slices containing parts of the ventricles and clicked on the "Update 3D Model". In about 10 seconds, the 3-D surface model is rendered in the window. At this point, the user may use the mouse to rotate the object in any direction.

Another example of the capability of our semi-automated approach to segmentation is shown in Figure 15. In this case, the user had hand-traced the boundary of the skull in one 2-D cross-section from the Visible Human data set. A neural network was trained on that data. The user then used

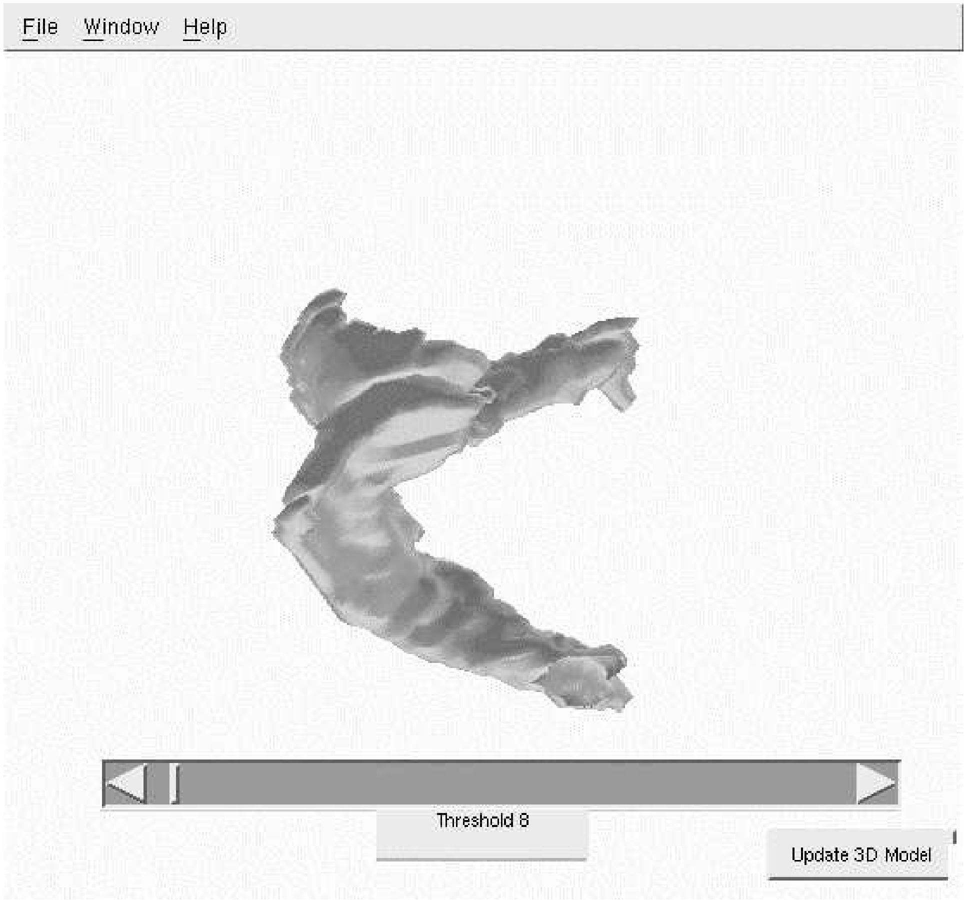


Figure 14: 3-D model resulting from the tracing of the ventricles in all slices of the MRI head scan.

the network to assist in the tracing of the skull in two other slices in other parts of the head. The figure shows that the skull boundary is very accurately traced. Only 10% of the boundary pixels were traced by hand; 90% of the boundary decisions were made by the neural network. Thus, tracing the boundary for these images using our semi-automated method was ten times faster.

Another benefit of implementing our prototype in Matlab is that the data volume is very easy to manipulate using standard Matlab expressions. For example, Figure 16 shows that by setting a part of the intensity data to the maximum value of 255, the model created by the `polyr` function will have that part of the volume missing, as if it had been cut away. The `polyr` Matlab function is our interface to Jensen's code.

5 Evaluation

To evaluate the accuracy of our semi-automated tracing system, we performed the following experiments. Cross-section photographs of the visible human head were used to obtain an contour of the skin. One complete contour in one slice was performed manually. This data was used to train a neural network to duplicate the user's decisions. This network was then used to extend a trace as much as possible. Figure 17 shows an example from part of this boundary. The blue pixels were drawn during the initial, manual trace, and the purple pixels were drawn by the trained neural network.

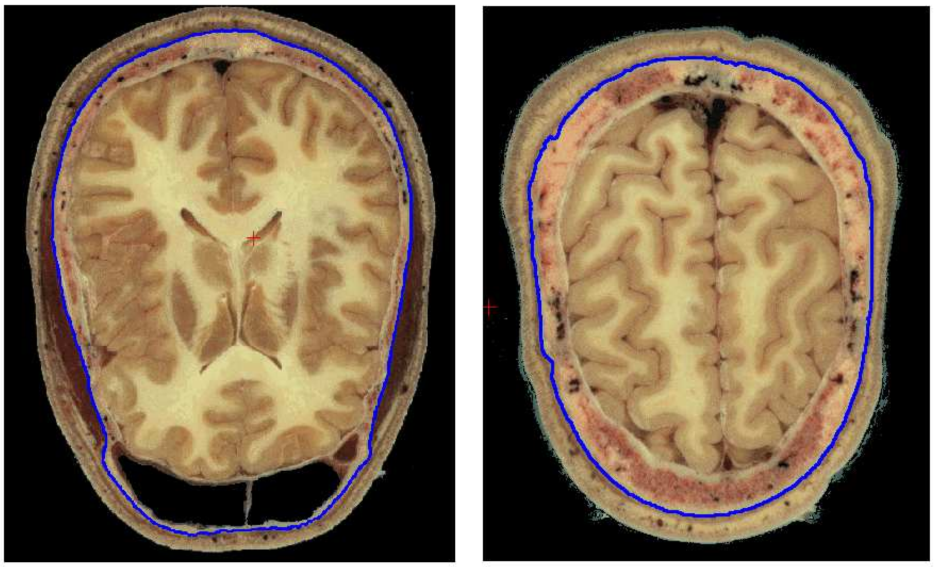


Figure 15: A user has trained a network to trace the skull boundary on one slice from the Visible Human dataset. Two additional slices were then traced with the assist of the trained neural network.

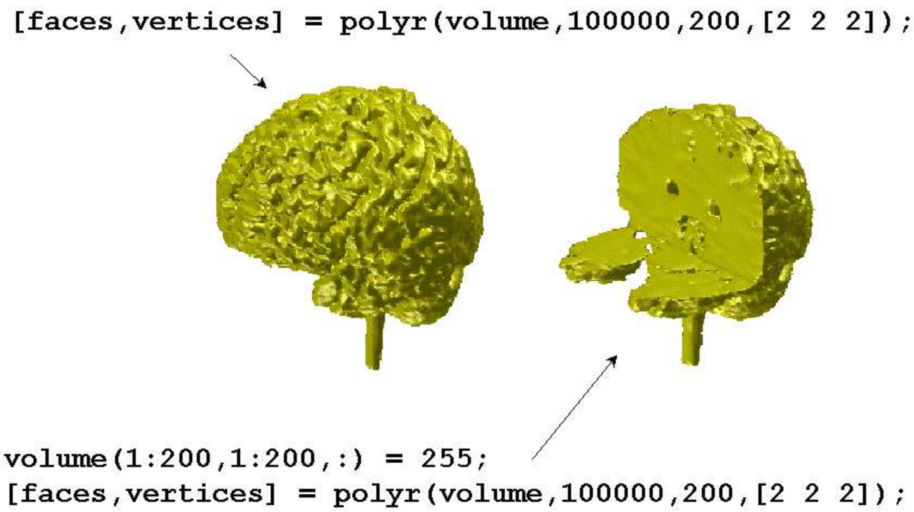


Figure 16: Cut-away of brain model produced by simple manipulation of the data volume.

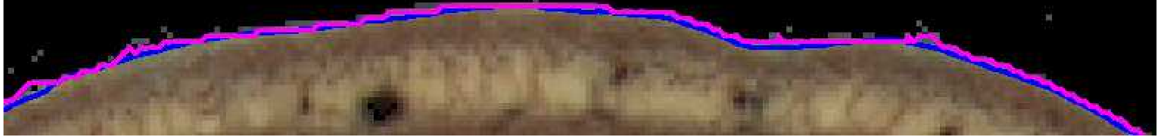


Figure 17: Trace done manually superimposed on trace produced by a trained neural network. The blue line is the manual trace and the purple line is the neural network trace.

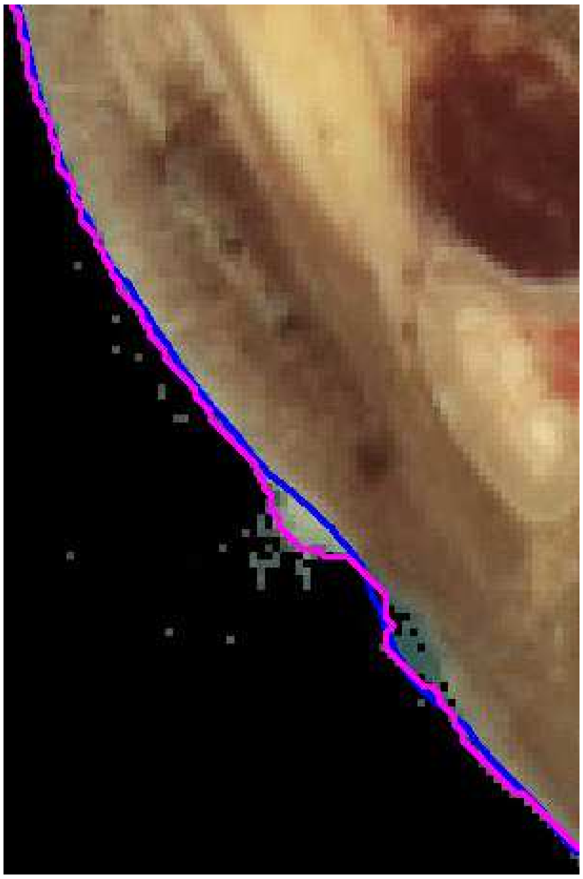


Figure 18: Manual trace (blue) is drawn through the uncommon skin artifact, but the neural network trace (purple) continues around the artifact outline.

Figure 18 shows that there are cases where the neural network follows a boundary too closely. Here the user wants to produce a smooth skin boundary that does not include the unusual artifact. The network, however, includes the artifact's boundary in its trace. With additional training and if the necessary input features are provided, the network can learn to duplicate the user's trace through the artifact.

The accuracy of a neural network trained to duplicate the tracing decisions of a human tracing the skin of the head is shown in Figure 19. Accuracy is measured by the RMS error in the distance (in pixels) between the points traced manually and the closest point traced automatically. There are a small number of points with high error, but most are within one pixel. The RMS error for the first trace is 0.87 (standard deviation of 0.75) and for the second trace the RMS error is 0.999 (standard deviation is 1.13).

This model of human-computer interaction can help capitalize on what both do best. Humans are good at visually identifying tissues of interest in an image, and they provide the initial guidance. Once a well-defined boundary is learned by the machine, it takes over the repetitive tasks of delineation across a series of slices.

6 Technology Transfer

The Visible Human Project, sponsored by the National Library of Medicine, has supplied Visible Productions with data sets of the male and female. These are the basis for the production of 3-D, computer-generated, anatomical models. The male data consists of 1,800 axial slices, cut at 1mm intervals, totaling 9 gigabytes of data. The female data set consists of 5,400 axial slices, cut at 300 micron intervals, and is a total of 39 gigabytes of data. Each image is 2,048 $\times$ 1,600 in full color.

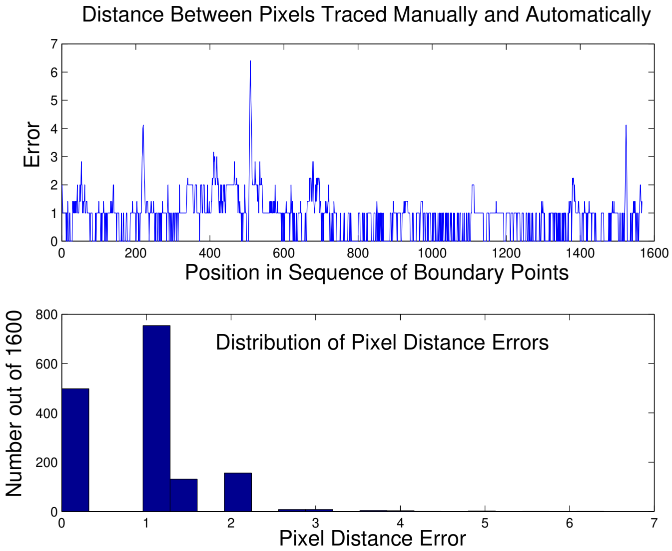
Visible Productions supplied the researchers at CSU with their images. Development of the semi-automated tracing method was conducted on the network of workstations in the Department of Computer Science at CSU. The algorithms used in the prototype of the semi-automated segmentation system was transferred to Visible Productions. The staff of Visible Productions are now integrating this algorithm into their segmentation software.

7 Networking

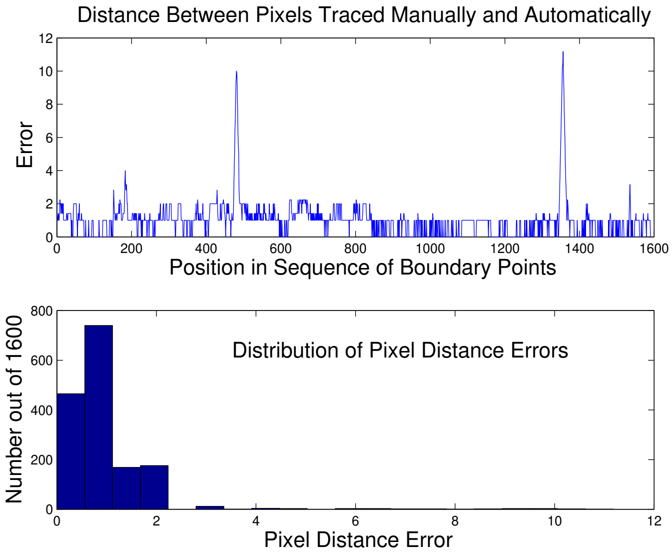
Our work on this project was presented at the following conferences.

- *Machine-Learned Assist for Boundary Contour Tracing*, NIH Second Visible Human Project Conference, Washington, DC, October, 1998.
- *Machine Learned Contours to Assist Boundary Tracing Tasks*, IEEE Southwest Symposium on Image Analysis and Interpretation, Tucson, AZ, April, 1998.
- *Semi-Automated Tracing and Visualization of Medical Images Using MATLAB*, MATLAB'97, Mathworks Conference on MATLAB, San Jose, CA, October, 1997.
- *Boundary Tracing of Medical Images for Three-Dimensional Model Development Assisted by Neural Networks*, Hewlett-Packard Labs, Palo Alto, CA, October 8th, 1997.
- *Neural Nets Facilitate Boundary Tracing Tasks in Medical Images*, IEEE Workshop on Neural Networks for Signal Processing, VII, Amelia Island, FL, September, 1997.

Papers and examples of our results are available online at Anderson's and Crawford-Hines' sites at www.cs.colostate.edu/~anderson and www.cs.colostate.edu/~sgcraw. This has resulted in contacts from a number of researchers. An example of a new contact in Colorado is Eric Kihn, a physicist at NOAA/NGDC in Boulder. He discussed with us the possibility of using our methods for segmenting



First Trace



Second Trace

Figure 19: Accuracy of boundary traced traced by neural network versus those traced by a person. The top graph shows the RMS error in pixel distance for each point traced during the tracing of a complete contour. The bottom graph is a frequency histogram of the pixel location errors. Most pixels traced by the neural network either match the human-traced pixels or are off by one pixel.

satellite photos of clouds to aid them in their work on fire monitoring, population studies, and other projects. We have some of his photos and are currently investigating this.

We have also discussed with D. Flanagan of SymSystems the possible use of our approach for helping them construct 3-D world models from satellite photos. They use these models in their flight simulations. Currently, they manually trace roads, rivers, and other features in the photos to create accurate 3-D models to be included in their simulated worlds.

8 Publications

Publications resulting from this grant are listed here.

1. Crawford-Hines, S., and Anderson, C.W. (1998) Machine Learned Contours to Assist Boundary Tracing. In *Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation*, Tucson, AZ, April, 1998.

2. Crawford-Hines, S., and Anderson, C.W. (1997) Neural Nets in Boundary Tracing Tasks. In *Neural Networks for Signal Processing, VII, Proceedings of the 1997 IEEE Workshop*, ed. by J. Principe, L. Giles, N. Morgan, and E. Wilson, pp. 207–215.

9 Follow-on Funding

As a result of the work funded by this CASI grant, the PI and collaborating company applied for and were awarded the following NSF SBIR Phase I grant:

- National Science Foundation, SBIR Phase I, $85,596, 1/1/98–6/30/98, with T. McCracken, Visible Productions, Inc., Fort Collins, *Complete Software System for 3D Surface Modeling of Anatomy from 2D Sections.*

We have also submitted a follow-on Phase II proposal this is currently under review:

- National Science Foundation, SBIR Phase II, $398,132, 6/1/99–5/31/01, with T. McCracken, Visible Productions, Inc., Fort Collins, *Complete Software System for 3D Surface Modeling of Anatomy from 2D Sections.*

References

Crawford-Hines, S., & Anderson, C. W. (November 1994). Interactive region bounding with neural nets. In *NNACIP'94—International Workshop on Neural Nets Applied to Control & Image Processing*, pages 58–61. Mexican Association of Automatic Control (AMCA) and IEEE.

Crawford-Hines, S., & Anderson, C. W. (1997). Neural nets in boundary tracing tasks. In Principe, J., Giles, L., Morgan, N., & Wilson, E., editors, *Neural Networks for Signal Processing VIII, Proceedings of the 1997 IEEE Workshop*, pages 207–215.

Crawford-Hines, S., & Anderson, C. W. (1998). Machine learned contours to assist boundary tracing. In *Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation*, Tucson, AZ.

Höhne, Pflesser, Pommert, Reimer, Schiemann, Schubert, & Tiede. (1996). A virtual body model for surgical education and rehearsal. *Computer*, 29(1):25–31.

Kass, M., Witken, A., & Terzopoulos, D. (June 1987). Snakes: Active contour models. In *Proceedings of the First International Conference on Computer Vision*, pages 259–268, London, England.

Kerr, J., Ratiu, P., & Sellberg, M. (January 1996). Volume rendering of visible human data for an anatomical virtual environment. In *Medicine Meets Virtual Reality: Health Care in the Information Age (Proceedings of Medicine Meets Virtual Reality 4)*, pages 352–370. IOS Press, San Diego, CA.

Özkan, M., Dawant, B., & Maciunas, R. (1993). Neural-network-based segmentation of multi-modal medical images: A comparative and prospective study. *IEEE Transactions on Medical Imaging*, 12(3):534–554.

Raya, S. (1990). Low-level segmentation of 3D magnetic resonance brain images - a rule-based system. *IEEE Transactions on Medical Imaging*, 9(3):327–337.

Visible Human Project, N. L. o. M. http://www.nlm.nih.gov/research/visible/visible_human.html.

A Intellectual Property Disclosure Form