DISSERTATION

MACHINE LEARNED BOUNDARY DEFINITIONS FOR AN EXPERT'S TRACING ASSISTANT IN IMAGE PROCESSING

Submitted by

Stewart Crawford-Hines

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Summer 2003

COLORADO STATE UNIVERSITY

30 June 2003

WE HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER OUR SUPERVISION BY STEWART CRAWFORD-HINES ENTITLED **"MACHINE LEARNED BOUNDARY DEFINITIONS FOR AN EXPERT'S TRACING ASSISTANT IN IMAGE PROCESSING"** BE ACCEPTED AS FULLFILING IN PART REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.

Committee on Graduate Work

_____

_____

_____

_____
Adviser

_____
Department Head

ABSTRACT OF DISSERTATION

MACHINE LEARNED BOUNDARY DEFINITIONS FOR AN EXPERT'S TRACING ASSISTANT IN IMAGE PROCESSING

Most image processing work addressing boundary definition tasks embeds the assumption that an edge in an image corresponds to the boundary of interest in the world. In straightforward imagery this is true, however it is not always the case. There are images in which edges are indistinct or obscure, and these images can only be segmented by a human expert. The work in this dissertation addresses the range of imagery between the two extremes of those straightforward images and those requiring human guidance to appropriately segment. By freeing systems of *a priori* edge definitions and building in a mechanism to learn the boundary definitions needed, systems can do better and be more broadly applicable. This dissertation presents the construction of such a boundary-learning system and demonstrates the validity of this premise on real data.

A framework was created for the task in which expert-provided boundary exemplars are used to create training data, which in turn are used by a neural network to learn the task and replicate the expert's boundary tracing behavior. This is the framework for the Expert's Tracing Assistant (ETA) system.

For a representative set of nine structures in the Visible Human imagery, ETA was compared and contrasted to two state-of-the-art, user guided methods – Intelligent Scissors (IS) and Active Contour Models (ACM). Each method was used to define a boundary, and the distances between these boundaries and an expert's ground truth were compared. Across independent trials, there will be a natural variation in an expert's boundary tracing, and this degree of variation served as a benchmark against which these three methods were compared. For simple structural boundaries, all the methods were equivalent. However, in more difficult cases, ETA was shown to significantly better replicate the expert's boundary than either IS or ACM. In these cases, where the expert's judgement was most called into play to bound the structure, ACM and IS could not adapt to the boundary character used by the expert while ETA could.

Stewart Crawford-Hines
Department of Computer Science
Colorado State University
Fort Collins, Colorado
Summer 2003

Acknowledgements

Chuck,
for optimistically sticking it out as my advisor for 10 years.

My committee, of course, Bruce, Ross, & David,
for their gentle pushes and clarifying commentary.

The Colorado Advanced Software Institute and the National Science Foundation,
for funding pieces of this work.

Eric Mortenson,
for his time in setting his Intelligent Scissors to our test.

My family,
who learned to stop asking, "When... ?"

Cheryl & Julie,
for their guiding hands from afar.

Just in time, I heard,
As I listened, attentive.
Thanks, Muse, for the song...

Dedication

John Day Crawford,
in your memory and honor.

Table of Contents

——== Chapter I ==——

Overview of Learned Expert Boundary Definitions: Context and Rationale

The motivation behind the work of this dissertation is to assist human experts in computer-assisted boundary tracing tasks by reducing the time required and increasing the accuracy and consistency of the boundaries they generate. Most systems addressing this problem to date have embedded in them the assumption that an edge in an image corresponds to the boundary of interest in the world. However, this is not always the case. There are images in which edges are indistinct or obscure, and these images can only be segmented by a human expert. By freeing systems of *a priori* definitions and building in a mechanism to learn them as needed, systems can do better and be more broadly applicable. This dissertation presents the construction of such a boundary-learning system and demonstrates the validity of this premise on real data.

The approach developed herein is an application of machine learning techniques to the definition of boundaries in imagery, as defined by an expert. Large sets of imagery will usually have a repetition and redundancy on which machine learning techniques can capitalize. A small subset of the imagery can be processed by a human expert, and this base can be used by systems to learn and semi-automate the processing task.

In this chapter, Section I.1 establishes the overall problem context of defining structural boundaries in large image sets and reviews the shortcomings of current methods.

Section I.2 summarizes the objectives, and Section I.3 poses the key issue of learning definitions from experts. Section I.4 presents the overall framework of the Expert's Tracing Assistant (ETA) system used in this study, and Section I.5 summarizes its comparison to manual tracing and to the state of the automated art. Section I.6 overviews the remaining chapters of this dissertation.

I.1 The Boundary Delineation Problem

The biomedical domain is a rich source of large, repetitive image sets. For example, in a computed tomographic (CT) scan, a common medical imaging modality, cross-sectional images are generated in parallel planes separated by millimeters. At a 2mm separation between transverse image planes, approximately 75 images would be generated in imaging the complete brain. Image sets such as this, generated along parallel planes, are called *sectional imagery*. Such sectional imagery abounds in medical practice: X-ray, MRI, PET, confocal imagery, electron microscopy, ultrasound, and cryosection technologies all produce series of parallel-plane two-dimensional images (see Mudry, et al., [2003] for overview and discussion of these imaging modalities).

Often experts are seeking to find, measure and characterize anatomical structures, and often these structures must be bounded on each of the images on which they occur. For example, in the generation of three-dimensional polygonal models from two-dimensional images, the process, illustrated in Figure I–1, follows these four steps:

1. for a particular structure (in this example, a cervical vertebra), trace the boundary of the structure on each image in a set;

2. triangulate between the boundaries defined on adjacent layers;

3. generate a polygonal mesh from all the triangulated boundary pairs;

4. then, finally, render the structure.

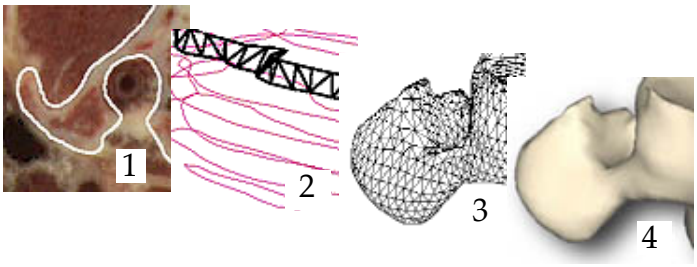


Figure I–1: Four steps in generating a polygonal model from sectional imagery: (1) define boundaries on individual images; (2) define triangles between adjacent boundaries that (3) generate an overall polygonal mesh; and (4) render with standard lighting models.

Steps 2, 3, and 4 are straightforward computations (Crawford and McCracken [2002]) and proceed without much user intervention; they require computation time on the order of seconds, or possibly minutes for large models. Step 1, however, is time intensive, requiring hours or days, or in the worst case weeks, of an expert's time to precisely bound a structure of interest. One initial motivation of this research is to reduce the amount of time required.

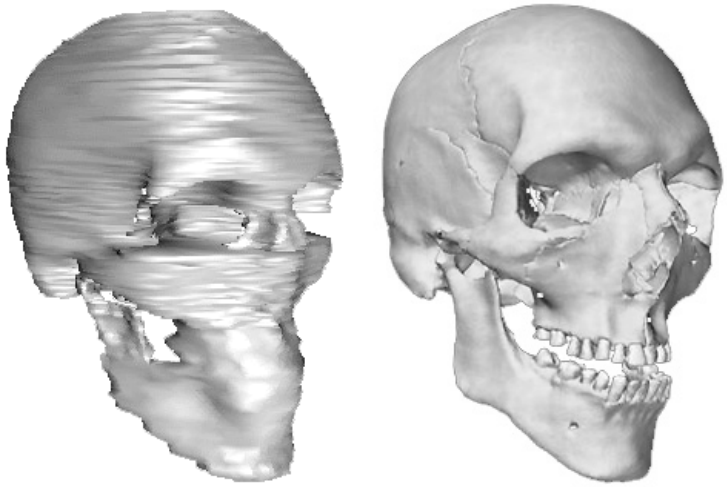


Figure I–2: Two skull models based on image boundaries created by (at left) automatic boundary detection techniques and (at right) an expert's manual tracing.

Not only is time an issue, boundary quality is also. The final quality of the three dimensional model is highly dependent on the initial boundary definitions. Figure I–2 shows a comparison of two different models. The rendering on the left was from a model created by Wang, et al., [1998] using active contour models as a

tool to automatically define the boundary of the skull in its image set. The rendering on the right was created from an expert's set of manually traced boundaries.

Durikovic, et al., [1998], note, *"Although 3D reconstruction is widely used in CT and MR imaging, the methods do not fulfill all the needs in anatomy. Anatomists seek information about the exact overall shape..."* Currently, the reference standard for high-quality outlining tasks is an expert's delineation of the region, and the state-of-the-practice is that boundaries are traced manually. Therefore, a second key motivation of this research work is to maintain the quality of an expert's manual tracing by learning from the expert's examples to better replicate what and how they trace.

I.1.1 Providing Expert Assistance

Generating three dimensional surface models of large structures requires bounding the structures across an image set, which is a repetitive, tedious, and error-prone process when totally done manually. Ozkan, et al., [1998] note the specific motivation of their work is to relieve physicians of the manual task of tumor tracing when planning a patient's radiosurgery. Providing viable assistance for such boundary tracing tasks will prove beneficial in several regards:

- the specialist's time can be significantly reduced;
- errors brought on by the tedium of tracing similar boundaries over scores of similar images can be reduced; and
- the automated tracing is not subject to human variability and is thus reproducible and more consistent across images.

The system developed, discussed, and studied in this dissertation is named the Expert's Tracing Assistant (ETA). The system was motivated and funded by organizations seeking these benefits. Among them, Visible Productions is a company generating highly accurate, three-dimensional polygonal models from two-dimensional sectional imagery for the educational and biomedical markets. They are working extensively with the National Library of Medicine's Visible Human Imagery [NLM 2001], a series of sectional photographic images (approximately 1600x2000 pixels each, 1850 for the male and 6000 for the female). Structures of interest must be bounded across a series of these images. In each, a precisely located, closed boundary around the structure is needed. Especially for large, well-defined structures, tracing the structure over many levels in these collections is both time consuming and error-prone due its tedious nature.

There has been much research directed toward automatic edge detection and image segmentation, from which a boundary outlines can be extracted. The following sections briefly discuss the cost of user adjustments and the assumptions implicit in these methods. The assumptions impact the ability of the system to adequately assist the expert on boundary definition tasks.

I.1.2 Costs of Adjustment

As an example, consider Figure I–3, from the Visible Male imagery. The upper image is a cross-section through the skull, and the lower two images illustrate standard edge detection operators on this cross-section. The lower-left image is tuned for strong edges, and there is little detail evident within the cortex. The lower-right image is tuned for weaker edges, to pick up the finer white–grey matter boundary; much of that detail is now visible, but at the expense of cluttering the visualization with all the other edges within the image. For this to be useful in capturing the finer detail in the image, the user must have an effective method to filter the spurious clutter.

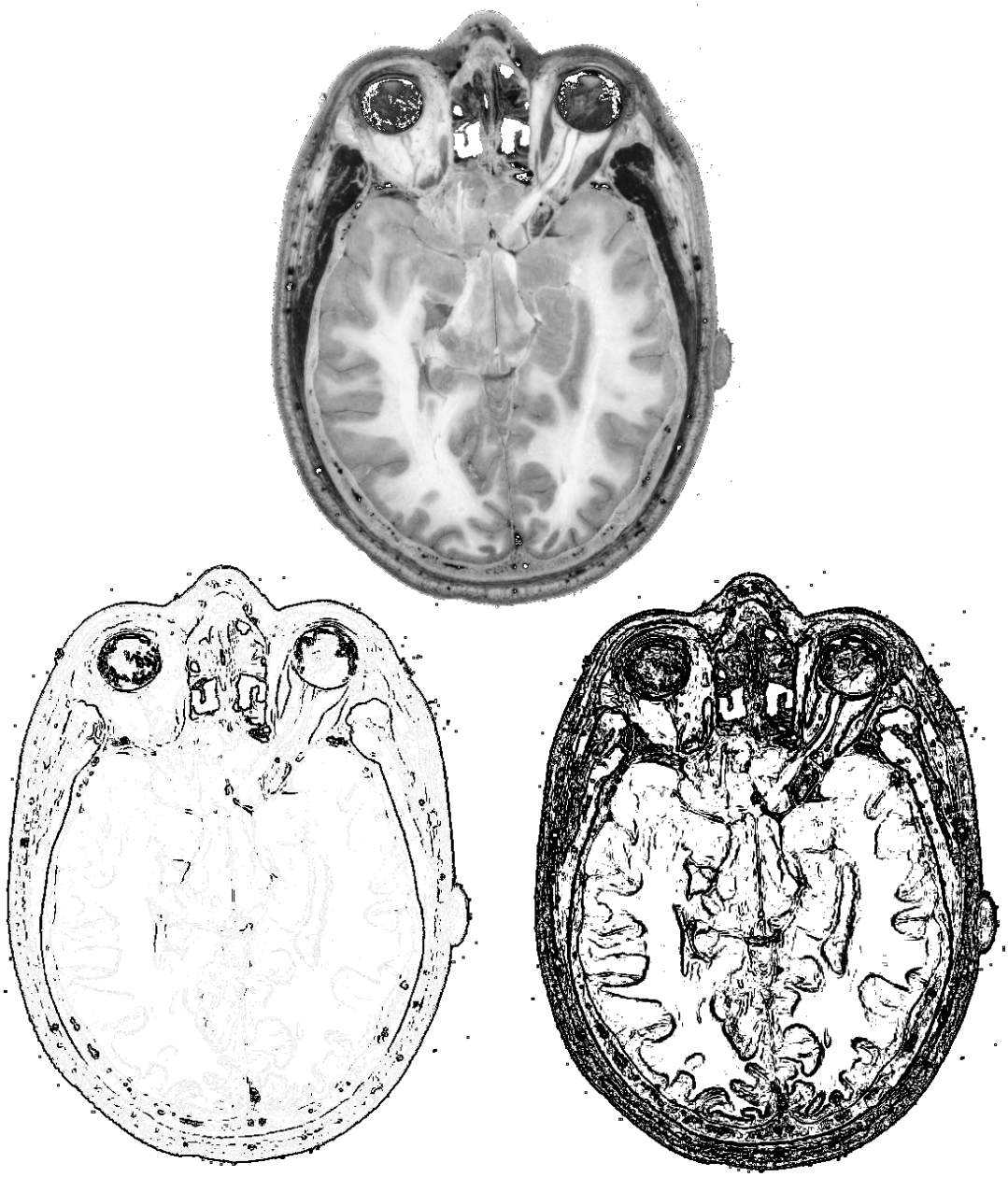


Figure I–3: A raw sectional image (top) through the skull at the level of the eyes. Edge detection image, tuned for strong edges (lower-left); same technique, finding weak as well as strong edges (lower-right).

Other user-adjustment costs arise with the active contour models (ACM) proposed by Kass, et al., [1987]. Briefly stated, an ACM settles into a stable state through an interactive relaxation algorithm (further details are in Chapters II and IV). Users have an opportunity to modify the results by adjusting the position and tension on "springs" which modify the boundary properties. Problematic situations for ACMs are boundaries with sharp corners or high curvature; users need to make adjustments similar to adjusting Bezier curve handles in drawing programs to handle the problems.

Such modifications typify problems of existing methods. No autonomous technique will completely give only the desired boundaries, except in the simplest cases. This implies that human intervention will be required to adjust intermediate results on images of any complexity. As suggested in Figure I–3, the task of removing clutter can be more costly than manually outlining the region of interest from the start. When users judge that the effort of adjustment outweighs the effort of manually running the cursor around the boundary to define the region, they will not bother with automated methods. For example, Cordier and Thalmann [1998] studied boundaries defined by snakes and shape-constrained deformable models; physicians were the actual system operators. Cordier noted during his conference presentation that after the study was done and the physicians knew how to use these tools, they continued to trace boundaries manually rather than making the adjustments necessary in the automated system.

I.1.3 Reliance on *A Priori* Definitions

An underlying reason why automatic edge detection and segmentation techniques have not been transferred into general practice for segmenting medical images is the implicit reliance of these methods on *a priori* edge definitions. Absent domain specific information, the best one can do is to make plausible, mathematically tractable

assumptions and proceed. For example, the boundary defined for a ramp edge, an edge blurred over several pixels, is *a priori* chosen to be the midpoint of the ramp or its point of maximal intensity change. In straightforward or synthetic imagery, techniques with these embedded assumptions work quite well. However the real world is rarely so kind. Real imagery can be confounded with noise, less-than-adequate resolution, and confounding artifacts of the image itself, such as similar tonal ranges for subject and background, or visually indistinct boundaries between structures, such as the connections of white ligament to white bone in anatomy. Fenster and Kender [2000a] note, *"In medical images, some structures are neither sharp edged nor reliably different in color from the surrounding structures. Thus, despite the research, some organ contours must still be outlined manually."*

However, within a limited domain, one can always do better with specific knowledge of the case at hand. Konishi and Yuille [2000], after noting that, *"Although there has been recent progress in general purpose image segmentation, it remains an extremely difficult problem,"* they then demonstrate that simple statistical knowledge of the domain powerfully aids segmentation.

In the case of sectional imagery, there are a large number of similar images. In this situation, one can take a small, representative subset of the imagery and by understanding it more fully, create domain-specific boundary definitions that will do better than any generic, general-purpose edge detection scheme. The redundancy in such image sets presents the ideal opportunity to use some few images as a learning base, and then capitalize on that learning to analyze the remainder.

I.2 Objectives

One overriding problem is that the boundary and edge assumptions limit the range of applicability of the methods in which they are embedded. The main premise of this dissertation is that by freeing systems of *a priori* definitions and building in a mechanism to learn them as needed, systems can do better and be more broadly applicable. This dissertation presents the construction of such a boundary-learning system and demonstrates the validity of this premise on real data.

The overall methodology followed is to develop a framework for boundary learning and tracing, and verify its adequacy on sample imagery (Chapter III). A comparison study, discussed in Chapter IV, is designed to compare this framework to other state-of-the-art, user guided methods. Each method is used to define a boundary, and the distances between these boundaries and an expert's ground truth were compared (Chapter V) . There is a natural variation in an expert's boundary tracing across independent trials, and this degree of variation serves as a benchmark against which these methods were compared. This work helps bring out the limitations of the framework and avenues for future work (Chapter VI).

I.3 Learning from the Expert to Improve Boundary Definitions

In beginning to address the problem of *a priori* definitions, it is important to distinguish between a ***boundary*** of a structure in the world, and an ***edge*** in an image of that structure. Standard practice is to equate a structural boundary with an image edge, where an edge is a discontinuity in the image and typically defined through some local measure of this discontinuity. In simple and straightforward cases, a boundary is coincident with an edge. However, this is not always the case.

The structure will always have a boundary, but that boundary will not always be evident as an edge in the image. There are many reasons why a boundary may not have an edge. For instance, the structure may be occluded in the world, the imaging modality may not capture the boundary as an edge, or the pixel characteristics of the structure may be identical to its surround and thus indistinguishable. In these worst case scenarios, a human can bring general knowledge or domain specific expertise to fill in the boundary where it is not evident as edge images.

This work acknowledges that there is a broad spectrum of cases between these two extremes, the straightforward and the deeply problematic, where there is some regularity in the image that can be used to locate a boundary. The basic tenet of this thesis is that this regularity can be learned, and systems which can learn these novel boundary definition patterns will be superior to those grounded in *a priori* edge definitions.

I.3.1 Example of a Learned Boundary

The following example illustrates the basic idea of how a learned boundary definition can overcome problematic situations where *a priori* boundary definitions fail. The upper half of Figure I–4 shows a raw image with a problematic section highlighted, and the lower half shows a training sample and a learned boundary definition. This image is a detail from a CT through a dog's legs, where the image parameters are tuned to highlight muscle tissue: the brightest area is muscle tissue, the grey surrounding it is skin, and the darkest untextured grey is the background.

The domain knowledge that humans bring to this situation of basic dog physiology: we know a dog leg has a mass of muscle surrounded by a layer of skin. The strongest boundary in the image is the exterior of the skin, where there is a strong contrast between the light grey skin and the untextured darker grey background. But in considering the

skin-muscle boundary, the boundary of interest is the less well-defined edge between skin and muscle that is several pixels inside the exterior of the skin. In the raw image of the upper leg, a weak edge can be seen a few pixels to the inside of the stronger external skin edge. However, in the circled area of the raw image, this weak distinction between skin and muscle is lost; without any domain knowledge of the image, one would be hard-pressed to accurately place any boundary there. When manually tracing the muscle/skin boundary, though, one can adjust for the ill-defined portion of the image and continue to trace a few pixels to the inside of the skin.

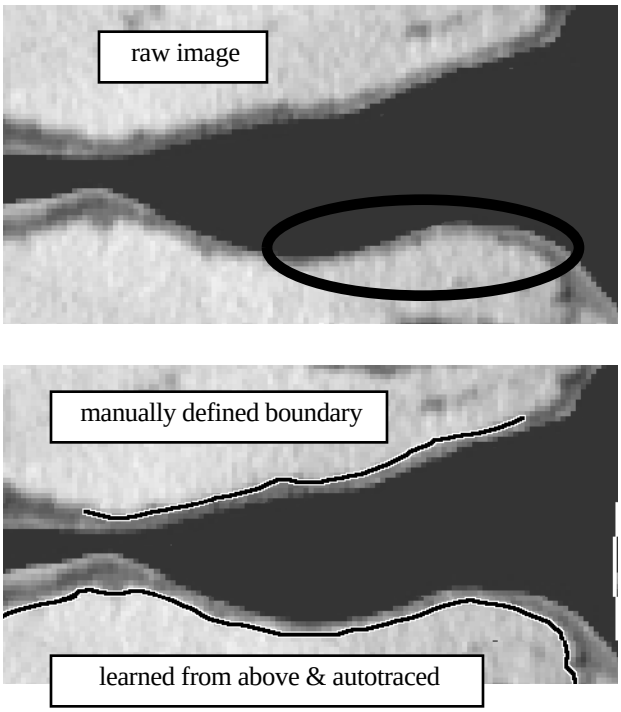


Figure I–4: Dog leg CT, tuned for muscle (brightest area) definition; the lower half shows a boundary segment used for training, and a learned boundary based on that training. The boundary is shown as a black line outlined by a thin white border.

The key elements that make this highlighted section troublesome to general automatic methods are (1) a boundary of interest that is weakly defined, (2) an image with inadequate resolution around that weak boundary, and (3) nearby strong edges, which confound generalized methodologies relying on local *a priori* edge definitions.

The lower half of Figure I–4 illustrates the benefits of a learned boundary definition for these muscles. The upper boundary was manually traced by an anatomist. Based on those characteristics, a boundary definition was learned and then the lower boundary was automatically traced based on that definition, using the ETA system outlined in

Chapter III. Note the learned boundary tracks cleanly through the indistinct area identified in the raw image, and this learned boundary definition was not confounded by the stronger, nearby edge.

The main point herein is that integrating learning into boundary definition tasks is crucial to their flexibility and situational usability. The choice of a particular learning methodology is not the focus of this research work, though for any system based on this work to be successful, the boundary tracing system must learn well, learn quickly, and be amenable to human override, or it will not be useful. The main thrust of this system is ***learning well***. There is a large body of knowledge on how to learn quickly, which focuses on the optimization of learning algorithms; that is not a research focus of this work. This work also makes no attempt to optimize a human-machine interface that facilitates easy human interaction to override the automated techniques when needed.

I.4 Framework for an Expert's Tracing Assistant (ETA)

In summary, the basic principles of a system to adequately assist in general boundary tracing tasks are the following.

- Learned boundary definitions will lead to more adaptable and more generally useful systems. *A priori* edge definitions are insufficient to precisely handle the spectrum of boundary contours found over a broad range of digital imagery.
- No matter how well a boundary definition might be learned, there will always be circumstances that will require correction by an expert user. Humans are good at perception within a larger context, and machines are good at repetitive work in a well-characterized domain. Systems are needed that capitalize on these disparate strengths: systems that move forward on their own in well defined territory, but also facilitate easy intervention and correction when needed.

ETA provides real-time learning and trace-ahead capabilities to assist experts in defining accurate structural boundaries. ETA follows these basic steps.

1) An expert manually defines representative boundary segments along a structure of interest.

2) These representative segments are used to construct a set of positive and negative exemplars, which in turn is used with a supervised learning methodology to learn the pattern that characterizes the boundary.

3) The system can then automatically extend a partially defined boundary when it is statistically confident in its choices. The user monitors the boundary-in-progress and corrects any mistakes or takes control when the system's statistical confidence measures are low.

The framework outlined has had several progressively better software implementations from 1995 through 2001, starting with a research prototype and then later developed for biomedical applications through grants from the Colorado Advanced Software Institute (Anderson [1999]) and from the NSF's SBIR program (McCracken [1998] and McCracken [2001]). In partnership with Visible Productions, the results of this approach began to be quantified.

Through application, it was seen that interactive systems that learn boundary definitions to suit specific situations can both improve an expert's tracing speed and generate more consistent and accurate boundaries. As an example, the Visible Male's skin boundary is represented on all 1,800 images in the cryosection set. Due to the large boundaries required on each image, the uniformity of these boundaries, and the total number of images, the skin is incredibly tedious to trace manually in its entirety. Those

who manually traced the skin the first time estimate it took three to four staff-weeks to complete. Because of inconsistencies in the skin model developed from this tracing, experts retraced the skin of the Visible Male using an ETA implementation integrated into their production tracing system. With the automated assist, the skin was retraced in three staff-days. These new skin boundary contours are used in Visible Productions's current models, since they are of better quality and more globally consistent than the prior set of skin boundaries.

The details of the issues, considerations, and experimentation that went into the construction of ETA are presented in Chapter III. This system makes no claim to be optimal; it stands as a vehicle to explore the validity of the premises in this thesis.

While comparison study of this dissertation is made within the rich set of Visible Human imagery, this work is applicable in other domains as well. Fast outlining of structures in MRI imagery by an expert can lead to three-dimensional models created in close to real time. Imagery in initial experiments was aimed at helping understand mental illness by delineating brain structures and studying their associated volumes. It is are also applicable to aerial imagery, in diverse applications such as road following and cloud boundary delineation. And techniques similar to this, specifically the Intelligent Scissors of Mortensen and Barrett [1995, 1998], have found a home in general purpose commercial products such as Photoshop.

I.5 Comparing ETA to Experts and Other User-Guided Methods

With this viable framework established, a fundamental issue is how well ETA compares to the manual boundary tracing of an expert. A basic hypothesis of this work is that the system can learn specific peculiarities of an expert's boundary tracing behavior and

replicate that behavior. To quantify this issue and verify the hypothesis, boundaries created by both an expert and ETA are directly compared.

Once a boundary definition is learned, ETA is totally deterministic and the tracings it performs are reproducible. This stands in contrast to humans, who will exhibit some variation when manually tracing a boundary. Brahmi, et al., [1999] note that boundaries drawn by experts may display substantial variation, especially in areas where contrast is poor. Karayiannis and Pai [1999] also note inconsistencies among experts in a complex segmentation problem. ETA should thus not be required to exactly replicate an expert's boundary trace to be successful, but the learned boundary should fall within the range of variation for either a specific user or across a group of users. Experiments presented in Section V.1 look at the variation within boundary definitions made by the same user at different times, and demonstrate that ETA is consistently within the range of this intra-user variance.

Given that ETA can replicate an expert's manual boundary tracing, how does this compare to the ability of other techniques, already known and realized in image processing, to assist an expert? The key hypothesis of this work is that by learning a representation to match the expert, systems can improve on the state of the art. This will be demonstrated in a comparative study of ETA with two methods currently representing the state of the art and practice.

To briefly summarize an example comparison, consider the user-guided method of the Intelligent Scissors of Mortensen and Barret [1998]. With this tool, the user places an initial seed point on a boundary, and a continuous, single-pixel-wide path is formed from that seed point to the current cursor location.

The left half of Figure I–5 shows a detail of the boundary defined by this intelligent scissors tool for an MRI image. To outline the weak edge just inside the stronger outside edge, the contrast sensitivity of the tool is set very low (a high setting implies sensitivity only to strong edges), and the distance parameter is set small (which means the boundary definition stays close to the cursor as the user loosely traces out the boundary). In general, the tool roughly follows the desired boundary, but the boundary is seen to jitter when a strong edge runs near a weaker edge, which is problematic when the weaker edge is the boundary of interest.

The right half of Figure I–5 shows the same image detail, with the pixels chosen by ETA. The smoother line in the lower part of this image is a piece of the training segment, while the line defined by the trained neural network is stair-cased, since the points selected are on pixel centers. In this case, the system effectively learned to follow only the weak edge and ignore the strong neighboring edge. Note the automatically traced edge is consistently within one pixel of its training exemplar when it traces over it.

Another user-guided methodology is that of Active Contour Models. Chapter IV presents a methodology for comparing Active Contour Models, Intelligent Scissors, and

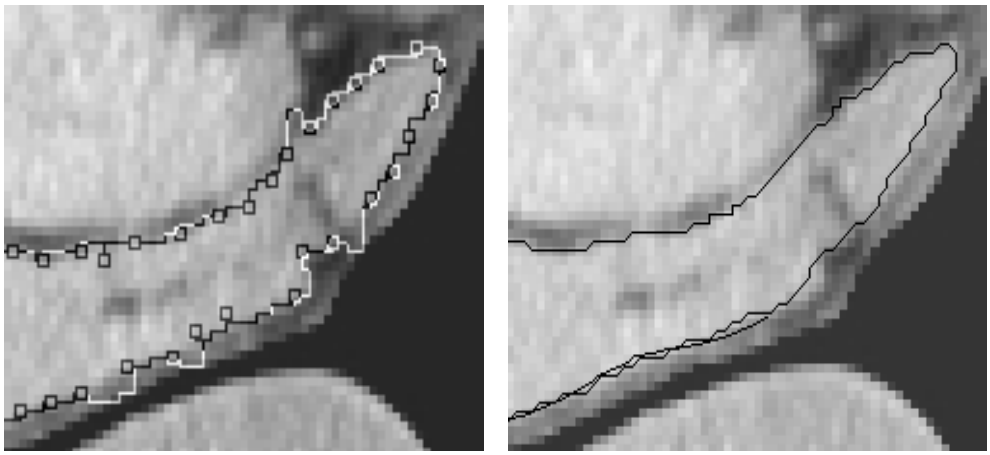


Figure I–5: Defining a boundary with Intelligent Scissors (at left) and with the Expert's Tracing Assistant (at right).

the Expert's Tracing Assistant, on a representative set of boundary definition tasks in the Visible Human imagery. Measures are derived to quantify the differences between the boundaries by selecting a tolerance such as one pixel and stating, *"The boundary curves are within one pixel of each other 86% of the time"* or by selecting a percentile such as 90% and stating, *"The boundary curves are within 1.1 pixel of each other 90% of the time"*. These are the key measures of the differences between the boundaries. Chapter V presents the results of the comparison.

I.6 Overview of the Remaining Chapters

Chapter II surveys recent and classical work in boundary definition tasks, organized along two independent considerations. The first consideration is whether the work deals with boundary definitions that are defined *a priori* or whether the definitions are learned. The second consideration is whether the approach is interactive or autonomous. Chapter II reviews recent research in the field, based on the extent that the methodology is either *a priori* and autonomous (Section II.1), learned and autonomous (Section II.2), *a priori* and user-guided (Section II.3), or learned and user-guided (Section II.4).

Chapter III details the engineering of the ETA system to sufficiently implement the ideas presented. The details of sampling a local neighborhood in order to create an adequate training set are covered. Experiments used to come up with an appropriate neural network architecture are presented. And the question of re-representing the input space to facilitate better and faster learning is explored.

Chapter IV presents the methodology used to compare Active Contour Models, Intelligent Scissors, and the Expert's Tracing Assistant, to experts on a representative set of boundary definition tasks in the Visible Human imagery. Chapter V details the

results of that comparison study, and the relative merits of the different approaches. Finally, Chapter VI provides a summary of this work and its conclusions, and a discussion of limitations and future work.

——== Chapter II ==——

Background

The basic tenets of this dissertation are:

- that *a priori* edge definitions are insufficient to handle the spectrum of contours found over a broad range of digital imagery, and

- that no matter how well a boundary definition might be learned, there will always be exceptional circumstances that will require some correction by an expert user.

The phrase *a priori definition* is used specifically here to denote a definition that is based on assumptions about, rather than driven by data from, an image and the world it represents. For example, defining an edge to be the location of the extrema of the first derivative of an image intensity function is a definition based on tractable mathematics. It is a definition which can be made *a priori*, i.e., before consideration of any specific image. Research based on *a priori* edge definitions will first define edges mathematically, then process imagery to generate edges based on those definitions, and finally (possibly implicitly) relate those edges to boundaries in the world.

In contrast, research based on learned edge definitions starts with boundaries in the world which are mapped onto images, then from the labelled imagery edge definitions

are constructed. At the risk of oversimplifying, *a priori* methodologies start with definitions then move to data, while learned methodologies start with data then move to definitions. In practice, these are two ends of a continuum, and there are many possible methodologies in the middle combining both *a priori* and learned aspects.

This chapter explores recent and classical published work directed toward identifying boundaries in images. Such work is mostly found within the literature of edge detection or image segmentation. Since this literature is vast and spans decades, the works discussed in this chapter were selected either (1) as a sampling of recently published work or (2) because they are classically cited works. The research represented is categorized in regard to two considerations: (1) whether the work relies on *a priori* edge definitions, or it incorporates learning of an edge definition; and (2) whether the system is intended as autonomous, or if user assistance is easily incorporated when the edge definition becomes insufficient to adequately define boundaries. This review is organized into four subsections, based on the extent that the methodology is: *a priori* and autonomous (Section II.1); *a priori* and user-guided (Section II.2); learned and autonomous (Section II.3); or learned and user-guided (Section II.4).

II.1 *A Priori* and Autonomous Systems

An edge in an image can be characterized by changes in image brightness or color in some local area of the image. When brightness (either overall, or in some channel) is represented as a function, these changes are also represented in the function's derivatives. What is referred to here as the "*classical*" approach to edge detection is the study of these functions and derivatives, for example locating edges at first derivative extrema that identify where intensity change is most abrupt, or at zero-crossings of a second derivative that identify those first derivative extrema. Chellappa's tutorial

[1992] summarizes these classical ideas. This section is by no means an exhaustive review of classical edge operator research. Rather, its intent is to identify the main themes of the work, and to identify common assumptions and problems.

Since image pixels are quantized and usually represented at integer coordinate values, derivatives are realized as differences across pixel values. The simplest first derivative operator creates a new edge image E, where each edge pixel is the difference between neighboring pixels in the original image.

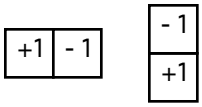


Figure II–1: First-difference filters, approximating a first derivative over an image, are used to identify vertical edges (left filter) and horizontal edges (right filter).

The first difference horizontally, $E_x(x,y) = I(x,y) - I(x+1,y)$, quantifies the strength of vertical edges in the image; the analogous difference in y, $E_y(x,y) = I(x,y) - I(x,y+1)$, quantifies the strength of horizontal edges. These differences are represented graphically in Figure II–1. To use these graphical visualizations, the boxes of a filter are conceptually placed over a corresponding field of pixels, the coefficients in the boxes are multiplied by the intensity values of the pixels beneath them, and the results are summed together. For the horizontal filter on the left side of Figure II–1, +1 is multiplied by a pixel value at (x,y) and -1 is multiplied by the pixel's right neighbor which is (x+1,y), which yields the above equation $E_x(x,y) = I(x,y) - I(x+1,y)$. These operators are called by several names: masks, filters, convolutions, or kernels.

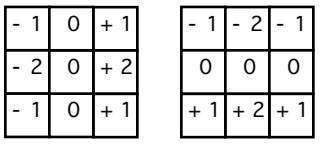


Figure II–2: Sobel filters will smooth vertical edges (the left filter) or horizontal edges (the right filter).

One problem with derivatives is their sensitivity to noise or texture in an image. The Sobel operators, illustrated in Figure II–2, are one example of a class of operators that attempt to reduce the effect of noise or texture by averaging

pixels in a direction parallel to the edge being detected. These two masks can be thought of as determining the horizontal and vertical components of an edge going through the center pixel; standard vector calculations can be used to calculate the edge's orientation and magnitude. A pixel is determined to be an edge pixel if the resultant filtered value is above some threshold.

Zero-crossings of second derivatives indicate extrema of first derivatives, and thus also are useful tools for edge locating. Figure II–3 shows a prototypical filter of a second derivative operator, the Laplacian. This operator is isotropic, i.e., it is non-directional. Since it works for edges at any orientation, it thus avoids the issues of combining independently determined horizontal and vertical components into a total edge strength measure. A pixel is determined to be an edge pixel if it represents (or is close to) a zero crossing; an edge pixel will have a positive value of the Laplacian operator to one side and a negative value on the other side. This criteria is more complex than the simple thresholding required for a first derivative mask. Second derivative measures are also very sensitive to noise and texture in an image.

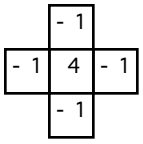


Figure II–3: A simple second-difference filter approximates a second derivative.

A Gaussian smoothing of an image is often used in conjunction with the differential operators to reduce the effects of noise and texture. Canny [1986] derived a step edge detector, which is close in form to the first derivative of a Gaussian operator. Marr and Hildreth [1980] proposed using the Laplacian of Gaussian (LoG) convolution mask, also known as the "Mexican hat" function for its distinctive shape. The LoG can be closely approximated by a Difference of Gaussian (DoG) functions, using Gaussians with differing spread parameters. This DoG mask is computationally more attractive since the Gaussian is separable and can be realized as successive row and column operations.

Huertas and Medioni [1986] fit a surface model to the LoG response, and then interpolated sub-pixel accuracy in identifying edge locations.

Reichenbach, et al., [1990] studied optimal small kernels for edge detection. Their filter, shown in Figure II–4, is a 5x5 kernel set to identify structures on the order of one pixel wide.

0.02	-0.04	-0.13	-0.04	0.02
-0.04	-0.24	0.06	-0.24	-0.04
-0.13	0.06	1.48	0.06	-0.13
-0.04	-0.24	0.06	-0.24	-0.04
0.02	-0.04	-0.13	-0.04	0.02

Figure II–4: A filter designed for identifying single-pixel-wide structures.

Gaussian filtering is a low-pass filtering, where the larger the spread of the Gaussian, the lower the pass band of the filter. Such filtering can minimize the effect that noise and texture have on derivative operators, since the Gaussian blur removes the high-frequency components of an image, the characteristic frequencies of noise and texture. Unfortunately, the high frequency components of a signal are necessary to locate an edge precisely in space. Canny [1986] demonstrated the uncertainty principle inherent in these classical edge operators – the tradeoff in performance between edge detection and edge localization. Even Gaussians of modest width may wipe out the information of interest, for example in a fingerprint image where ridge lines are closely spaced.

As an alternative to derivatives and their associated noise problems, Wang and Jenkin [1992] transform the image using complex Gabor filters. In this complex space, they use phase, amplitude, and frequency information to identify both edge and bar primitives in images. The first four steps they outline provide a classic example of an *a priori* methodology: (1) define a mathematical model of a feature (edge or bar); (2) choose a proper filter; (3) filter for the feature; (4) derive phase properties that identify the feature. The tolerances around the properties can be tuned to very specific

characteristics, for example bars of a particular width can be identified without also identifying wider or narrower bars. The basic assumption, however, identified early on in the paper, is that, *"significant scene structure is often associated with sharp changes in image intensity"*.

Haralick [1985] approaches edge detection from a signal processing perspective, where the image is thought of as a signal in combination with additive noise. He proposes the use of facet models to filter out noise; a surface is fit to a neighborhood using a least squares fit, and the characteristics of the (more regular) fitted surface are used instead of the (noisy) pixel intensities to find edges. Statistical techniques are then used to determine, at some degree of confidence, whether a pixel is an edge pixel or not. While the mathematics behind this work are strong, so are the assumptions, including uniform image areas, step edges, Gaussian additive noise, and independent, identically-distributed noise.

In summary, classical edge operators raise the following issues.

- Noise or texture in a scene generates high frequency events in an image; derivative operators are sensitive to this noise and texture, generating clutter in the edge images.

- Limiting the effect of noise or texture implies reducing the high frequency components of the image, which in turn implies a loss of edge localization.

- There is a natural uncertainty principle between detection and localization of an edge; the better the detection, the worse the spatial localization.

- Many of the analytical performance evaluations of these operators were based on "ideal" edges.

- Many operators exist; which is appropriate?

Some key assumptions are implicit in these classical edge operators. One is that an edge is identified with the steepest change in the image intensity profile. This sounds reasonable for the image alone, but may not map to an actual boundary in the real scene, due to the imaging process and the interpretation of other viewers and experts. Also, the idea of thresholding a first derivative operator reflects the assumption that the most significant edges in an image are the strongest ones. This assumption is sometimes wrong – in Figure I–3 for example, the grey matter / white matter boundary is significantly weaker than neighboring skull. To generally identify crisp, continuous, single-pixel width boundaries in an image, more assumptions or additional information beyond pixel intensities is needed.

II.2 *A Priori* and User-Guided Systems

This section covers research where the user is incorporated into the system to interact and guide a boundary definition process, though still making strong *a priori* assumptions, possibly specific to a certain domain. The Road Followers discussed in Section II.2.1 have a human in the loop, delineating roads in aerial imagery. Active Contour Models, discussed in Section II.2.2, involve users in initializations and adjustments. Intelligent Scissors, discussed in Section II.2.3, require a user to loosely guide a boundary wire. Miscellaneous user interventions are covered in Section II.2.4.

II.2.1 Road Followers

One specialized line of research has focused on the tracking of roads in aerial images. In the general methodology, a user starts by identifying an initial road segment and the automated method "follows" the road from that initial seed. A comparison study by

McKeown and Delinger [1988] broadly divides the road-following research into three categories: region-based followers; correlation trackers; and edge linkers.

A ***region-based*** road following heuristic grows the road along its length based on some characteristic (e.g., pixel intensity or local texture) of the initially defined segment. From its initial seed, a road may be followed in the manner of a flood fill algorithm: areas of similar intensity or texture bounded by a contrasting edge are assumed to be road, and added to the existing segment. Alternatively, areas may be identified across an image and then linked together.

Region-based followers depend on the fact that a road generally has a relatively uniform texture and intensity in the aerial imagery, as contrasted to its surround. This criteria can potentially be satisfied by selecting an appropriate image modality. For example, Bajcsy and Tavakoli [1976] used only one band of LANDSAT-1 imagery, presumably since the contrast of road and surround in that band was greatest. Sometimes the image modality available does not provide a sufficient distinction, however. For example, Airault, et al., [1994] show a histogram of their region-homogeneity criterion, where both roads and fields have a peak at the same value, thus making them hard to distinguish from each other.

In the best case, when a road is generally distinct from its surround, region-based followers run into problems whenever the uniformity of road or distinctness of boundary is interrupted. Problematic situations include road occlusion (overpasses and obstructions), shadows crossing a roadway, vehicles on the roadway, changes in road surface (e.g., a transition of asphalt to cement), and changes in boundary contrast. Due to these practical limitations, region-based heuristics are not used in isolation, but possibly in combination with other methods.

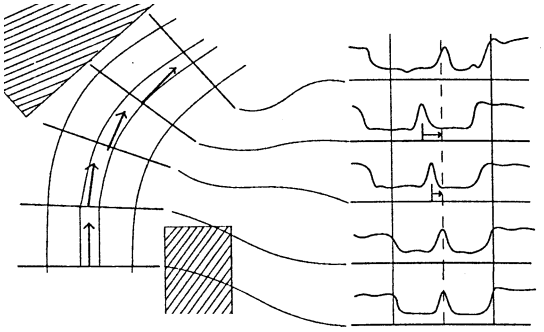


Figure II–5: A Correlation Tracker: road at left and profiles to the right. (from McKeown and Delinger [1988])

A ***correlation tracker*** heuristic relies on an intensity profile of the road and its adjacent area, in a direction perpendicular to the road, as shown n Figure II–5. Starting from an initial seed, a road segment is extended by first projecting the road ahead, and then adjusting the placement of the road's center point so that the new perpendicular intensity profile correlates well with the road's established profile. An indistinct edge of a road will cause problems for a region-based follower, however the correlation tracker uses other information in the profile to establish the best placement for the road center. This technique was used by Quam [1978] in SRI's HAWKEYE road expert system.

A surface intensity correlation tracker relies on several assumptions about characteristics of a road to work well: a constant width, a relatively constant (or possibly a slowly changing) surface intensity profile, and a slowly-changing direction over a reasonably predictable path. To the extent that these assumptions are not met, other methods need to be employed to follow the road through problematic areas. Large-scale occlusions and anomalies need to be handled through other procedures.

An ***edge linker*** heuristic is based on linking identified edges together to create road-like objects. Consider the basic example of a straight, dark road on a light background. It has two distinct edges, parallel to each other and equidistant from each other. A directional edge operator will label the edges as opposite in direction, and they are thus called *anti-parallel edges*. Generically, an edge linking heuristic involves these steps: (1) an edge operator is run over an image (or some part of it) and "significant" edges are

identified; (2) edges are grouped into anti-parallel pairs in accord with some separation criteria; and (3) pairs are geometrically extended to join other "likely" pairs into longer road segments.

The work of Nevatia and Babu [1980], Zhu and Yeh [1986], and Vasudevan, et al., [1988] follow these lines. Some problems noted are the difficulty of grouping anti-parallel lines properly, and the non-consistency of contrast between a road and its border. The main difference among them is the methodology used to join independently identified segments: Vasudevan, et al., detail an intricate sequential algorithm to cluster the segments; Zhu and Yeh provide a set of production rules implemented in a variant of OPS5.

There is much to specify in the steps above: how are "significant" edges defined, and what separation criteria are used? These parameters may vary, depending on whether straight, multi-lane highways or twisty, rural roads are the subject of interest. The same assumptions about road characteristics, as discussed for correlation trackers, underly this work; additionally this work relys on the presence of a significant number of straight road segments to be initially identified.

Attempting to overcome the limitations of these individual methods, the ARF (A Road Follower) system developed by McKeown and Delinger [1988] used multiple lines of analysis in parallel. They surveyed the best heuristics then available, and built a system using both an edge linker and a correlation tracker working off common data structures. They demonstrated the ability of the somewhat complementary methods to help overcome the shortcomings of each. As a distillation of best-available techniques used in combination, their work represents a culmination of much of the prior decade's work. The following paragraphs summarize the key aspects of their system; underlined phrases indicate tunable parameters of the system.

The correlation tracker (which they call a "surface tracker") used is taken straight from Quam's SRI work. The basic procedural steps of this process are:

1) project the road 1-step ahead; this is based on a quadratic extrapolation from recent past points (the step size is not the same as the pixel size);
2) extract a cross section of the road at the projected point perpendicular to the direction; use bi-linear interpolation for non-integer pixel values;
3) "cross-correlate": evaluate various lateral offsets of this cross-section to the reference cross section; find the smallest sum-of-squares difference;
4) generate a mask for the cross-section where actual values differ from expected values by some threshold;
5) re-correlate, as in step #3, using only the unmasked elements (to reduce the effects of non-road elements on the correlation);
6) if the correspondence is "good", use the new offset to determine the next road center point, update the cross-section model (using an exponential decay), and iterate the process.

When the correspondence is "not-good", the system attempts projections further than 1-step ahead, based on the assumption that some transient obstruction has caused the cross-section correspondence to fail. If the system finds a matching road within some reasonable look-ahead, it is assumed to be the road's continuation, and missing points are interpolated. A change of road-surface may require the system to be restarted with a new cross-sectional model.

The edge-based tracker they used follows the work of Nevatia and Babu [1980], tracking the edges of the road by linking points with both high gradient and consistent orientation. The basic procedural steps of this process are:

1) project the road 1-step ahead using a quadratic extrapolation;
2) compute a 5x5 Sobel operator (giving both edge strength and orientation) for points along the perpendicular to the road direction;
3) score each point (on a [0,1] scale) based on edge strength, orientation, and difference from neighboring point's edge strength and orientation;
4) calculate a weighted sum of these component scores;
5) choose the highest-scoring point (which exceeds some threshold) as the next edge point;
6) mark the next road point as the midpoint between the two edges.

If only one edge is successfully found, assume the road width is unchanged and mark the next road point. If no edges are successfully found, guess ahead possible points as previously noted.

These two road tracking mechanisms are ARF's low-level analyses. An intermediate level of analysis attempts to automatically restart one mechanism when it fails from the other. For example, a change of road surface from asphalt to concrete will generate a failure in the correlation tracker (which relies on surface characteristics), but if edge strengths remain strong the edge tracker will provide the continuation needed for the correlation tracker to automatically restart its road model. Differences between the methods and differences between the expected and actual surface models are used in attempts to identify road width changes, intersections, vehicles, overpasses, surface changes, and other possible occlusions.

McKeown and Delinger analyzed the viability of this system on a series of aerial imagery of highways. Some images were used to tune the system (manually adjusting the parameters identified above), and others to test it. The two methods were able to

compensate for one another to some degree (30% of the confusions were automatically resolved in their documented test case), many obstructions and intersections were processed appropriately, and the exponentially-damped road profile was robust to gradual changes of road character. System failures were characterized by situations where a road had:

- high curvature,
- similar intensity to its boundary material,
- severe/sudden width changes,
- curvature which is not smooth or reverses direction (e.g., an "s" curve),
- inconsistent road-surface pattern, or
- combination effects (e.g., an obstruction on a curve).

These failure modes are certainly not suprising, in light of the key assumptions behind all these road tracking methods. Each road follower starts from some *a priori* definition of a **road**, which usually results in these assumptions:

- road topology: constant width and (anti-) parallel edges,
- relatively slow changes in direction,
- separable characteristic of road from boundary material (intensity, texture), and
- high-contrast road boundary.

Some intervention is expected upon failure, and no further learning is applied to adapt to exceptions. The interactive nature of these systems is similar to the Expert's Tracing Assistant; many of the problems encountered by the road followers will be avoided in the ETA by minimizing assumptions such as those listed above.

II.2.2 ACTIVE CONTOUR MODELS

The name *Active Contour Models* refers to a general set of models, also called *Deformable Models* or *Snakes*, published by Kass, et al. [1987]. An ACM is an energy minimizing spline, which is initialized close to a structure of interest and then settles into a local energy minimum over a course of iterations. The energy function is defined so that minima correspond to boundaries of interest in the imagery. Since the initial contour is closed, the final result will always be a closed, continuous curve. These methods have been applied in segmenting, matching, and tracking anatomic structures in medical imagery. McInerney and Terzopoulos [1996] summarized the early development and application of these ideas.

In these deformable models, four basic things need to be defined: (1) a shape model, either a single continuous curve or a combination of segments; (2) an energy function to minimize, that combines the shape energy (penalizes undesirable shapes) and image energy (responds to strength and proximity of edges or gradients); (3) an optimization technique, such as gradient descent; and (4) an initial, rough boundary specification.

The contour is represented parametrically by a point-valued function and the shape of the contour is defined to have an energy which is primarily the sum of two terms. The first term is a measure of the geometry of the contour's shape and the second term is derived from the character of the image pixels on which the contour is superimposed. Further details of ACMs are given in Chapter IV.

Active Contour Models are guided by the user in several ways. An initial contour needs to be defined in the beginning of the modelling process. At the end of the process, after the contour has settled into its equilibrium state, a user may specify adjustments to the contour's final resting position.

Mao, et al., [1999] studied the robustness of equilibrium states of ACMs with respect to their initialization. They used ACMs to outline arterial lumen. Since lumen are approximately circular, the contour is initialized with a circle which an operator defines by specifying the center of the circle. They use a discrete dynamic contour model which is a polyline connecting a set of discrete vertices.

The center point is varied over a 15x15 matrix of seed points. For each seed point, the active contour converges to a solution, and a binary image is generated with the interior of the region set to 1 and the exterior set to 0. These 225 binary images are added together – if they were exactly alike, the summed image would exhibit a sharp edge, however the summed image has soft edges, with the degree of softness or slope indicating variation in the final shape. Some of the resultant shapes were dramatically different. The variability of the final contour was studied over several images, the mean difference among contours being within two to four pixels with a standard deviation between two and three. This large a standard deviation with respect to the mean implies a long tailed distribution, thus there were a significant number of places where the contours differed by ten or more pixels.

Gill, et al., [1999a] did similar work for three-dimensional surfaces rather than two-dimensional ACMs. They looked at the variation among equilibrium meshes. The variation measure used was the standard deviation of the final surfaces, which ranged from 0 to 1.3 mm. Unfortunately, the voxel size is not mentioned in the paper, so there's no easy way to interpret the surface error in terms of pixel/voxel dimensions.

To overcome this sensitivity to initialization, Ladak, et al., [2000] used an anatomically-derived initial shape model specific to the domain. Problems they note when applied to prostate ultrasound images are that the contrast between tissues is low and dependent on the system's transducer and ultrasound frequency, and that speckle, shadowing, and

refraction artifacts exist. These problems make the initial contour specification crucial for the correct convergence of an active contour. To solve this, the operator specifies four specific, anatomically defined points on the boundary in the ultrasound image, and prostate-specific shape info is then used to create the starting outline. The system may still settle into a wrong shape. The user can then drag a point onto the boundary, clamp it, and let the contour re-deform. In 36% of the cases, the first settling of the contour was good enough, while 51% required one editing operation, and 13% required two or more editing operations.

Distance-based and area-based metrics were used to compare the system's results to an expert's tracing. Distance differences were calculated along rays from a centroid (this will not evenly sample the contour along its length). Boundaries averaged around five pixels difference.

II.2.3 INTELLIGENT SCISSORS

The *Intelligent Scissors* (IS) of Mortensen and Barrett [1995, 1998], also known in the literature as the Live-Wire tool, is a user-guided method of boundary definition. With an initial mouse click, the user places a starting point on a boundary of interest; the system then follows the edges in an image to define a path from that initial control point to the cursor's current screen location. As the cursor moves, this path is updated in real time and appears to be a wire snapping around on the edges in an image, hence the terminology "live wire" for this tool. If the user is satisfied with a segment of the boundary that is currently defined by the live wire, they click again on the boundary to lock the segment in place, and the live wire now uses that newly defined control point as the start for an edge-following path to the cursor. To complete a boundary, the user clicks on the starting point when the live wire comes back to it. This guarantees a

complete and closed boundary definition. Eric Mortensen [1999] noted this live-wire technique was incorporated by Adobe into Photoshop from version 5.5 forward as the "magnetic" selection tools.

In a preprocessing step, for every image pixel a local cost from that pixel to each of its eight neighbors is computed. This local cost is a weighted sum of several features (discussed in detail in Chapter IV), all scaled so that strong edges result in low values. The image is recast as a weighted graph, with the pixels as the nodes, and each pixel-node has weights on the eight graph arcs (edges) connecting that pixel to its eight adjacent neighbors. As the user then places control points on boundaries of interest, the system computes a minimal cost path from the most recent control point to every pixel in the image by computing an optimal spanning tree of the image using a graph searching algorithm. Thus as the cursor is moved from pixel to pixel in the image, the optimal path can be quickly determined and redrawn at interactive speeds from control point to cursor as it moves.

Due to its reliance on minimal cost paths, IS favors shorter paths over longer ones. When used to bound a structure that has a long protuberance with a thin neck, IS will favor a shortcut across the neck rather than following its full extent around the protuberance. This is discussed further in Section V.3.1. Another artifact occurs when a weak edge of interest lies near a strong edge. In this situation, the minimal cost path pays a small cost to cut over to the strong boundary which is overall cheaper for a long run of boundary pixels.

The human-machine interface engineered for IS is enviable. The overall approach is close in concept to ETA in that IS uses an incremental extension process, always working off a human supplied reference point.

II.2.4 MISCELLANEOUS INTERVENTIONS

Heinonen, et al., [1998] deal with isolating and calculating the volume of the lesions of a brain; thus it's a binary labeling task – lesion / non-lesion – for each pixel. They note that due to variable shape, size, and intensity of plaques and different locations near cerebrospinal fluid spaces, totally automated segmentation techniques are not able to detect all lesions. Their process follows these four steps:

(1) 3x3 low-pass filtering (an optional step, since it does impact the accuracy of the results);

(2) thresholding – three bitmaps for three intensity ranges, because lesions vary in intensity;

(3) manual editing – region growing to isolate the lesions, but manually placed lines (their example) are used to prevent regions from growing into immediately adjacent similar structures such as ventricles and sulci; and

(4) combine the three bitmaps and superimpose on the original image for context; lesion volumes are calculated by summing the regions identified over all the slices.

To judge their results, they tested with phantoms (i.e., known ground truth objects) and did inter-observer and intra-observer studies. The inter-observer study consisted of four experts and six patients, and the variability across experts was 7%. In the intra-observer study, using two experts and six random image sets each four times over two weeks, the variability was found to be 3-4%.

The time required to process images was 5-20 minutes per image set of 21 slices, when looking for plaque (the hard task). The easier task of looking for cerebral infarction took 2-10 minutes per set (with possibly more images).

II.3 Learned and Autonomous Systems

The concept of *learning* in a system has many different implementations. This section reviews a sample of recent research where learning is integrated as a component of the system.

As discussed in the opening of this chapter, systems which learn will start with the data and derive boundary definitions based on that observed data. One pure learning approach is to create definitions based on statistical properties of system inputs; such systems are discussed in Section II.3.1. Between the extremes of pure learning and pure *a priori* modelling, a parameterized model may be defined in advance, and the system "learns" by setting the parameters based on the data; Section II.3.2 reviews research with this approach. Training of fuzzy classifiers is also similar in this regard, and is reviewed in Section II.3.3.

II.3.1 Statistical Characterizations

Konishi and Yuille [2000] note that, *"Although there has been recent progress in general purpose image segmentation, it remains an extremely difficult problem."* Their goal is to learn segmentation cues within a particular domain, rather than to rely on *a priori* assumptions of color, textures, or other clues in a general case. Their strategy is to partition a labelled dataset, statistically characterize one part, and examine how well those statistics correctly label the pixels in the other part. In the dataset they use – a collection of images of English county roads – all pixels have been manually identified as one of six classes: edge, vegetation, air, road, building, or other.

Their pixel classifications are based only on local information. The basic set of filters they used are color intensity, gradient, Nitzberg corner edge detector (Nitzberg,

et al. [1993]), and LoG (Laplacian of Gaussian). The filters were used over both intensity and color values and at a variety of scales. They report that the most effective filters were color intensity and the Nitzberg operator, which was originally designed as a corner detector but is generally good at distinguishing regions of different textures. Gradient and LoG filterbanks were less effective, though it is noted that they may have been more effective if sufficient data were available to enable training them at a larger number of different scales.

Once filters were chosen, the response values were quantized into six possible response bins. These bins were selected by running the filter over the image, and evaluating a histogram of responses to the six different classes; the histograms were normalized to give the six conditional distributions P(ResponseBin | Class). Priors are estimated by the number of pixels in each class computed over the entire dataset, and a Bayesian classification rule developed based on the conditional and prior distributions. Six bins for each dimension of the filter using six coupled filters resulted in 6^6 (~46,000) quantized bins overall. Using more bins leads to overfitting problems and increased computational cost, while using fewer bins leads to cruder classifications.

Using the best selection of filters and bins for color and texture over several scales, they found that the probability of correctly classifying pixels as vegetation or air or road is around 90%, however the probability of correctly classifying edge pixels is 60%-70%, depending on the prior used. Besides the ill-defined 'other' class, the most frequent error occurred in the 'edge' class; the texture of vegetation produced a large number of small edges, which would then confuse the edge detector.

Kim, et al., [1999] present an approach to the problem of supervised texture segmentation using nonlinear support vector machines (SVMs). SVMs are well-defined

for two-class problems, however multi-class SVMs are still an active research area. They address this issue in the framework of an R-class texture discrimination problem. For each texture class, a nonlinear SVM is constructed which separates that class from others in aggregate. The segmentation is then achieved by using the outputs of the SVMs as inputs to an arbitration system which determines the correct classification from the pooled responses. They propose using SVMs in place of neural networks, because the SVMs are based on statistical learning theory, they have shown better generalization performance in some cases, and the number of hidden units and their weights are optimally and automatically determined. Interestingly, this arbitration system is a neural network, trained with standard error backpropagation, to determine the winning SVM for the R-class problem.

For a 16 class problem, their classification results worked well within the body of the texture. However, the majority of misclassifications systematically occurred at the edges. Edges are where any texture-recognition system, by its nature, will have problems, thus texture classifiers will likely not make good edge detectors.

II.3.2 Learned Model Parameters

The work of Fenster and Kender [2000a] is aimed at learning a model of shape likelihood, given an image. The shapes they use are piecewise cubic polynomials. The objective function sums the gradient strengths over the shape boundary. For their objective functions, they assume independent, identically-distributed values of intensity and directional gradient at all points along a shape S. Their definition of *learning* is to recover two Gaussian distributions for the intensity and gradient. The joint probability of these at every point around the contour is modeled by the probability of observing those features on S in image I; the negative log of this product of Gaussians is the image

energy. These models assume a uniformity around the shape boundary, which is often not true, leading to *sectored snakes*, where the complete S is divided into regions (sectors) and dealt with independently.

A goal of their work is to characterize the performance of objective functions by measuring their evaluation of near-correct shapes. This is done in two real domains, abdominal CT and echocardiogram (heart ultrasound) imagery. The ground truth shape should score better than any other shape, and as the shape approaches the truth its score should improve (for gradient descent to work). In studying perturbations to the ground truth of a shape, they find that traditional ACMs, attracted to the strongest edges, incorrectly gave 15% of the perturbed shapes a better score, and at higher blurs (coarser scales) the ACMs did worse.

The following points summarize their work. ACMs with learned parameters outperformed the unlearned "traditional ACMs" when homogeneously treated around the shape. Of the four statistical methods they used to train ACMs, the best method was dependent on the domain (CT vs. echocardiograms). In the echocardiograms, false positive rates were unacceptable for all functions tested (in other words, none of the functions worked well).

Durikovic, et al., [1998] note that, *"Although 3D reconstruction is widely used in CT and MR imaging, the methods do not fulfil all the needs in anatomy. Anatomists seek information about the exact overall shape and try to ascertain the features that build it."* These researchers are working to define highly accurate boundaries of anatomical structures, similar to the work at Visible Productions, but at a finer detail by one-to-two orders of magnitude; they are building embryo models based on slices 7-30 microns thick, while the Visible Human slices are 300-1000 microns thick. They face the same problems of registration, segmentation, and topological reconstruction.

ACM methods were troublesome in defining a highly concave structure or a structure whose topology changes (branching/merging) between sections. In their embryo reconstructions, ACMs were useful only in reconstructing higher contrast regions, such as the outer skin. They developed a user-guided approach and extending ACMs to track both boundary and topology across sections by (1) using an "area energy" rather than an energy along the line, (2) assuming structures were expected to maintain an average texture, (3) adding a contour splitting operation to deal with branching, and (4) setting initial parameters based on an initial estimation of the structure boundary. With this approach, they found that 94% of the contours were correctly connected, and construction of three-dimensional models was roughly quantified as being reduced from a few months to only several days.

Working with neural networks, Brahmi, et al., [2000] developed an operational system devoted to the segmentation of virus-infected areas in images of the retina. The images were divided into 16x16 windows, and the task is framed as a window-labeling task – classifying windows as showing infected areas versus healthy areas. A 128x128 pixel window was divided into eight rows and columns to produce the 16x16 pixel sub-windows. They used principal component analysis to find the first 20 eigenwindows of the sub-images, and then used the 20 projections onto these eigenwindows as inputs to a neural network classifier. In addition to reducing the size of the input space, this procedure also decorrelates the inputs. They trained the neural net to discriminate between healthy and infected retina. On separate retina images (not in the training set), they correctly classified 84% of the windows.

II.3.3 FUZZY CLASSIFIERS

Schalkoff, et al., [1999] describe a general image analysis and segmentation method using fuzzy set classification and learning. Their application is part of an autonomous robot designed to inspect U.S. Department of Energy warehouse waste storage drums for rust. Drum surface images are acquired under controlled conditions and subsequent visual inspection classifies the drum as "acceptable'" or "suspect'". The method uses a learned fuzzy representation of pixel region characteristics, based upon the conjunction and disjunction of extracted and derived fuzzy color and texture features. The problem is setup as a two class recognition problem, where the two classes are (1) acceptable image regions, and (2) suspect or flawed surface characteristics. Color images are represented in six dimensions for each pixel, using the three standard HSI (hue-saturation-intensity) planes and three derived texture planes in each of those HSI dimensions. The texture measure is the standard deviation over a 3x3 neighborhood.

The learning part of this system follows these steps. First, the trainer selects positive and negative exemplars, then training data is calculated for each pixel based on a local 3x3 neighborhood. Positive and negative data sets are then clustered by k-means (Duda and Hart [1973]), with a user-defined number of clusters. Finally, each dimension is used to generate membership functions for positive and negative sets.

Once the fuzzy membership functions are learned, the classification is done by: (1) using the six dimensions, AND-ing the fuzzy dimensions for positive and negative membership; (2) OR-ing the data together for each cluster; and (3) classifying the image as positive if the fuzzy positive measure exceeds the fuzzy negative measure. Ninety-five percent correct classifications were cited.

One interesting note from their case study: four images with flawed areas were used to generate positive membership functions, then false-positive classifications using that positive membership measure were used to generate the negative membership function. This is an efficient way to develop the negative exemplars, since the false positives are the cases which need correction, and they are likely to provide the strongest distinctions between the “acceptable'” or “suspect'” cases.

Karayiannis and Pai [1999] defined a methodology for learned segmentation using Fuzzy Algorithms for Learning Vector Quantization (FALVQ). LVQ, generally, consists of grouping feature vectors into clusters and representing each cluster by a prototype; a feature is then determined to be of the type of its closest prototype. They distinguish between crisp algorithms, which assign each feature vector to a single cluster, and fuzzy algorithms, where a membership function is used to assign feature vectors, by degree, usually to multiple clusters. Crisp LVQ, characterized by the methodology of Kohonen [1997], allows the update of only the “winning” (i.e. closest) prototype in the competitive network. Soft LVQ algorithms, in contrast, allow all the prototypes to be updated for a given input to the network. The authors develop a family of FALVQ algorithms, whose differences are basically in how the competition between prototypes is regulated during learning.

They test their methodology on brain magnetic resonance (MR) imagery, trying to identify a tumor in the scan across the three standard MR channels T1, T2, and SD. After the patient took Gadolinium, the tumor was easily seen in its entirety in a T1 scan, and this is the ground truth for the task. The Kohonen LVQ methodology did not segment the tumor well; however, the FALVQ family has several tunable parameters, and at appropriate settings, they segmented the tumor very closely to the truth.

Only four images were used in this study – this paper demonstrates the potential of a technique rather than presenting a detailed analysis. There are no comparisons to other methods besides Kohonen. The authors write that the "*use of unsupervised LVQ algorithms does not rely on a priori information provided by human experts*." However, they spent considerable effort in tuning the parameters across a family of algorithms to find settings which gave the result desired. It is not obvious that this will generalize to finding tumors in other images, or to identifying other cerebral abnormalities.

II.4 Learned and User-Guided Systems

In later variants of the Live Wire / Intelligent Scissors techniques, Mortensen and Barrett [1998] used a basic statistical characterization of recent boundary history to "train" some of the edge strength functions used in the overall cost function. For example, they tracked a profile of pixel values on the boundary of interest, and then assigned a low cost for boundary segments consistent with the profile and a high cost otherwise. They demonstrated this on one example, preferentially following a weak rather than strong edge, however, they provided no analysis on the overall effectiveness of such training.

Falcao, et al. [1998], working independently of Mortensen and Barrett but with the same basic Live Wire idea, extended the technique in two further ways; first, in a more sophisticated learned cost assignment, and second, using adaptive neighborhood sizes dependent on tracing speed. For the cost assignment, a simple neighborhood of six pixels around the boundary arc of interest is used. The pixel values are weighted in several combinations to form seven features, and these features are converted to cost functions through six parameterized transformations. The transformations are either linear, Gaussian, or modified hyperbolic in form. The transforms are defined such that

features indicating a boundary of interest result in low values, consistent with defining boundaries as minimum cost paths along the graph's arcs.

The "training" of their system consists of the automatic selection of features, transforms, and parameters for the cost function. To train their system, the user "paints" a series of typical boundary segments. Basic statistics of the painted pixels (minimum, maximum, mean, and standard deviation) are then used to define parameters of the transforms. Any of six transforms can be applied to each of the seven features, and they evaluate all possible feature subsets to find the closest match of their overall evaluation function to the example segments supplied for learning. The results of this system compare well to the richer set of hand-crafted features used in Intelligent Scissors. This performance equivalence shows the strength of this method which starts with only six pieces of pixel data but searches the power set of {{features} X {transformations}} to find the best evaluative subset for the task.

The second concept Falcao, et al., introduce they call *Live Lane*. They define a *lane* around the boundary being defined, and the lane limits the extent of the graph search in seeking the minimal cost boundary path. The width of the lane can be controlled dynamically by the user. For example, the user may move the cursor quickly across an obvious and well-defined boundary but may slow down their cursor motion for ill-defined or atypical boundary segments, and the lane width can be adjusted automatically based on the speed of the cursor motion. In the extreme case where the user slows down dramatically for precise boundary placement, the lane width reduces to zero and the system is effectively in a manual-tracing mode. In evaluating their method across several users, the preferred mode of user interaction with this dynamic lane width was a user-initiated change, by key-stroke or mouse-click, that alternated the

mode between some wide width for straightforward boundaries and a narrow width for user-guided precision.

As Falcao, et al., improved Live Wire through additional learning, Cootes, et al. [1994], add a learned component to ACM methodologies. They define a statistically based technique for building compact models of shape and appearance, called Active Shape Models. Similarly to ACMs, the shape models are initialized near a structure of interest and they iteratively converge to the structural boundary, however the shape model is topologically constrained by prior exemplars. The shape model is derived from a large set of boundaries; as an example they use the left ventricle in 66 heart echocardiograms. The mean shape is calculated, and from the covariance matrix of residuals off that mean, the leading eigenvectors are used to constrain the variation in shape allowed during the iterative convergence. Work by Cootes and Taylor [2001] develops Active Appearance Models along similar lines, with boundary points being selected based on local textures rather than edge strengths.

This method is usually user assisted in two aspects First, an expert is required to bound the structure on all exemplars. Second, the user is often involved in the initial placement of the shape model or identification of key landmarks used by the model. The authors propose using a genetic algorithm for finding a suitable initial shape, but in general this is a hard problem, and one for which humans are appropriately adept, for instance by roughly setting a bounding box around the structure of interest. This methodology works well given a consistent topology across exemplars, as for example with sets of hearts or sets of faces. However, this methodology fails on the sulci of brain or the branching of minor arteries and veins, all of which have a topology that differs from case to case.

The Expert's Tracing Assistant of this disssertation has both strong interactive and learned components, as in the other systems discussed in this section. Historically, the

initial ideas presented by Crawford-Hines and Anderson [1994] framed boundary definition as the result of learned segmentation. A neural network was trained to distinguish structure from non-structure, and standard flood-fill algorithms were used to create the segmentation using this learned distinction of structure. The basic boundary tracing framework and initial experimentation with input and output representations were published by Crawford-Hines and Anderson [1977a, 1977b]; that work is thoroughly detailed in Chapter III. The full Expert's Tracing Assistant system was detailed by Crawford-Hines [2000] and was placed in the overall context of a three-dimensional model generation system by Crawford-Hines and McCracken [2002].

——== Chapter III ==——

Building a Viable Expert's Tracing Assistant (ETA)

The Expert's Tracing Assistant (ETA) is a software system developed to validate the ideas set forth in this dissertation. The overall framework of the ETA is presented in Section III.1. Successive sections discuss experiments that investigate the capabilities and limitations of this framework and the sensitivity to design choices and parameters. Section III.2 presents experiments with the viability of different system output representations. Section III.3 addresses the issue of balance in the creation of a training set. Section III.4 demonstrates the impact of richer input representations on learning speed. Experiments in Section III.5 show the path to defining a flexible set of input primitives for the robust learning of boundary definitions.

III.1 ETA Framework

Presented in this dissertation is a general and flexible formulation for following boundaries in images. The general process follows these steps:

A) sample pixel values through the neighborhood of representative segments of a user-defined boundary;

B) build a training set, based on the methodology to be followed in step D;

C) learn dynamically an evaluation function to distinguish *on–boundary* versus *off–boundary* points;

D) as new boundary segments are started, evaluate potential next pixels adding the most likely next pixel to the boundary, and iterate;

E) and maintain user control of the process, so that the expert can easily override the system when needed to refine its choices or correct its errors.

The system works from whatever boundary piece has been established and extends it forward. This behavior parallels the "road follower" systems outlined in Chapter II. Those systems, however, were hardwired with *a priori* knowledge from their application domain. For a more flexible solution, the structural boundary representation is learned as needed in ETA. The goal is to develop a flexible system architecture and learning method so users can begin a tracing task, then have the system learn it and take over the repetitive parts of that task.

Neural networks are used to learn the boundary character. The inputs to the network are features of a local neighborhood around the boundary. Standard error backpropagation algorithms are used with a three-layer network (one input, one hidden, and one output layer). The output layer is interpreted as an indicator of whether a candidate pixel is a boundary pixel or not; alternative output representations are explored in Section II.1.2.

III.1.1 Creating Exemplars by Sampling A Neighborhood

Figure III–1 illustrates the options for continuing a boundary trace forward. The figure shows detail along a grey-white boundary, where each square in the picture represents a pixel. A partial boundary is defined, in a direction from left to right, by the pixels

marked by X. Under basic assumptions of an 8-connected boundary that doesn't backtrack on itself, the possible candidates for the next pixel on the boundary are labelled 1 through 5.

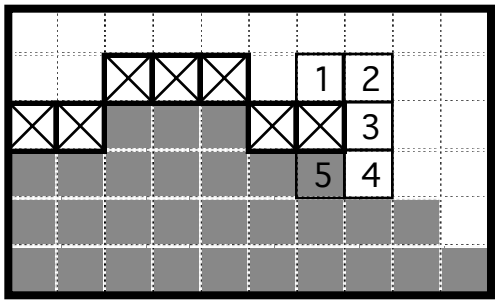


Figure III–1: A partially traced boundary is marked by X's, and options for the next pixel on the boundary are numbered 1 through 5.

Clearly in this simple case candidate 4 is the correct choice for the next pixel along the boundary, while 1, 2, 3, and 5 are wrong choices. Note that the correct next-pixel choice is not a function of the pixel itself, since pixels 1 through 4 are all white, yet only pixel 4 is correct. A decision rule for picking boundary pixels could be stated as this, *"Boundary pixels are those which are white and whose immediate neighbor is grey."* The key thing to note is that boundary is not a function simply of a pixel itself, but of its local neighborhood.

A *boundary* separates things. To accurately locate a boundary, it is important to monitor the things that are being separated. For example, to demarcate the boundary of a river' flood plain just after a flood, you could find the edge of the flood's silt along the drainage, then walk the boundary by keeping all the silt-covered landscape to one side of you and the clean higher ground to the other side. What's ahead or behind you doesn't so much matter, but what does matter is what's off to your immediate left or right as you walk, i.e., what's perpendicular to your direction of travel.

Figure III-2 sketches the situation. A direction for the boundary is arbitrarily chosen, indicated by the arrowheads. This direction serves to orient the perpendicular neighbors so they can be labelled consistently as *left* or *right*. At several spots along the boundary a point has been circled and labelled C. The two neighbors perpendicular to the

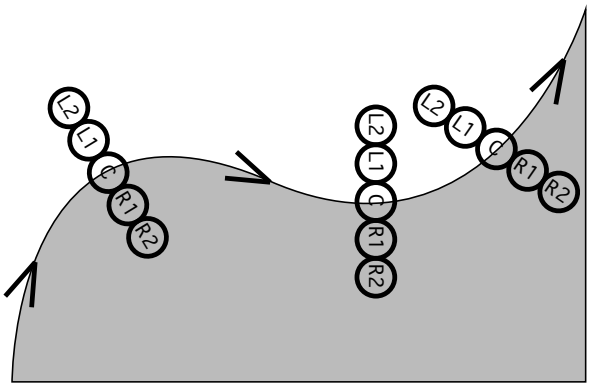


Figure III–2: Three spots (labelled by 'C') on a directed boundary; left and right neighbors to C are defined perpendicular to the boundary's tangent at C and with respect to the direction indicated by the arrows.

boundary at that point, both to the immediate left and right, are labelled L1, L2, R1, and R2, respectively. These neighbors will be used as reference points in the ETA framework.

Using a boundary direction to distinguish left from right, the decision rule for Figure III-1 might now be re-worded as, *"For the next pixel on the path, choose the white pixel whose immediate right neighbor is grey"*. To quantify this rule, data is needed for each candidate pixel and its immediate neighbors.

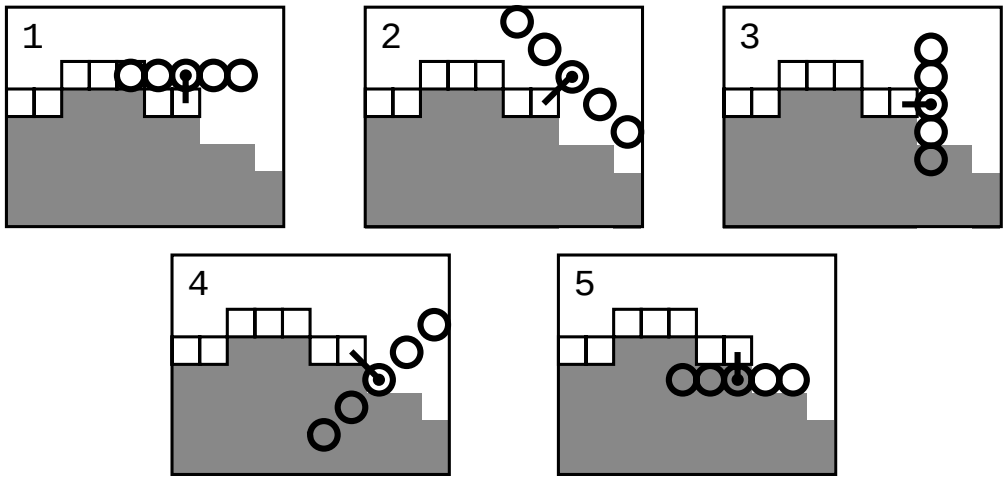


Figure III–3: Five neighbor sets for consideration, based on the direction to the proposed candidate pixel; the five diagrams correspond to the cases defined in Figure III–1.

Figure III-3 shows five diagrams, one corresponding to each of the candidates identified in Figure III-1. In each diagram, the five circles show the candidate point with its two left neighbors and two right neighbors, following the layout of Figure III-2. The

orientation direction is established from the previous point to the candidate point, indicated by the short line segment.

As a basic case, consider using the pixel values at these locations to define a training vector for each of these candidates. Representing white with a value of 1.0 and grey with a value of 0.5, these five candidates yield the five training vectors shown in Table III-1. The values of *true* and *false* in the table are taken from Figure III–1, where it was observed that case 4 represented the desired boundary pixel and the other cases did not.

Table III-1: The five cases of Figure III-3 produce this training set of exemplars.

case	— *data* —					— *response* —
	L2	**L1**	**C**	**R1**	**R2**	**on the boundary ???**
1	1.0	1.0	1.0	1.0	1.0	false
2	1.0	1.0	1.0	1.0	1.0	false
3	1.0	1.0	1.0	1.0	0.5	false
4	1.0	1.0	1.0	0.5	0.5	TRUE
5	1.0	1.0	0.5	0.5	0.5	false

The learning procedure takes data such as the five vectors listed here, and learns the response from the data. For the simple decision rule stated earlier, *"Choose the white pixel whose immediate right neighbor is grey"*, a quantitative rule could be crafted as:

$$PixelValue(\mathbf{C}) + PixelValue(\mathbf{R1}) = 1.5 \quad ==> \quad \mathbf{C} \text{ is a boundary pixel}$$

This decision rule would select the white edge pixels when tracing left-to-right in Figure III–1, however it would select the grey pixels when tracing right-to-left.

Section V.3.3 shows this behavior occurring in practice, when the boundary tracer gets turned around by mistake and continues on, at some offset, in the opposite direction.

III.1.2 Neural Network Architecture for Boundary Learning

The choice of a particular learning methodology is not the focus of this research work, though certainly for any system based on this work to be successful, it must learn quickly and efficiently to be able to aid the experts in their tracing tasks. Given n–dimensional vectors of data (the training data) and responses (positive or negative), there are many avenues to learn appropriate responses given the data, such as neural networks, decision trees, support vector machines, and Bayesian networks. The choice was made to use neural networks because they are well understood and straightforward to implement, their feed-forward calculations are quick, they can learn non-linear mappings of data to response, and they are flexible in their configurations of layers and nodes. This section presents the standard exposition of an error-backpropagation neural network (Rumelhart, et al. [1995]).

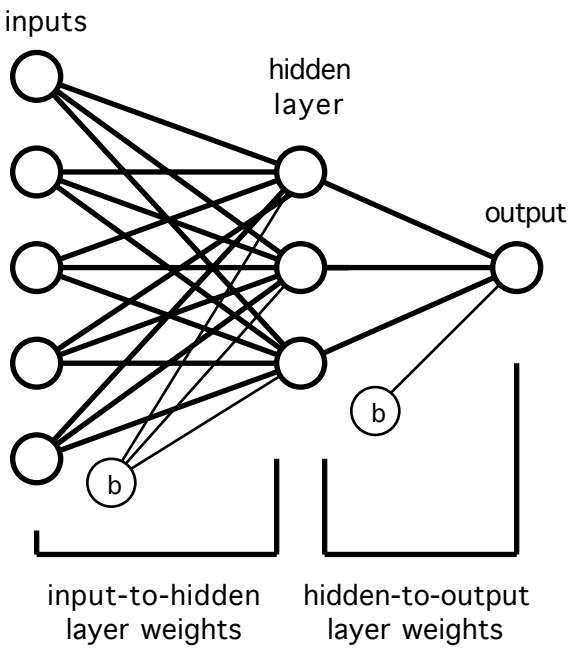


Figure III–4: The 5-3-1 neural network architecture has 5 input units, 3 hidden units, and one output unit; the bias units are shown labelled with a 'b'.

Figure III–4 illustrates the architecture of a basic three-layer neural network with several inputs, subscripted by i and denoted here as x_i, one hidden layer of several hidden units, subscripted by j with outputs denoted here as h_j, and one output

unit with output y. The bias units, labelled 'b', feed in a fixed signal of one to the hidden and output layers. Essentially, the bias unit at the input layer acts as input x_0, and subscript i for **n** inputs run from zero to **n**; the hidden layer bias unit is treated analogously.

In the general case, there may be several output units for a neural net, but in ETA only one output unit is used, so this exposition limits itself to this case. The notation *"5-3-1 network"* will be used as shorthand for this configuration of five inputs, three hidden units, and one output.

Doubly-indexed weights w_{ij} denote weights from input unit i to hidden unit j. Singly-indexed weights w_j denote weights from hidden unit j to the output unit. The weights are initialized to small random values, and the network "learns" the mapping of inputs to outputs by adjusting these weights based on gradient descent of the mean-squared error function of the network output over its training data with respect to the weights.

For each hidden unit, the inputs are multiplied by the corresponding weights, summed, and that result is passed through a sigmoid "squashing" function $f(x) = (1 + e^{-x})^{-1}$ which remaps the sum to the interval [0,1]. A step size d is calculated in the direction of the gradient and weights are adjusted by some fraction α of this value. This value is the *learning rate* of the network, and α^y is used to adjust the weights w_j and α^h is used to adjust the weights w_{ij}. Momentum μ is used to potentially speed up the gradient descent by adding some fraction of the most recent weight change to the current change. For a vector of input values x_i and a desired correct value c, the weights evolve through iterations governed by these equations, computed by this sequence of equations, left-to-right, top-to-bottom:

$$h_j = f(\sum_i w_{ij} x_i) \qquad y = f(\sum_j w_j h_j)$$

$$d^y = (c-y)y(1-y) \qquad d_j^h = (w_j d^y)h_j(1-h_j)$$

$$w_j^* = w_j + \alpha^y d^y y_j + \mu\Delta w_j \qquad \Delta w_j = w_j^* - w_j$$

$$w_{ij}^* = w_{ij} + \alpha^h d_j^h x_i + \mu\Delta w_{ij} \qquad \Delta w_{ij} = w_{ij}^* - w_{ij}$$

The first pair of equations represent value propagation from the inputs to the hidden units to the output. The second pair of equations represent the error gradient information at the output unit and the weight-proportional propagation of this error information from the output to the hidden units. The third equation pair shows the updating of the hidden unit weights, where w^* represents the new value, and the fourth shows the updating of the input weights.

III.1.3 Single Channel Inputs

Only a single eight-bit value was used for the raw data associated with each pixel. This is a common denominator across many imagery formats and provides a computationally compact set of inputs. Inputs are normalized to the range [0,1].

Some imagery data comes in at more than eight bits, for example the DICOM format for CT imagery has 12 bits per pixel. For this imagery to be displayed on a conventional RGB monitor, the data can be pseudo-colored across the three eight-bit color channels, or a sub-range of values can be re-mapped to eight-bits and then displayed in greyscale. In practice, the user adjusts a visual display to highlight the portion of the dynamic range that most accentuates the anatomical structure of interest, these values are mapped to the interval [0..255], and ETA uses these re-mapped values as it would any single 8-bit channel.

For RGB imagery, using three values per pixel would triple the number of inputs, triple the associated number of weights from input to hidden layers, triple the number of free parameters thus increasing the needed number of training exemplars, all of which imply increased computation for training. Also since the three color channels are highly correlated, progressively less information is gained, at a higher computational cost, by adding a second or third channel of data. For the purposes of this dissertation, it was decided to use only one channel of data. For general color imagery, the RGB channels are averaged to one greyscale channel. In the Visible Human imagery, the green channel provides the best contrast and distinction among structures overall as shown in Figure III–5, thus the green channel was selected as the single channel of input.

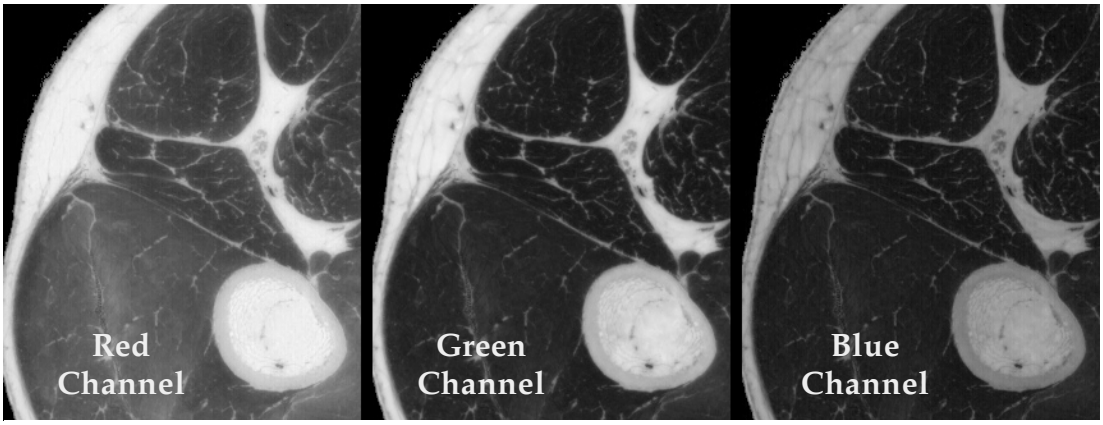


Figure III–5: The red, green, and blue channels for a detail from the Visible Human imagery are highly correlated, and the green channel provides somewhat better contrast and distinction among the structures.

III.2 Output Representations

The neural network produces a floating point number on [0,1] as an output, and this number needs to be related to the application. In the ETA framework, the output needs to be used to make the distinction between vectors that represent *on-boundary* points and those that represent *off-boundary* points. This section presents the results of experimentation with continuous-valued and discrete output representations.

III.2.1 CONTINUOUS-VALUED (SEF) OUTPUTS

One representation for the output unit is to use the continuously varying output over the range [0,1] as a measure of deviation to the left or right of the desired boundary for a candidate pixel. Example values for such an output unit are tabulated in Table III-2.

The output values for a candidate and its neighbors are interpreted as a low evaluation indicating the candidate pixel is off to the left of the boundary, a high evaluation indicating off to the right, and a value near 0.5 indicating the candidate pixel is on the desired boundary. This type of response is suggested from a control-systems perspective, where an action taken in response to being far-left of target is opposite to the action taken when far-right of target, and the degree of response is related to the degree of deviation.

Table III-2: A restatement of Table III-1, adding quantified responses for SEF outputs.

case	— *data* —					— *response* —	
	L2	**L1**	**C**	**R1**	**R2**	**boundary?**	**SEF**
1	1.0	1.0	1.0	1.0	1.0	far left	0.1
2	1.0	1.0	1.0	1.0	1.0	far left	0.1
3	1.0	1.0	1.0	1.0	0.5	near left	0.3
4	1.0	1.0	1.0	0.5	0.5	YES	0.5
5	1.0	1.0	0.5	0.5	0.5	near right	0.7

The network learns an evaluation function that produces smoothly changing values as a pixel and its neighbors change from left-of-boundary values, to on-boundary values, and then to right-of-boundary values. This output representation is named the Smooth

Evaluation Function (SEF), since as candidate pixels are considered from left to right with respect to the established boundary orientation, the single output unit should vary smoothly from low to high.

In an initial demonstration of ETA, the use of SEF output units proved promising. Figure III-6 shows an MRI image from the work of Hyde, et al., [1995] at the National Institute of Mental Health. To define the boundary between grey and white matter in this image, a short training segment of twenty pixels is identified by the star. A 5–1–1 neural network using inputs of [0,1] normalized greyscale values and an SEF output unit was trained to learn this grey/white distinction. The boundaries shown were then traced automatically by the ETA prototype. Loops in the boundary followed by gaps indicate where the system lost track of the boundary and it was manually restarted a few pixels later. Visually, it can be observed that the vast majority of the grey/white matter boundary could be well-traced automatically with this configuration.

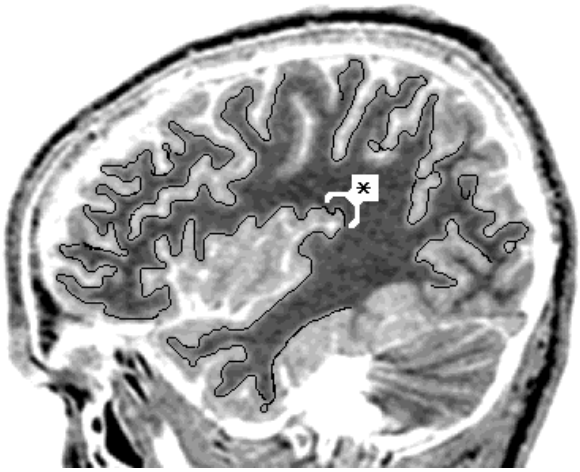


Figure III–6: The segment indicated by the * was used as training data to learn the grey/white matter boundary in this MRI image; the other boundaries shown were generated automatically.

III.2.2 Feature Detector (FD) Outputs

Another representation for the output unit is as a **feature detector**: the output goes high in the presence of a feature and goes low otherwise. In this case the feature is whether or not the candidate choice represents a boundary pixel.

For a trained neural network with a single output unit and sigmoid functions on the hidden and output layers, Rumelhart [1995] showed that an output unit with this interpretation can be interpreted as generating a probability – the probability of the feature given the inputs. In light of this probabilistic interpretation, the network's output can be thought of as a relative certainty of a pixel being a boundary pixel, and these boundary tracing actions are defined given a particular range of the network's output:

Output Value	**Interpretation**	**Action Implied for Candidate Pixel**
[.75, 1.0]	Likely to be a boundary point.	Add it to the boundary and continue.
[.25, .75]	Uncertain.	Pause and ask for user guidance; use this case for additional training
[0, .25]	Reasonably certain it's not a boundary.	Don't use it.

As a practical matter, targets of 0.1 and 0.9 are used instead of 0.0 and 1.0 to avoid saturating the units during training. Table III-3 shows the quantified target values for training of FD output units.

Table III-3: A restatement of Table III-1, adding quantified responses for SEF and FD outputs.

case	*data*					*response*		
	L2	**L1**	**C**	**R1**	**R2**	**boundary?**	**SEF**	**FD**
1	1.0	1.0	1.0	1.0	1.0	far left	0.1	0.1
2	1.0	1.0	1.0	1.0	1.0	far left	0.1	0.1
3	1.0	1.0	1.0	1.0	0.5	near left	0.3	0.1
4	1.0	1.0	1.0	0.5	0.5	YES	0.5	0.9
5	1.0	1.0	0.5	0.5	0.5	near right	0.7	0.1

III.2.3 SEF VERSUS FD COMPARISON

To evaluate the relative merits of these two possible output representations, a study was done to compare the representations. The blob and the single-pixel-wide line in Figure III-7 were used to explore the attributes of these two different output units in a controlled image environment. The two tasks studied were how well the system could be trained to follow the blob's edge and to follow the line The training segment for the edge-following task, in black, is identified at the bottom of the blob, bordering the yellow shape immediately to its outside. The sharp aliased edge of the blob and the boundary defined just to its outside make this case just like the detail shown in Figure III–1.

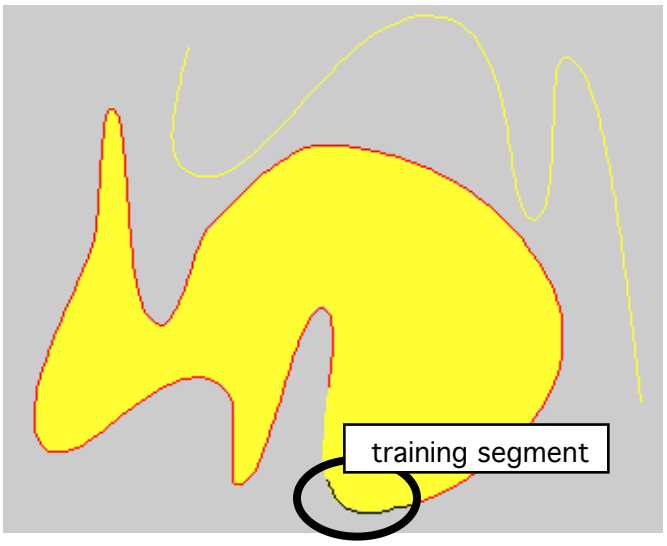


Figure III–7: Synthetic image for edge and line test cases, to compare SEF and FD output units.

The system, configured with either SEF and FD outputs, learned quickly to follow the edge of the blob. The red boundary line shown was traced out automatically, tracking the blob boundary cleanly, except for missing a few pixels in the areas of high curvature. Both SEF and FD systems performed comparably on this edge-following task; any slight difference between them was insignificant compared to the dramatic difference on the line-following task. When the system configured with an SEF output was put to the task of learning to follow the single-pixel-wide line, it failed miserably.

For the system configured with an SEF output, Figure III–8 shows the error for the two tasks plotted against the number of training epochs. The error measured is the difference between the neural net's output and its target; the root mean squared error is measured across the whole training set. For the edge following task, the steep descent of its error curve shows the task was learned quickly and well. For the line following task, however, there's little change in the overall error. While this result is for but one initialization of the neural network's underlying parameters, no matter how the training parameters were set, the system did not learn the line-following task. The problem is not in the parameters or training regime. There are serious flaws, both theoretical and practical, with an SEF output in the line following task.

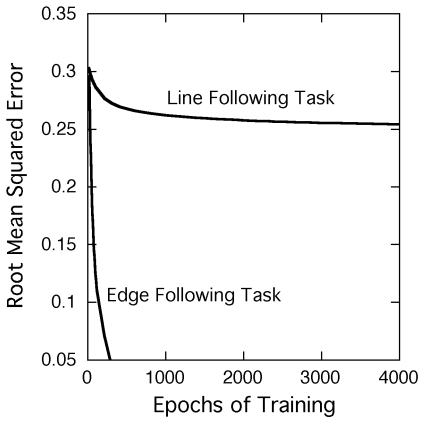


Figure III–8: Error curves for the SEF output configuration, for the line and edge tasks; the line task was not learned.

Consider a white line on a black background. The line's neighbors off to the far left have low values as do the neighbors off to the far right. So the five inputs for the far-off-to-left-candidate, all black, are driving the network toward low target values and the five inputs for the far-off-to-right candidate are driving the network toward high target values. But far-left-candidate's inputs and far-right-candidate's inputs are the same, because the background of the line is the same on each side. Since a neural network learns a functional form and this situation is now no longer a function, this is essentially un-learnable and the neural network weights do not converge onto any useful solution.

In addition to this fundamental limitation on SEF outputs, there is an additional practical problem. On the tasks where it is successful, such as the edge following tasks, the practical problem with SEF units is the interpretation of an output value of 0.5 as

being *on–boundary*. In following a boundary, it is useful to have some degree of confidence that a pixel classified as on–boundary really is a boundary point. But given a neural network with a [0,1] output unit and with randomized weights, any input vector to the network will produce an output of approximately 0.5, just through the statistics of randomization. Thus a totally random network, for example a just-initialized network, will always report that any input configuration presented to it represents an on–boundary point.

When the system was configured with FD output units, it proved able to learn both the edge-following and the line-following tasks (this can be observed in Figure III–14). Additionally, an FD output can be partitioned into three ranges, representing *correct*, *incorrect*, and *uncertain* responses, which are useful both in characterizing how well a network has learned its task, and in providing a trigger for an automated system to stop and ask for user guidance in areas of uncertainty. For these reasons, FD will be used in the remainder of this thesis.

III.3 Proportion of Positive and Negative Exemplars

A common problem when learning rules from data is an uneven representation of exemplars across different responses in the dataset. For example, LeCun [1997] noted, when training a hand-written digit recognition system, that it was crucial to balance the training set with approximately the same number of cases across the different digits.

As a general rule, there are typically far more ways to do something wrong than there are ways to do it right. As was shown earlier in Figure III-3, when generating exemplars for training the network, there can be four times as many negative cases as positive ones. When the number of negative exemplars is much larger than the number of positive ones, it is possible that the network will learn well the negative cases and have difficulty

learning the positive cases. The following study explores this issue in the context of ETA and illustrates problems which can arise when the numbers of positive and negative exemplars are out of balance.

Section III.3.1 illustrates a "normal" case, showing a reasonable evolution of the FD response as training progresses. Section III.3.2 presents the problematic situation which can be resolved by balancing the training set, as shown in Section III.3.3.

III.3.1 Histogram Distributions of Successful Learning

Figure III–9 shows four paired histograms which typify the evolution of the FD output unit's response over the course of a neural network's training. The FD output unit response is in the range [0,1], and its value is distributed among the eleven bins of each histogram. The heights of each set of bars have been scaled so that the largest response bin in each histogram will be the same, full height. For these examples, the important aspect to note is the overall shape and horizontal placement of the bars, not the absolute number represented by each bar's height.

In each pair of histograms, the top histogram, labelled **HIGH**, bins the responses of the network to the positive training vectors, which are trained to the value 0.9. The bottom histogram, labelled **LOW**, bins the responses to the negative training vectors, which are trained to the value 0.1. The first pair of histograms (in the upper-left) shows the outputs for the network in its untrained state. When all the weights are randomly set, the network's output for any given input is approximately 0.5. Thus all the responses for both positive and negative cases are in the central [.45 , 0.55) bin.

The following three histogram pairs (upper-right, then lower-left, then lower-right) show the network responses over the course of training. The overall responses of the HIGH

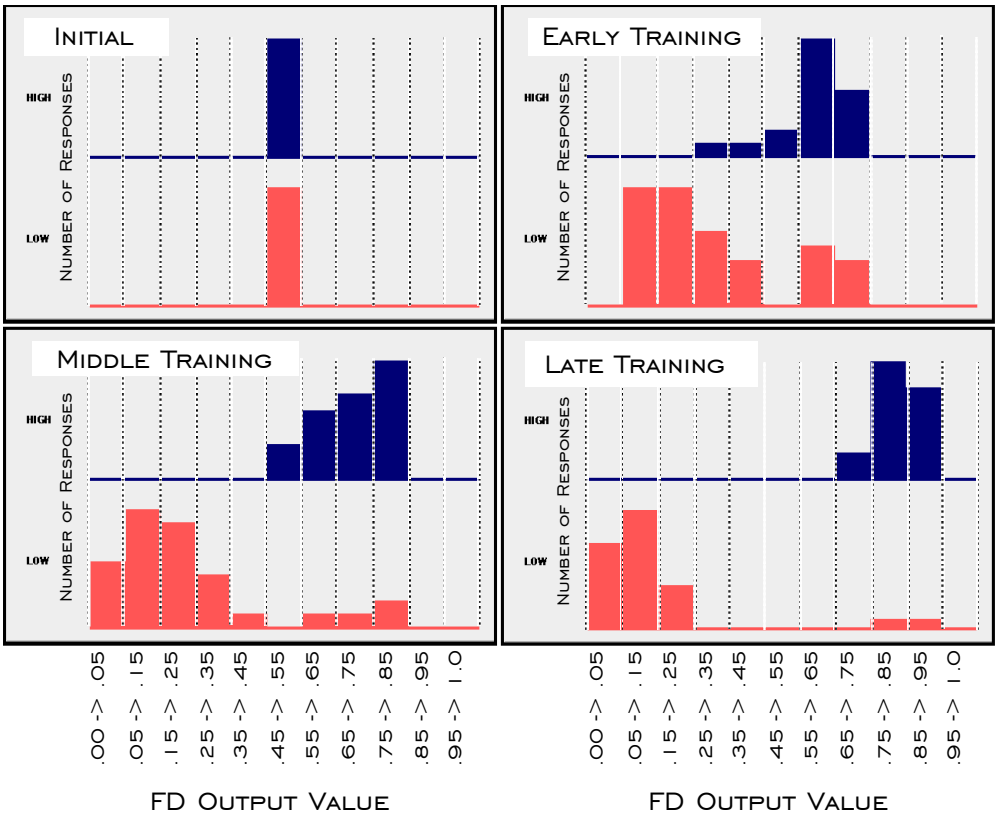


Figure III–9: Histogrammed responses of the FD units as training progresses: upper-left shows all units all around 0.5 in initial state; as training progresses from early to middle to late phases, the output units cluster toward their goals.

units are seen to spread out and move to the right; similarly the responses of the LOW units spread and move to the left. In the final paired histogram, the responses indicate the network has learned its task reasonably well: the HIGH units are mostly responding in the range [.75 , 1.0] and the LOW units mostly in the range [0.0 , 0.25). The few exceptions on the right-side of the LOW histogram are fairly common. Since there is often some variation in the trace used as the training exemplar, some of the data will inevitably be misleading, and one goal of this statistical learning procedure is to find the dominant trend in spite of such misleading and possibly contradictory data.

Also note in the final histogram pair the absence of any responses in the [0.35 , 0.65] range. This separation in response values between positive and negative exemplars is

one indicator of a well-learned case. Using a probabilistic interpretation of a network's output, new data that produces a value falling in the [0.35 , 0.65] range is an indicator of an uncertain response. Such a response can be used to stop the systems's automatic tracing and wait for user guidance. Additionally, as the user then provides a manual trace through this uncertain area, these manually traced boundary points can be used to generate further training data, which can then extend the range of the system's valid responses.

III.3.2 Abnormal Learning with Negative Over-Representation

The nominal training pattern shown in Figure III–9 may be disrupted by disproportionate representation in the training set. The upper half of Figure III–10 shows the data for this study. Magnified here to show the pixels, a dark line on a light background is corrupted by Gaussian noise. The exemplar used as the basis for a training set is shown in the lower half of Figure III-10, a tracing which basically follows the upper edge of the dark line. The trace varies slightly, sometimes with excursions into the light background. The corrupting noise and varying trace serve to generate a wider variation in the training set.

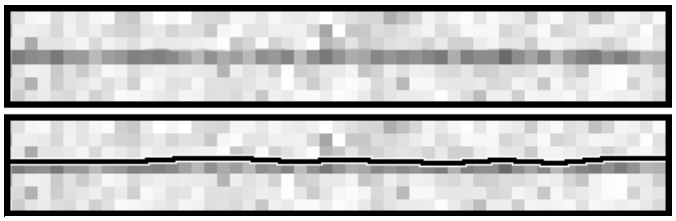


Figure III–10: An enlarged view of the pixels used when learning a noisy line: the raw image data *(above)* and the boundary trace, black outlined by white, superimposed on the image data *(below)*.

The 45-pixel-long trace is sampled at 22 points, generating 110 training samples in total (each point generates 4 negative exemplars and 1 positive one). Approximately 75% of the samples are selected for the training set; these exemplars are used to learn the boundary definition. The remaining samples are the validation set, which is used for determining when to stop the iterative

learning process. The allocation of samples to validation set and training set is purely a function of the initial random seed; other random seeds result in similar, though not exactly the same, behavior as the histograms shown in Figures III–11 and III–12. The neural network configuration was 5-1-1, thus a total of eight weights were learned (six plus the two bias weights).

Figure III-11 shows the evolution of learning after 50, 120, 500, and 1,000 epochs of training when the negative exemplars outweigh positive by 4-to-1. Thirty samples are in the validation set (used to stop the training) and 80 samples are in the training set; of the 80, 16 are positive and 64 are negative cases. In the first histogram pair of this set, the whole mass of the histogram has been pulled to the left due to the dominant presence of negative examples. In the second histogram pair, some of the positive exemplars have started moving to the right, a trend that continues as training goes on. In the third and fourth histogram pairs, at epochs 500 and 1,000, of the 16 positive training examples, the left-most 7 will be false negatives, the right-most 6 will be true positives, and the middle 3 will evaluate in the "uncertain" range. Less than half the training set positives are learned correctly. Virtually all the negative exemplars evaluate as true negatives, with a few evaluating as uncertain.

III.3.3 Balanced Exemplar Learning

There are several approaches to mitigating this imbalance. In this application, there is a surfeit of training data, and a simple and effective way to bring into balance the number of positives and negatives is by random sampling and using only one-quarter of the negative cases. Figure III-12 shows the evolution of learning when approximately 25% of the negative exemplars are chosen to be in the training set resulting in an approximately 1-to-1 ratio of positive to negative exemplars. In this case, there are now 45 samples

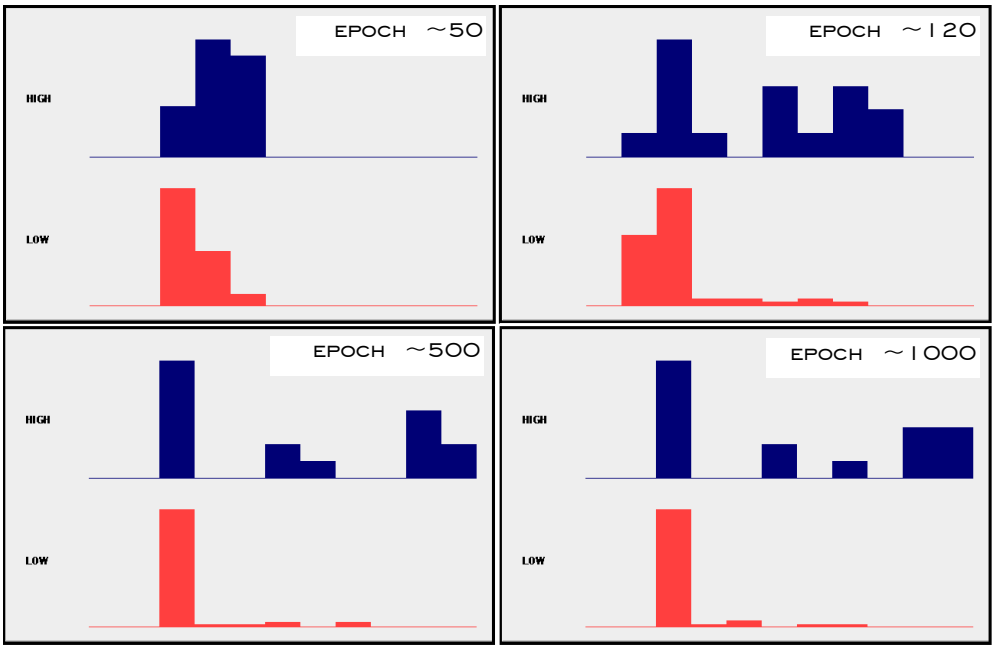


Figure III–11: A histogram of FD outputs after 50, 120, 500, and 1,000 epochs of training with 4:1 ratio of negative to positive exemplars.

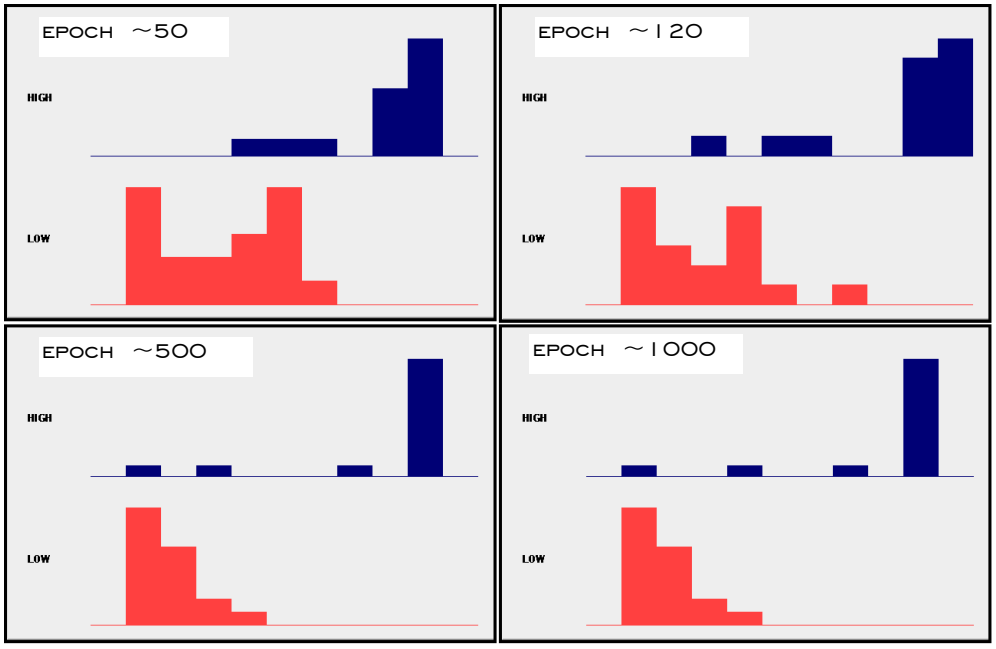


Figure III–12: A histogram of FD outputs after 50, 120, 500, and 1,000 epochs of training with 1:1 ratio of negative to positive exemplars.

overall, 13 samples are in the validation set and 32 samples (14 positive and 18 negative) are in the training set. In the first histogram pair of this case, the HIGH cases can be seen clustering to the right and the LOW cases clustering to the left. This trend continues with the clusters tightening around their goals through the following histograms. After epoch 1,000 (not shown here), the outliers continue to get "reeled-in" to their appropriate corners.

This example illustrates the skewed learning observed in ETA when different classes of output are represented in grossly different proportions in the training set. And in this framework, it is more important to learn correctly the positive cases than the negative ones, since the system continues automatically only when a candidate pixel evaluates as *"on–boundary"*. Thus, as a general guideline, only a balanced fraction of the negative exemplars should be used in training. In practice, this has not been a problem since even a modest user trace will generate a large number of data vectors; enough vectors so that some may be discarded with ample remaining to provide well-behaved statistics.

III.4 Input Representations

This section details an experiment designed to evaluate the hypothesis that suitable transformations of the input space can simplify the structure of the problem and thus speed the learning of its solution in the ETA. In the straightforward synthetic imagery discussed thus far, such as Figure III-7, simple inputs of five normalized pixel values were used. Realistic cases are more complicated, however, and require a larger set of neighborhood inputs to distinguish among the options.

In principle, a large neighborhood of normalized pixel values could be used as inputs, and the neural network would learn some set of filters on the input space which provides the best discrimination for the task. Given enough intermediate layers, enough

training data, and enough time, any filters or masks appropriate for the task could possibly be learned. While this is certainly a flexible approach, LeCun [1997], commenting on his hand-written digit recognition system, notes that allowing training to propagate back through many layers of a neural network took weeks of computation time. He also noted that the filters learned were basically those already known from basic image processing operations. In an application where the evaluation needs to be learned in a time scale of minutes rather than weeks, the problem can be avoided by fixing those weights deep in the network, fixing them based on primitive filtering operations that have already proven their utility.

In ETA, efficient learning is an issue, since one goal of this system is to keep pace with human operators. In these more complicated real-world cases, pre-processing data can make the problem easier and thus computationally quicker to solve. For example in statistics, log-transforming exponential data transforms it to a linear problem space with simpler solution methods. To explore this possibility within the ETA framework, experiments were designed to evaluate the hypothesis that suitable transformations of the input space can simplify the structure of the problem and thus speed the learning of its solution (Crawford-Hines & Anderson [1997a]).

The evaluation criteria used to determine whether one input representation is better than another is based on performance improvement in learning speed. System performance was quantified based on the error measured between the neural net's output and its target. The root mean square of that error (RMSE) across the training set was followed as it evolved over the training epochs. Two criteria for measuring learning speed are: (1) the number of epochs needed to come to within a percentage of the asymptotic RMSE value; and (2) the number of epochs needed to reach an RMSE of 0.1.

III.4.1 PRE-FILTERING THE INPUTS

Rather than simply using five normalized pixel values as inputs, taken at the locations defined in Figure III-3, a filter centered at each of those five locations uses further neighboring pixel values to come up with the filtered input values for the five locations. The experiment studied the learning speed of the task as a function of the input representation. The tasks are the edge-learning and line-learning tasks shown in Figure III-7. In addition to using single normalized pixel values, two input representations illustrated in Figure III–13 were chosen. These representations are basic filters common to both image processing and the neurobiology of vision. While there are many choices from which to select, these simple filters are sufficient to prove the hypothesis.

The center-surround filter is characteristic of early retinal processing and is a simple approximation to the classical Laplacian filter of image processing. This is a filter in standard usage, and the 3x3 size is the minimal that can be symmetrically placed. The oriented gradient is a directional filter, placed perpendicular to the direction of the boundary. This 1x5 filter is a simplified version of filters characteristic of the complex cells in the primary visual cortex (Kandel, et al., [1991]). The center weights were set so that coefficients in these three representations all sum to +1. All pixel inputs are converted to greyscale by averaging the RGB channels then normalized to the range [0,1] before any additional processing.

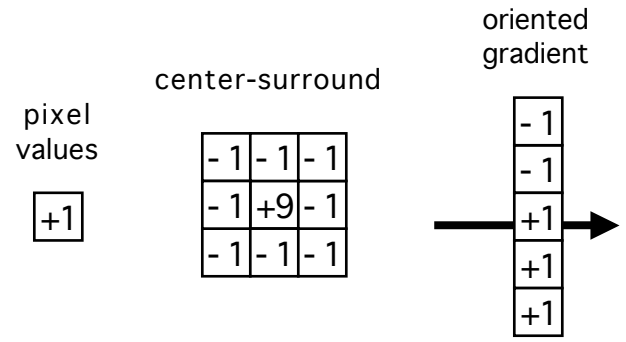


Figure III–13: Three input representations: the simple pixel, a non-directional center-surround filter, and a gradient filter oriented perpendicular to the boundary.

III.4.2 Representations Improve Learning Speed

In three separate trials, each of the input representations of Figure III–13 were used in the ETA system, and it was run through both learning tasks. The system was configured with a 5-10-1 neural network, to maximize the range of possible internal representations, and a feature-detector output unit. Parameters held constant across the runs were the initial training segment used, the pseudo-random network initialization, and the backpropagation learning and momentum rates. The learning and momentum rates were chosen, after trying several possibilities, as those that provided stable and fast learning.

The graphs of Figure III–14 summarize the results. The upper graph illustrates the learning curves for the edge-following task, and the lower curve illustrates the learning curves for the line-following task. For the edge-following task, the center-surround and oriented-gradient filters both reached the 0.1 RMSE criteria in just under 200 epochs versus approximately 900 epochs for raw pixel inputs, representing a 4.5-fold improvement in learning speed.

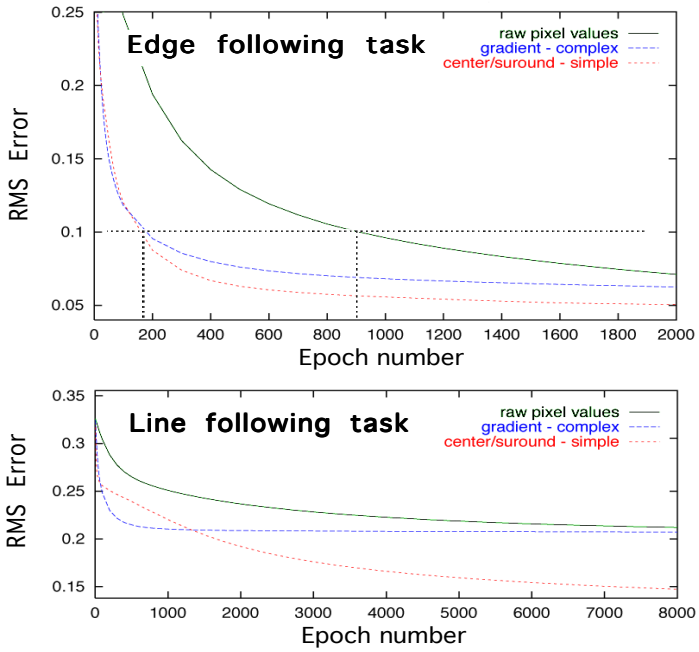


Figure III–14: Progress of the learning tasks for the input filters from Figure III–13 ; root mean square error is on the vertical axis, number of epochs is on the horizontal.

Using the percentage-of-asymptote criteria, visual inspection of the graph shows again a dramatic win for either filter over simple pixel values. For the edge-following task shown in the upper graph of Figure III-14, asymptotes can be approximated for the oriented-gradient (~.07) and center-surround (~.05) representations. The oriented-gradient filtered network came to within 10% of its asymptote (~0.77) at approximately epoch 400. The center-surround filtered network came to within 10% of its asymptote (~0.55) at approximately epoch 800. The center-surround has better asymptotic performance, but takes longer to get there. The asymptote for the raw-value case is not even on the map for this dataset, so the within-10%-of-asymptote epoch will certainly exceed both of the other cases. Quantification of these results depends on the percentage specified, and on training for many more epochs since the asymptotic value for the raw pixel learning curve is not obvious within the 2,000 epochs of the graph.

Although the system did learn to follow the line, neither curve reaches the 0.1 RMSE criteria in the allotted 8,000 epochs shown. This pre-selected criteria was too tight for this task, though the task was still learned successfully. The gradient filter, with a high asymptotic RMSE value, gets to its asymptote first, in roughly 800 epochs, compared to the unfiltered input case which more slowly approaches basically the same RMSE asymptote. The center-surround filter reaches an overall better result, though at roughly the same speed as the unfiltered inputs. While not as clearly an improvement, the filtered inputs can be said to either get the system to the same (bad) solution much quicker, or to a better solution at the same rate. This is still an overall win.

The conclusion is that the filters are better than straight pixel values, since they allow the system to either find solutions faster or with better accuracy. Though simplistic in its constructs, this experiment is an existence proof that in the ETA framework, improving the input space representation can improve the system's learning speed and lead to

better solutions. This now leads to the question, "If one is going to re-cast the input space of a problem by somehow transforming the data, what transforms are appropriate for the task?" This question will be explored further in Section III.5.3.

III.5 Interpreting the Hidden Layer's Learning

By filtering the raw pixel data, essentially another layer is added to the neural network, as shown in Figure III-15. This new layer has a somewhat different character than the three neural network layers, however, since the weights are fixed in this new layer by the filtering defined on the inputs. Also, the transformed inputs are combinations of subsets of pixels rather than all pixels, as would be the case in a fully connected network with two hidden layers.

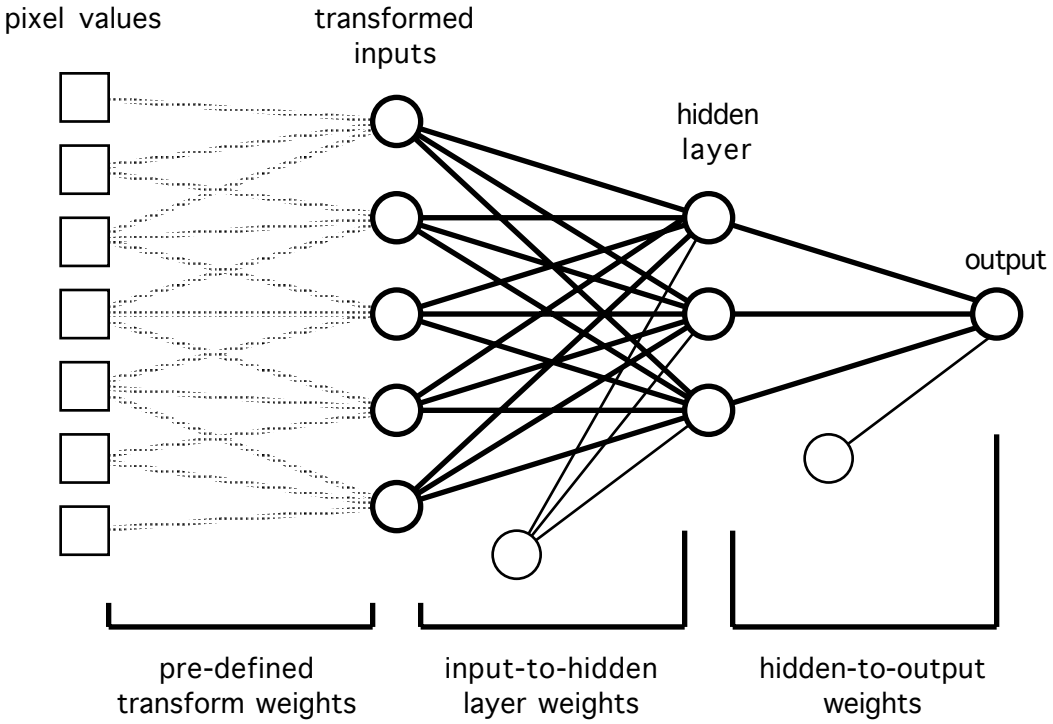


Figure III–15: Filtering the inputs effectively adds another layer to the network architecture, with the predefined filters viewed as an ealier set of network weights.

One general criticism of neural networks is that what they learn is opaque. This section details some exploratory work done in an effort to understand what is being learned. The input representation used was motivated by receptors found in the early visual cortex that respond most strongly to the visual stimulus of bars at specific orientations. The "bar-detector" cells have a preferred response only to a bar of a certain length and orientation on the visual field. The visual system thus needs bar detectors at a variety of orientations since a stimulus might appear at any angle in the visual world. In the ETA framework, though, the world is effectively normalized in advance by the direction established for the boundary in progress. Everything is calculated with respect to that orientation. Thus in ETA, a bar detector is needed in only one orientation. For each of the five standard candidates from Figure III-3, a bar parallel to the direction of the boundary is used. This bar is a five-pixel window, where output of the bar filter is the average over all the pixels in this window.

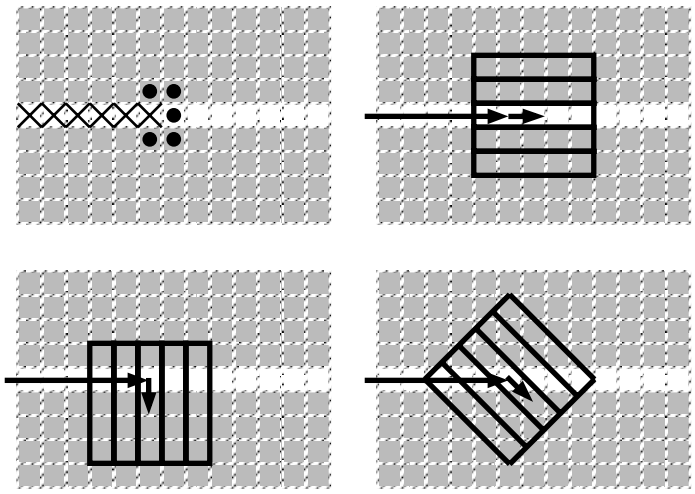


Figure III–16: The upper left shows a line-continuation task in progress (the X's) and the five candidate next pixels. The remaining three graphs show the coverage of five 5bar filters (averages over 5 pixels in a row) for the lower three candidate next-pixels.

Figure III-16 illustrates the basic input filtering. In the upper left, the pixels defined to be a part of the boundary are marked with X's, and the five possible next pixels are show as solid dots, following the structure laid out in Figure III-1. In the upper right is the "true", straight-ahead case; the small arrow indicates the direction the boundary would be taking in this proposed case, and the five bars are parallel to that arrow, centered on the candidate point, its two left neighbors and its

two right neighbors. The pixel values within the bar are summed and re-normalized to [0,1]. The lower two diagrams of Figure III-16 illustrate how the bars are aligned for the two incorrect off-to-the-right candidates; the two off-to-the-left candidates are mirror images of these bottom two cases.

The bars can be of various dimensions, for example 3x1 or 5x1 or 7x3. The shorthand notation for the five windows each averaging 5 pixels in Figure III-16 is *"5-5bar"*.

III.5.1 Reading Filters Off the Hidden Layer

Using the 5-5bar input representation, the system was trained with a 5-1-1 neural net on the noisy line of Figure III-10. Only one hidden unit was used to force the network to learn its most compact intermediate representation. After training, the neural network's weight configuration is graphically represented in Figure III-17.

The weights resulting from one training session are illustrated in Figure III–17. They are drawn so that the largest weight will always be the same size, and other weights are scaled in area proportionally to the largest. This scaling allows for quick visual scanning of the overall weight configuration. Light implies a positive weight, and dark implies a negative weight. The first six weights shown are the weights from the five inputs and the one bias unit to the one hidden unit.

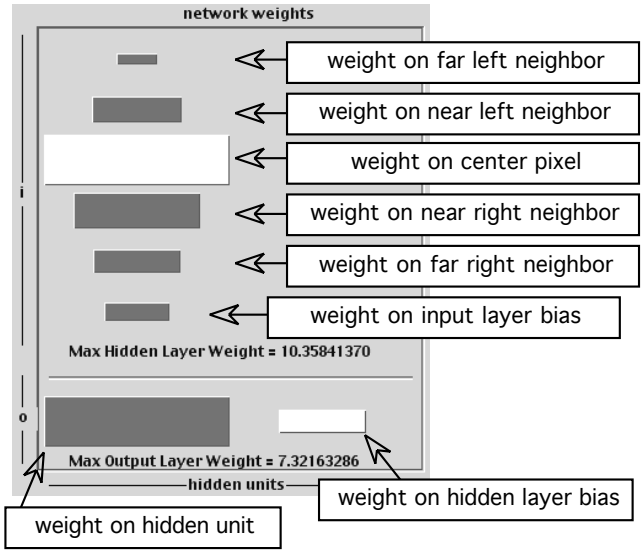


Figure III–17: Weight representation for the noisy line-following task using 5-5bar inputs. Light implies positive, dark implies negative weight; weights all scaled in proportion to the largest.

The two weights shown in the bottom row are the weights from the one hidden unit and the one bias unit to the one output unit.

The weights as shown in Figure III–17 can be easily linked to a common shorthand notation for well-known filters. Filters are often presented visually by an image of their intensity function. Figure III–18 shows a shorthand sketch, where, for example, an on-center, off-surround filter would be sketched as a white circle surrounded by a dark torus. The other two filters of Figure III–18 schematically show oriented second and third order Gabor filters (see their use, for example, in Freeman & Adelson [1991]).

Figure III–18: Shorthand sketches of center-surround (left), second order Gabor (middle), and third order Gabor (right) filters.

These filter visualizations help interpret the weight representation of Figure III–17. Looking at the top five weights (ignoring the sixth bias weight), visually a roughly symmetrical, oriented filter can be seen, similar to a third-order Gabor. In this system, the hidden layer has a natural interpretation as a filter, whose characteristics can be read off the network weights. This stands in marked contrast to the "opaque learning" of which neural networks are often critiqued.

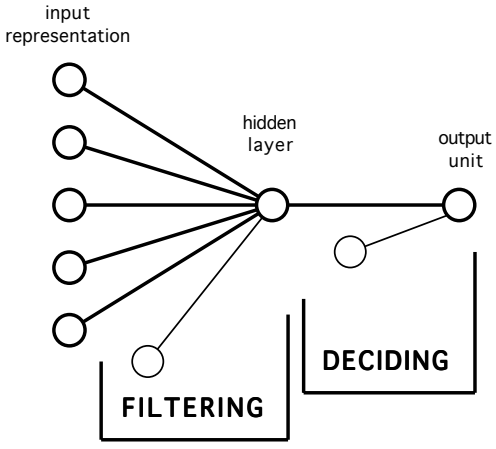


Figure III–19: The neural network architecture interpreted as filtering and deciding tasks.

As depicted in Figure III–19, each hidden unit, summing the results of the weights times the inputs, is behaving as a filter. And the output unit, with its incoming weights on those filter(s), is making an on-boundary / off-boundary decision based on the filters.

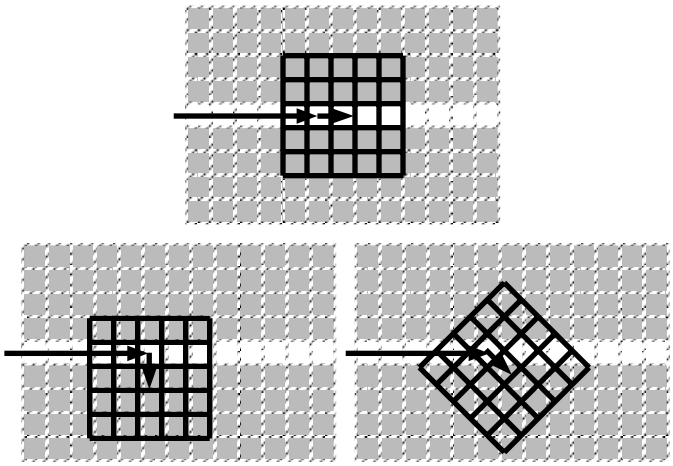


Figure III–20: Single pixel field, equivalent to coverage of Figure III–16.

To discern whether this learned, Gabor-like filter was a result of the 5-bar input representation, the experiment was re-run, this time using the 25 pixel value inputs that covered the same area as the five 5-bars. Figure III-20 shows this new input space, which is used for the same task as before, with a 25-1-1 neural network. The resulting weight diagram for one trial is shown here in Figure III-21; other random network initializations resulted in a similar picture.

Boxes have been placed around the five weights on the pixels that would have been aggregated by the 5-bar inputs. Visually summing the weights in each of the five boxes results in a filter consistent with Figure III–17. The same pattern emerges in both cases, however, the Gabor-like filter obvious in Figure III–17 is not immediately apparent here. This configuration is less desirable from a performance standpoint, as well. This 25-1-1 network has 28 free weights to learn as opposed to the 8 free weights of the 5-1-1 network, which thus implies both more computations per weight update and network evaluation, and

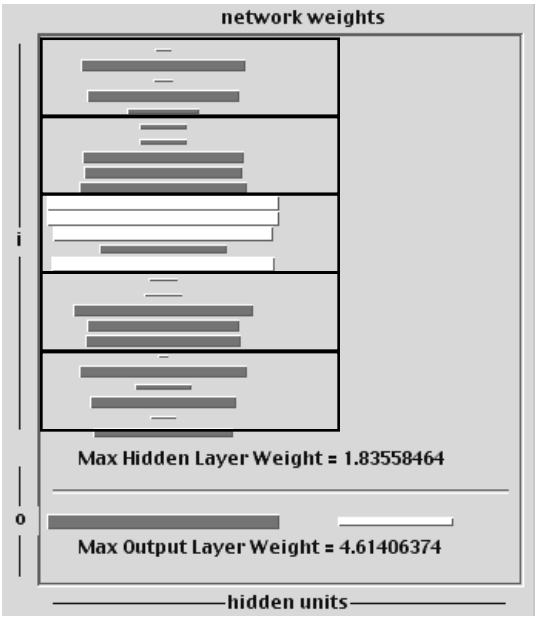


Figure III–21: Noisy line-learning with raw pixel inputs and a 25-1-1 network; weights are grouped in fives to show the correspondence to Figure III–17.

correspondingly more data needed in the training set. The filtered pre-processing is preferable from both a computational perspective and with a view to easy interpretation.

III.5.2 Boundary Extension Issue for Directional Inputs

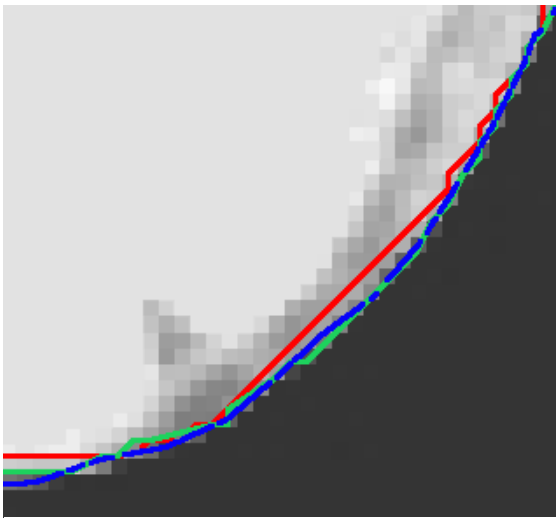


Figure III–22: A manual trace (blue) and an automated trace (red) that prefers the diagonal rather than the boundary; considering multiple directions (green) improves the tracing quality.

When using directional filters, such as the 5bar filter, problematic behaviors were observed in certain situations. In imagery with slowly curving surfaces and at places where boundaries are close to horizontal, vertical, or 45-degree diagonal, sometimes ETA was observed to trace straight ahead rather than be drawn to the appropriate edge. The red boundary shown in Figure III–22 shows such behavior, compared to the manually traced boundary shown in blue. The problem arises, as explained below, due to only considering one orientation for the filter at each possible new candidate point. Considering multiple orientations at each candidate pixel solves the problem, as shown by the green contour.

This trouble arises from the basic extension algorithm. Figure III–23 shows the choices considered for a next point. A boundary in-progress is illustrated by the shaded pixels, with the thick vector indicating the direction established by the edge so far. The five choices for consideration as the next point, along with the direction such a choice would imply, are numbered.

Consider tracing a nearly diagonal boundary. When the true boundary differs only slightly from 45 degrees, the edge is not truly represented by choice 2, but 2 is a better choice than 1 or 3 which are both further from the truth. Over several pixels, though, this slight error accumulates and the behavior illustrated by the red line in Figure III–22 results. For this specific example, the solution is to also consider pixel 3 with a direction vector of (1,1) in addition to the choice of pixel 2 with a (1,1) direction.

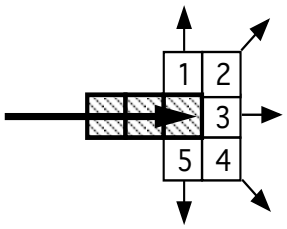


Figure III–23: One direction considered per candidate pixel.

More generally, the solution is to consider more directions at the possible points. Instead of only considering candidate 2 with directional inputs oriented with respect to (1,1), consider candidate 2 three times, each time using a different vector of the three shown associated with candidate 2 in Figure III–24. Similarly, multiple directions are considered at the other candidate points as well. All together there are now thirteen distinct choices, each choice a distinct combination of a candidate point combined with a direction. Using more directional options results in a trace much closer to its truth – the green boundary in Figure III–22.

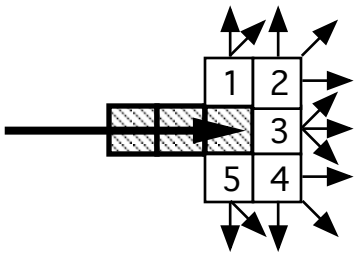


Figure III–24: Considering multiple directions per candidate pixel.

This situation arises only when there is exclusive use of directed filters in the input space. When the input features used are non-directional filters (e.g., raw pixel value, or symmetric center-surround fields), this straying behavior does not arise, since the direction at a point was irrelevant to the input values used there.

III.5.3 Defining A Robust Input Filter Set

Section III.4 showed that re-representing the input space can improve the system's learning speed. For this, there is a wealth of prior work in the domains of neural science and image processing. From the initial studies of animal visual systems, Kandel, et al., [1991] summarized some of the robust primitive transforms performed by mammalian systems, from the opponent center-surround behaviors of retinal bipolar cells, to the oriented filter responses characteristic of the ocular dominance columns in the primary visual cortex. From the decades of work in image processing, many operators and techniques have demonstrated their usefulness, from Sobel filters to Fourier analysis to morphological operators, in defining the boundaries of structures in an image.

If one is going to re-cast the input space of a problem by somehow transforming the data, what transform, from the large set of transforms available to us, is appropriate and best for the task? Since ETA was shown to combine simple filters into more complex filters as needed, the system can be initialized with a well-rounded set of basic filters, and the complex filters appropriate for a given task can be assembled from these building blocks. The appropriate combination of filters for the task is learned rather than hard-coded.

Linear combinations of Gaussian kernels at various scales and offsets form a basis for a wide range of filters. Malik and Perona [1992] showed these basic filters to be sufficient for recognition of a broad range textures. The difference-of-Gaussian (DoG) filter models the center/surround character of retinal ganglion cells. Oriented responses typical of area V1 in the the visual cortex can be modeled as differences-of-offset-Gaussians (edge detectors) and differences-of-offset-DoGs (bar detectors). They can be used at various scales to account for coarse and fine structure. And linear combinations

of these functions is exactly the function of the hidden units in the neural network architecture. Thus using Gaussian kernels as inputs, at multiple scales for each candidate point and its neighbors, allows for the flexible combination into higher-order filters.

A robust input filter set can thus be designed around flexible basic primitives. For the comparison experimentation in the following chapters, the following set of filters proved robust across a wide range of real imagery. A set of Gaussians were used at different scales, with σ of 1.0 and 1.5, and at different offsets. Bar filters of size 5x1 and 7x3 at different offsets were used as the basic directional filter building blocks. And the pixel value itself was used as the most fine-grained input.

III.6 Sensitivity to Hidden Layer Size and Weight Initialization

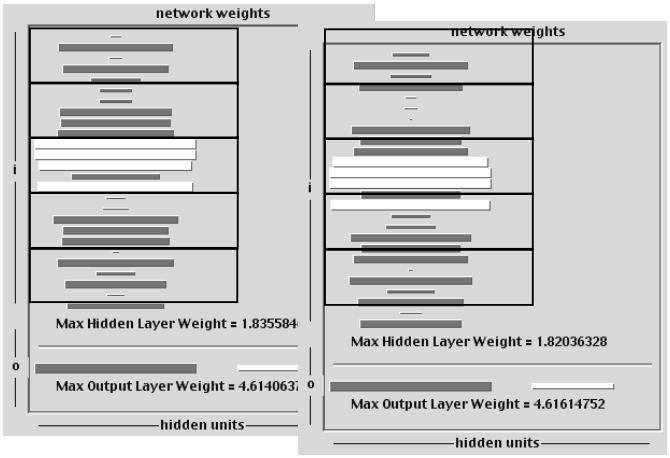


Figure III–25: Two separate trials differing in only network initialization of the 25-1-1 noisy line-learning task shown in Figure III-21. The weights are not identical due to the different initialization, but the overall pattern is the same.

The sensitivity of these results to initializations and to the number of hidden units was examined, for the example of Section II.5.1. Similar sensitivity analyses were done on real imagery with similar results (Section V.2.3). Figure III–25 illustrates the difference attributable to two different weight initializations for the 25-1-1 noisy line task. Several different initializations were tried and the same basic pattern emerged independent of the initial weights. These two weight sets typify observed differences.

Figure III-26 shows the weights that result when more than one hidden unit is used. The first column represents the weights from the 25 input units to the first hidden unit, the second column represents the weights to the second hidden unit, and so on. Each hidden unit can be seen to be learning basically the same filter. This was similarly observed at several different initializations. Since the behavior seen in the hidden layer is robust to the number of hidden units, this suggests only one hidden unit is needed for solving this problem.

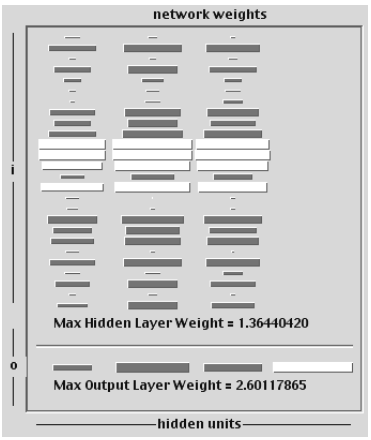


Figure III–26: Weights for a 25-3-1 network on the same task shows similar weight behaviors across all hidden units.

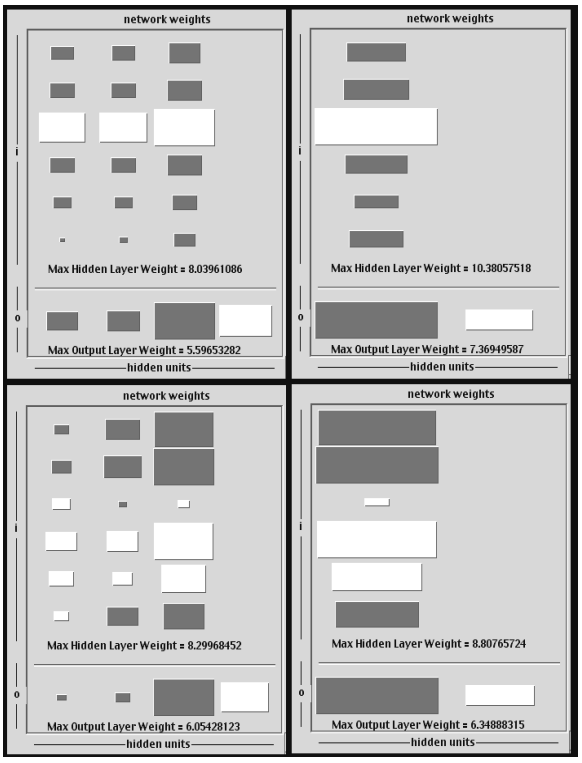


Figure III–27: Comparing the effect of varying the number of hidden units for the line following task (above) and a similar edge-following task (below).

Figure III–27 shows a comparison on the noisy-line task of Figure III–10 for three hidden units versus one hidden unit (on the top) and a similar noisy-edge task comparison (on the bottom). In both cases, five 5bar inputs were used. Since the networks are of different sizes, this implies different network initializations, as well.

For the line-following task (on the top), the weight patterns are very similar across all the hidden units. For the edge-following task (on the bottom), the training exemplar wanders slightly back

and forth on first the dark side of the edge then the light side. Because of this, the middle of the five inputs has little edge-discrimination value (since it is sometimes light and sometimes dark) and hence the weight on this unit doesn't grow. Within the bounds of this slight variation, though, the three hidden units appear to be learning the same edge-filtering behavior.

In these simple, synthetic tasks where there is only one thing to learn, one hidden unit suffices. As will be seen in the comparison study on real-world imagery in Chapter V, when structural boundaries have different characteristics across their lengths, for example a bone with boundaries against both muscle and ligament, different hidden units may learn different behaviors, possibly to help in that discrimination.

III.7 Summary

This section has outlined the basic ETA framework of neural network learning, trained with inputs from positive and negative exemplars constructed from boundary neighborhoods. Experimentation with two output representations led to insights into their choice of FD output units. Studies of the training set proportions led to more robust procedures for a balanced training regimen and thus better learning. Explorations of input representations provided interpretations of what and how the ETA system is learning, which led to a flexibly configured set of input filters. As a result of these experimental findings, the ETA framework used for the comparative studies in the next two chapters was designed with FD output units, balanced positive and negative exemplars, and input filter set as summarized at the end of Section III.5.

——== Chapter IV ==——

Comparing Systems and Experts: Methodology

To understand the relative merits of learning boundary contours, the Expert's Tracing Assistant (ETA) was compared to other user-guided methods representing the current state-of-the-practice for boundary delineation. The techniques of Active Contour Models (ACM) and Intelligent Scissors (IS) were chosen for comparison to ETA because they have been brought into practice, they have been studied and refined in the literature, and they represent benchmarks against which other novel methods are compared. The ground truth is an expert's manual tracing of a structure's boundary in an image. The **`GVFsnake`** software of Xu and Prince [1997] was used to generate the ACM boundaries in this study. The IS boundaries were made by Eric Mortensen using his IS software during a technical visit to BYU.

In medical images, some structures are neither sharp edged nor reliably different in color from the surrounding structures. These are the cases where a basic visual judgement or an expert's anatomical knowledge are needed to complete the boundary. The hypothesis underlying the ETA is that due to the ambiguities in medical images which confound the state of the practice, a method designed to align itself to a human's decisions can be of greater assistance in boundary tracing tasks. This study is designed to expose the relative strengths and weaknesses of the methods considered. This chapter presents the methodology of the study, and the following chapter presents the results.

The structures chosen for comparison were taken from the imagery of the Visible Human dataset, which is also a benchmark set upon which many image processing and visualization methods have been exercised. Several structures were selected as a representative cross-section, and for each, the IS, ACM, and ETA methods were used to define its boundary, and an expert manually delineated the boundary in two independent trials. Measures were derived to quantify the inter-curve distances by selecting a tolerance such as one pixel and stating, *"The curves are within one pixel of each other 86% of the time"* or by selecting a percentile such as 90% and stating, *"The curves are within 1.1 pixels of each other 90% of the time"*. These are the key measures of the differences between the boundaries.

Section IV.1 details the structures and imagery used. Section IV.2 discusses the IS, ACM, and ETA methods and how they were configured. Section IV.3 presents the statistical methodology used to compare the resultant boundaries. The results of the comparison are presented in Chapter V.

IV.1 Comparison Structures and Imagery

Figure IV–1 shows an enlarged greyscale image from the visible male dataset of the cross-section of the thoracic aorta. The boundaries defined by the ETA (in cyan), ACM (in yellow), and IS (in magenta) methods are shown superimposed. The interior wall of the aorta is straightforward to define in this image, since both the region inside the aorta and the arterial wall are homogeneous in their pixel characteristics. A quick visual comparison shows the three methods are all close to one another, each defining essentially the same boundary.

In this straightforward example, the boundaries are so similar there is little basis for favoring one method over another. Thus, in selecting structures for comparison from the

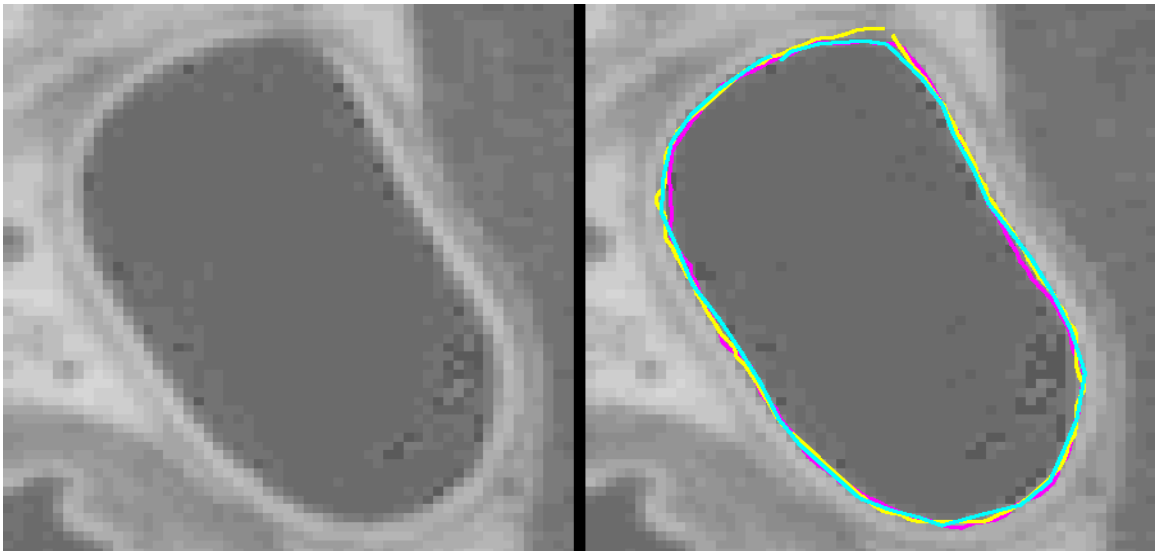


Figure IV–1: The raw image is shown at left; at right are superimposed contours of ETA (cyan), IS (magenta) and ACM (yellow), all very close to one another.

Visible Male imagery, the criteria are to select: (1) representative structures of general interest, and (2) more challenging structures than the simple example in Figure IV–1 so as to better differentiate among the methods.

Nine structures were selected on three images, and the images were cropped to the structures of interest to minimize the number of extraneous pixels. The three images, shown in Figure IV–3, Figure IV–4, and Figure IV–5, were taken at the approximate cross-section locations shown in Figure IV–2. The selected structures of interest are outlined in blue, with the outline drawn several pixels to the outside of the structure of interest so that the pixel character of the boundary itself can be visually studied in the image.

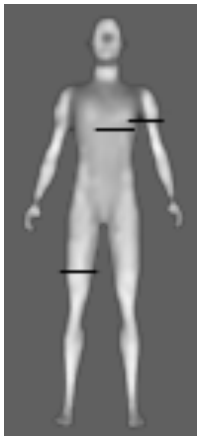


Figure IV–2: The three cross-sectional images studied were taken where indicated from the arm, thorax, and leg.

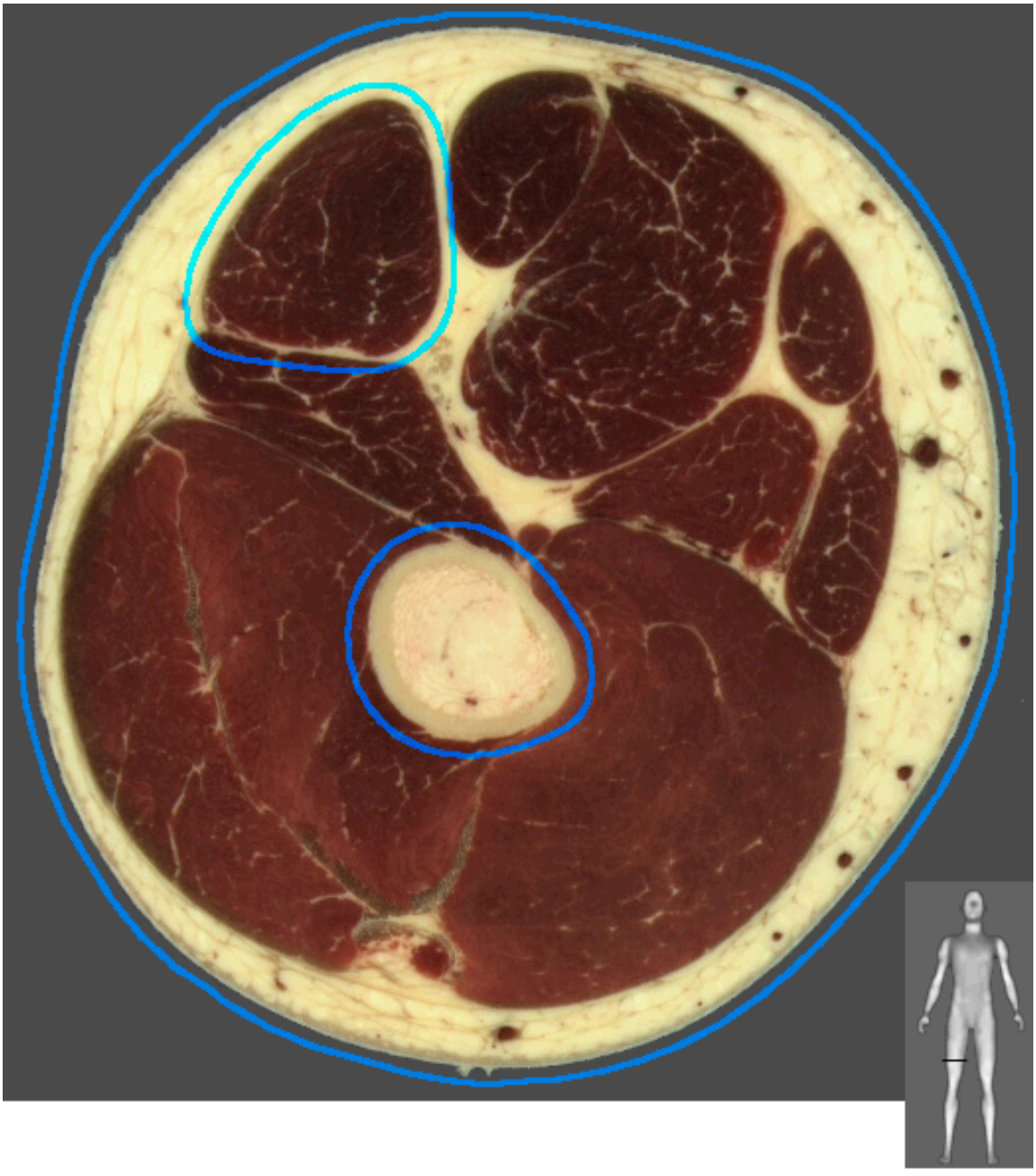


Figure IV–3: The three selected structures outlined are the femur (bone), the biceps femoris (muscle), and the skin, on transverse image #2186 through the leg.

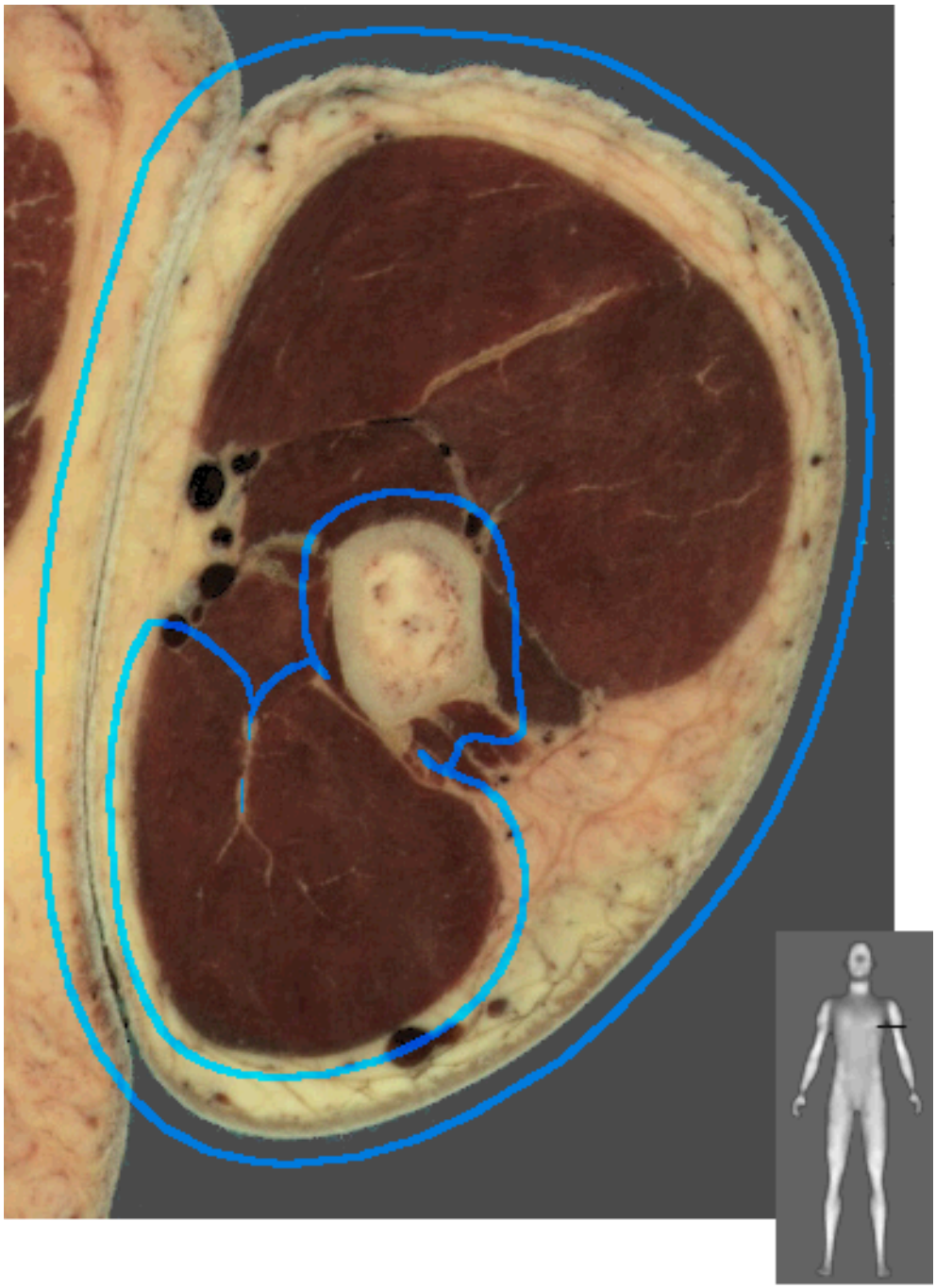


Figure IV–4: The three selected structures outlined are the humerus (bone), the biceps brachii (muscle), and the skin, on transverse image #1430 through the arm.

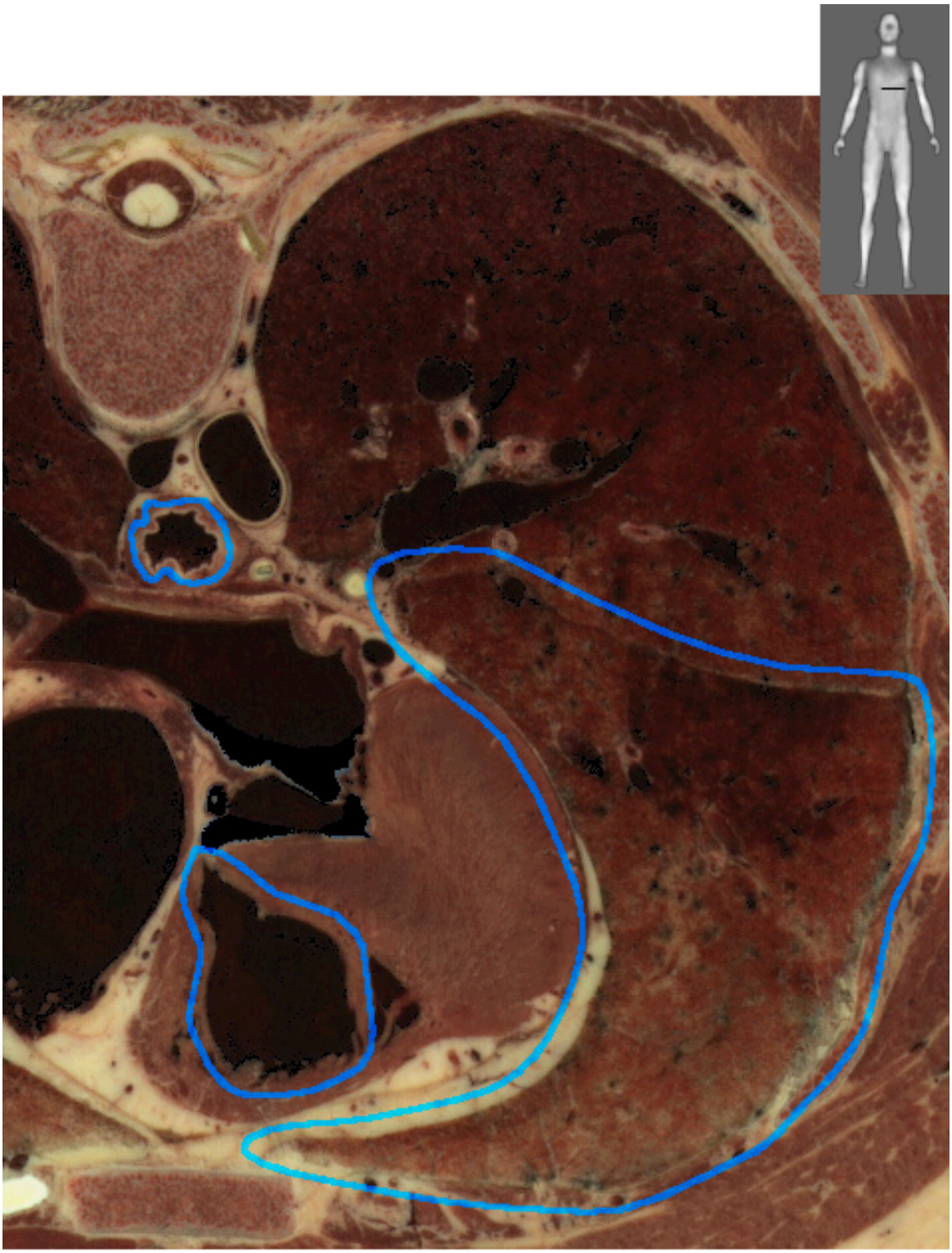


Figure IV–5: The three selected structures outlined are the esophagus, the right ventricle of the heart, and the upper lobe of the right lung, on transverse image #1432 through the thorax.

Spitzer and Whitlock [1998] cataloged the Visible Male sectional imagery, and the image numbers cited along with the images are their reference numbers. Figure IV–3, from transverse image #2186 at the leg, shows the femur (bone), the biceps femoris (muscle), and the skin. Figure IV–4, from transverse image #1430 at the arm, shows the humerus (bone), the biceps brachii (muscle), and the skin. Figure IV–5, from transverse image #1432 at the thorax, shows the esophagus, the right ventricle of the heart, and the upper lobe of the right lung. Skin, muscle and bone contours are most common, and both simple and complex examples of each are included in this set.

The leg image, Figure IV–3, represents reasonably straightforward cases. The leg bone and skin are shown clearly, without much confusion. The leg muscle is fairly typical, surrounded mostly by highly contrasting fatty tissue, however sometimes with only a thin channel of it between one muscle and the next. The arm image, Figure IV–4, represents more complicated versions of these structures. Connective tissue attached to the bone, at the bottom of the bone as this image is oriented, is difficult to distinguish from the bone itself. The arm's skin includes a very indistinct boundary in the armpit area, where the arm and chest are touching; this almost straight line is visually apparent, but progressively harder to distinguish in detail the further it extends toward the top of the image. Another indistinct boundary is exemplified by the upper-left of the muscle, where any boundary drawn is based on little evidence in the image; there's almost no variation at the pixel level indicating a boundary of any sort.

In the thorax cross-section, Figure IV–5, a ventricle of the heart was selected since, as a crucial body part, it is an often-studied structure in the medical imagery literature. The esophagus exemplifies structures with convoluted rather than smooth boundaries. The lobe of the lung shows much of the complex variety of soft tissue. The lung is bounded by structures each with different visual character – heart, fat, rib bone, cartilage, and

other lung – and these distinctions are often visually non-obvious and require expert judgement to adequately segment.

IV.2 Configuration of the Boundary Methods

On each of the nine structures of interest, the Expert's Tracing Assistant (ETA), Active Contour Models (ACM) and Intelligent Scissors (IS) were used to define the boundary. These methods were compared to one another, and to the ground truth of an expert's manual tracing of the boundary. Section IV.2.1 provides the technical background and implementation details for IS; Section IV.2.2 provides the technical background and implementation details for ACM. The methodology of ETA has been discussed at length in Chapter III, and Section IV.2.3 adds the specific details of how it was applied in bounding these nine structures. Section IV.2.4 discusses how the expert manually delineated the structures.

IV.2.1 INTELLIGENT SCISSORS (IS)

The user-guided "Intelligent Scissors" of Mortensen and Barrett was summarily reviewed in Section II.2.3. This section adds further detail relevant to the comparison study. Across the image, a local cost from every pixel to its eight neighbors is pre-computed. This local cost is a weighted sum of several features, all scaled so that strong edges result in low values. As the user places control points on boundaries of interest, the system computes a minimal cost path from the most recent control point to every pixel in the image. As the cursor is moved from pixel to pixel in the image, the optimal path from control point to cursor is quickly redrawn, providing feedback to the expert on the boundary definition as it is in progress.

The local cost function is a weighted sum of component cost functions. Following the exposition of Mortensen and Barrett [1998], if $\boldsymbol{p}$ and $\boldsymbol{q}$ are neighboring pixels, the local cost between them, $l(\boldsymbol{p},\boldsymbol{q})$, is defined as

$$l(\boldsymbol{p},\boldsymbol{q}) = \omega_Z f_Z(\boldsymbol{q}) + \omega_G f_G(\boldsymbol{q}) + \omega_D f_D(\boldsymbol{p},\boldsymbol{q}) + \omega_P f_P(\boldsymbol{q}) + \omega_I f_I(\boldsymbol{q}) + \omega_O f_O(\boldsymbol{q}) \ .$$

Empirically derived weights which, as the authors note, "work well in a wide range of images", are

$$\omega_Z = 0.3 \ , \quad \omega_G = 0.3 \ , \quad \omega_D = 0.1 \ , \quad \omega_P = 0.1 \ , \quad \omega_I = 0.1 \ , \quad \omega_O = 0.1 \ .$$

The component cost functions are defined as follows.

$f_Z(\boldsymbol{q})$: The Laplacian zero-crossing component is taken at several possible scales, from 5x5 to 15x15, to balance the sensitivity to fine detail of small kernels with the noise suppression of larger kernels. A zero crossing is defined as a near-zero value for a pixel whose neighbor has a value larger in magnitude and opposite in sign. A binary function is defined at each scale: if a pixel is on a zero-crossing, then that cost to all neighboring links is zero, otherwise it is one. The final cost function f_Z is a linear combination of the binary functions.

$f_G(\boldsymbol{q})$: The gradient magnitude component is also derived from information at several scales. At each scale, the gradient is approximated by the root sum of squares of partial derivatives in x and y of Gaussian kernels at that scale, then inverted and normalized to [0,1]. If the pixel has been determined to be a zero crossing, f_Z takes the value of the kernel for which the Laplacian at that same size produces the steepest slope. If not a zero crossing, the value for the 3x3 kernel is used.

$f_D(\boldsymbol{p},\boldsymbol{q})$: The gradient direction is calculated as the root sum of squares of the gradient's partial derivatives. This cost feature is set to a low value when the gradient direction of the two neighboring pixels are similar to each other and to the link between them. This adds a smoothness constraint by adding a high cost to sharp changes in direction.

$f_P(\boldsymbol{q})$, $f_I(\boldsymbol{q})$, $f_O(\boldsymbol{q})$: These functions are based on the value of the pixel itself, a pixel to the "Inside" of the boundary, and a pixel to the "Outside" of the boundary. A history of 32 to 62 values for these pixels is tracked, and if pixel $\boldsymbol{q}$ is consistent with these values, $f(\boldsymbol{q})$ is set to a low value, otherwise it is set to a high value. Comparing the current pixel value to recent history is an adaptive measure that allows the overall cost function to be sensitive to local circumstance.

The user-interface of this tool has been refined over time. For instance, as the user moves the cursor around, some portion of the path nearest the set control point will remain constant. This portion will be "frozen" into place by automatically generating a new set point further along the boundary.

Because the IS boundary points are all on pixel centers, an IS defined boundary will have a more rough look than that of a smooth curve, as shown in the upper half of Figure IV–6. This problem arose in the user acceptance of ETA, until a weighted-average smoothing was implemented. The weighted-average gave the boundary a smoother character, like manually defined boundaries

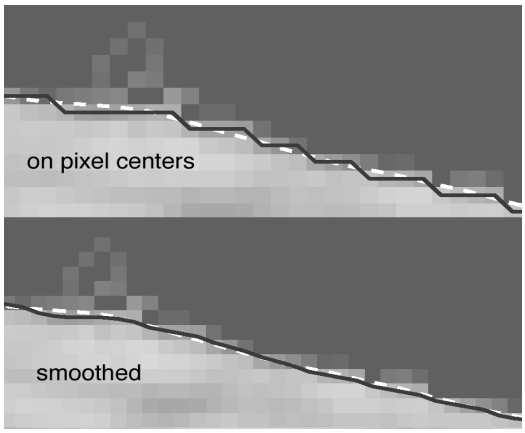


Figure IV–6: On top is an integer-valued IS boundary, drawn to pixel centers, superimposed on a smooth boundary (the dotted white line); the bottom shows the IS boundary after smoothing.

traced at high magnifications to allow sub-pixel resolution. The simple geometric average used to smooth each point replaced each point $\boldsymbol{p}_i$ by

$$0.1\ \boldsymbol{p}_{i-2} \ + \ 0.2\ \boldsymbol{p}_{i-1} \ + \ 0.4\ \boldsymbol{p}_i \ + \ 0.2\ \boldsymbol{p}_{i+1} \ + \ 0.1\ \boldsymbol{p}_{i+2} \ .$$

This straightforward smoothing resulted in greater acceptance of ETA-drawn contours by expert users. For consistency in comparison, this smoothing was added as a post-processing step to the IS contours as well. The results are illustrated in the lower half of Figure IV–6.

IV.2.2 Active Contour Models (ACM)

The user-guided, Active Contour Models of Kass, et al. [1987] were summarily reviewed in Section II.2.3. This section adds further detail relevant to the comparison study. The ACM is an energy minimizing spline, which is initialized close to a structure of interest and then iterated to a local minima. The energy function is defined so the minima correspond to boundaries of interest in images. Since the ACM is initialized with a closed curve, the final result will always be a closed, continuous curve.

The boundary contour is represented parametrically by a point-valued function $\boldsymbol{v}(s)$ and the shape of the contour is defined to have an energy, $E(\boldsymbol{v})$, defined as

$$\boldsymbol{E}(\boldsymbol{v}(s)) = \boldsymbol{S}(\boldsymbol{v}(s)) \ + \ \boldsymbol{P}(\boldsymbol{v}(s)) \ , \quad \text{where} \quad \boldsymbol{v}(s) = (\ x(s),\ y(s)\ ) \ \text{ for } \ s \ \text{ on } [0,1] \ .$$

The first term in the sum is a measure of the geometry of the boundary's shape, and the second term is derived from the character of the image pixels on which the boundary is superimposed. The shape contribution is defined as

$$S(\boldsymbol{v}(s)) = \int_0^1 \alpha(s)\,|\boldsymbol{v}'(s)|^2 + \beta(s)\,|\boldsymbol{v}''(s)|^2\,ds\,.$$

The function $\alpha(s)$ controls the contribution of the first-derivative of the curvature, $\boldsymbol{v}'(s)$, physically interpreted as the overall "tension" of the parametric curve, and $\beta(s)$ controls the contribution of the second derivative, $\boldsymbol{v}''(s)$, interpreted as the "rigidity" of the parametric curve. The image contribution is defined as

$$P(\boldsymbol{v}(s)) = \int_0^1 p(\boldsymbol{v}(s))\,ds\,,$$

where $p(\boldsymbol{v}(s))$ is a scalar function defined on the image plane such that its local minima correspond to intensity extrema, edges, or other features of interest in the image. The gradient of the image is large where image edges are strong, thus for $p(\boldsymbol{v}(s))$ the gradient can be inverted so that minimal values represent features of interest.

This formulation can be transformed to a dynamic system by adding a time dependency to $\boldsymbol{v}(s,t)$. By setting up Lagrangian equations for this dynamic system, equilibrium conditions can be established for local minima of the overall energy. This introduces two new parameters, γ and κ, that weight the first and second partial derivatives of $\boldsymbol{v}(s,t)$ over time, interpreted as the damping and inertial forces of the dynamic system.

When solving this dynamic system, $\nabla \boldsymbol{P}$ pulls the parametric curve toward edges in the image. Unfortunately, the “capture range” of traditional edge strength functions is small. The capture range depends in part upon the amount of smoothing built into the edge mask, and when using typical edge masks, the influence of an edge only reaches a

few pixels. This implies that the parametric curve must be initialized close to the desired boundary of interest, otherwise $\boldsymbol{P}$ will have no influence on the dynamic evolution of the parametric curve.

Xu and Prince [1997] replaced $\nabla \boldsymbol{P}$ by a vector field throughout the image. This Gradient Vector Flow (GVF) improves the capture range of the dynamic system, by pointing to the nearest strong edges in areas of image homogeneity where distant edge information would otherwise be lacking. This improves the range over which the parametric curve can vary and still be attracted to a minimum and helps the curve settle into concave boundary regions. They implemented their GVF model along with traditional snake and balloon models in Matlab and released their code for general use. This **`GVFsnake`** software (Xu and Prince [1999]) was used to generate the ACM models in this study.

The **`GVFsnake`** software starts with an image, an edge map, and an initial boundary, and then iterates the dynamic system until it settles into a minimum. The edge map used in the software is a scalar function over the image pixels that is high at desirable image points (in this case, image edges). Displaying the curves at periodic iterations provides a visualization of the evolution of the parametric curve to its equilibrium state. This implementation uses a constant value for $\alpha(s)$ and $\beta(s)$.

The parameters through which the user can control the ACM and its evolution through time are α, β, κ, γ, and μ. Figure IV–7 and Figure IV–8 show the effect of varying these parameters individually. Each of the small graphics is a detail of the ACM over a sequence of 30 iterations: the green line shows the initialization supplied by the user, the red line shows the state of the curve after 30 iterations, and the sequence of yellow lines shows the curve at five iteration intervals in between. This collection of graphics helps

provide further understanding of how these parameters influence the ACM and its evolution over time.

α : This parameter is referred to as controlling the overall "tension" of the parametric curve; increasing it favors boundaries of shorter length. As seen in the top row of Figure IV–7; increasing α eliminates extraneous loops and ripples. A reasonable initial choice for α is 1.0, which may be increased when a specific circumstance calls for more smoothing.

β : This parameter is referred to as controlling the overall "rigidity" of the parametric curve. As shown in the second row of Figure IV–7, when set to zero, the boundary can make sharp angles; increasing β introduces a smoothing curvature at these sharp angular places. The initial choice for β is 0.0 so the contour can find its way into tight corners.

κ : This parameter is referred to as controlling the inertial force of the dynamic system, the weight on $\partial^2 v/\partial t^2$. In the third row of Figure IV–7, note the change in spacing between the curves at different iterations. Higher values push the curve further in each iteration, and a value too high (in this case at 2.5) will result in poor temporal behavior, which can be seen here as the curve passes by a good boundary solution to a poorer one. The initial choice for κ is 0.5; this avoids the squirrelly behavior of high values and the slow convergence of low values.

γ : This parameter is referred to as controlling the "viscosity", or damping force, of the dynamic system, the weight on $\partial v/\partial t$. Figure IV–8 shows that at higher values, the curves move less during a set of iterations. The middle row shows that a high value (in this case 4.0) can be useful in tracking the long thin intrusion

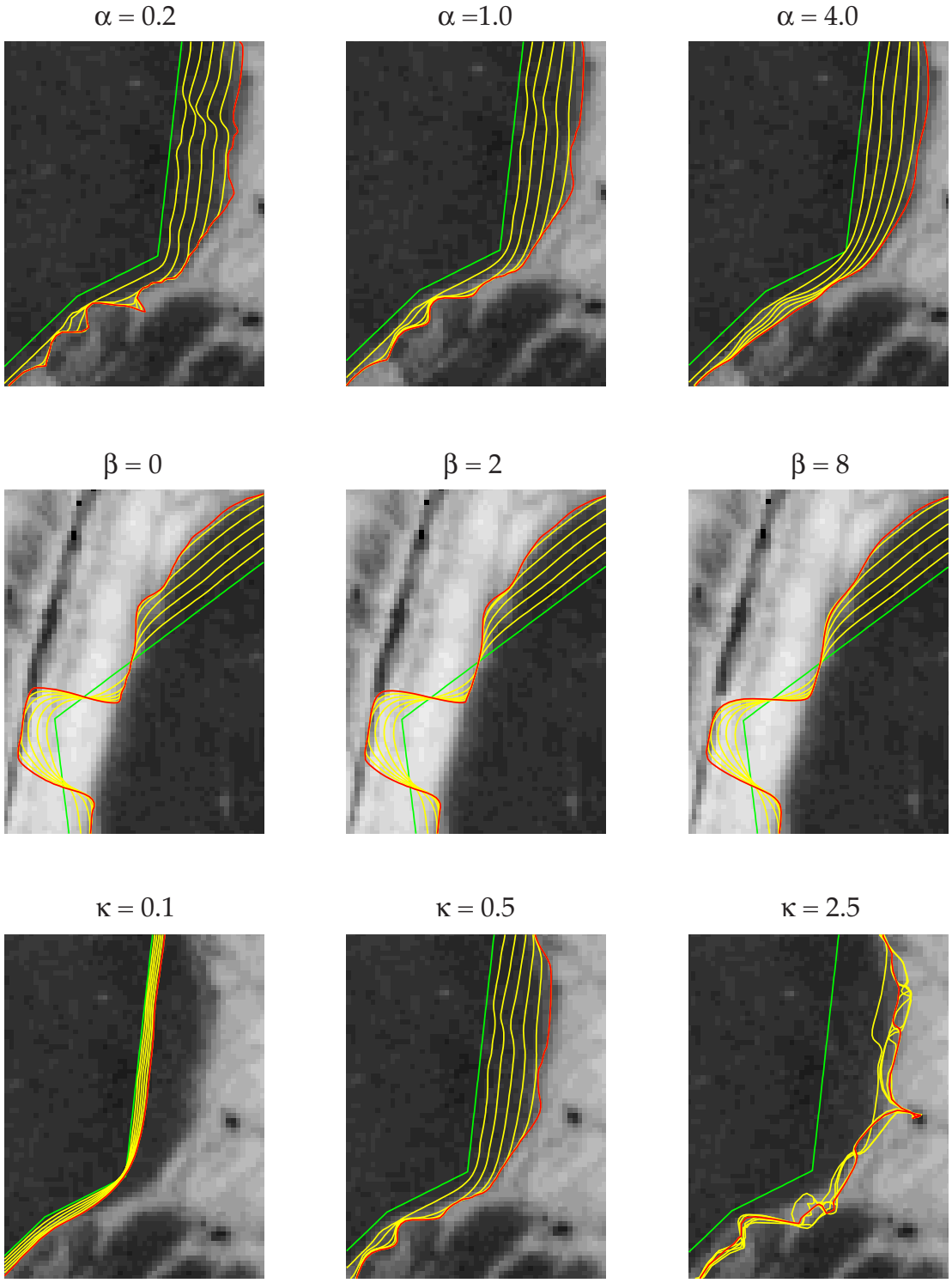


Figure IV–7: The effect the ACM parameters α, β, and κ is visually illustrated in these sequences. The green line shows the initial boundary segment, the red line shows the state of the ACM after 30 iterations, and the sequence of yellow lines shows the contour evolution at five iteration intervals.

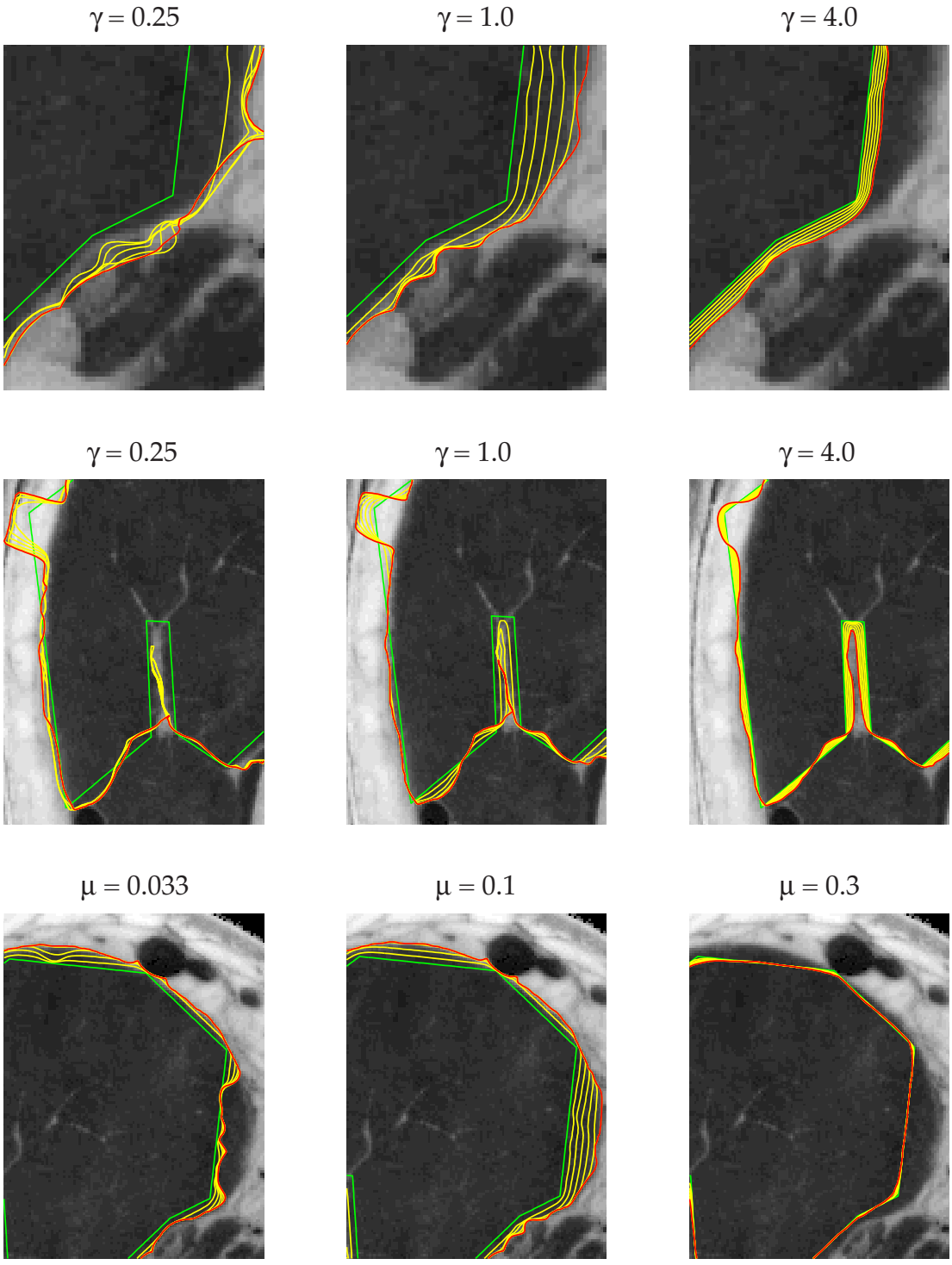


Figure IV–8: The effect the ACM parameters γ and μ is visually illustrated in these sequences. The green line shows the initial boundary segment, the red line shows the state of the ACM after 30 iterations, and the sequence of yellow lines shows the contour evolution at five iteration intervals.

which otherwise disappears over time. The initial choice for γ is 1.0, but this may be increased as needed to avoid the disappearance of long, thin structures.

μ : This parameter is referred to as controlling the pull of distant features on the parametric curve. In the bottom row of Figure IV–8, note that at a setting of 0.033, the curve evolves toward the boundary of the dark muscle mass only when the initial boundary is close to it. A value of 0.3, however, is too high, and the boundary can be seen to be collapsing inward over time. The starting choice for μ is 0.1, which may be reasonably decreased when the initial contour is close to the boundary of interest.

The default values presented here were arrived at after weeks of trial applications on the Visible Human imagery and initial hints by Xu and Prince [1997]. Figures IV–7 and IV–8 illustrate the effect of the parameters individually, with the others held at their default values. To some extent, the parameters control independent behaviors of the ACM, though there can be interaction effects. For instance, increasing the internal force κ while also increasing the viscosity γ may result in no effect, since the curve is being pushed harder but against a stronger restraint. For the final comparison, the parameters were varied for each structure as necessary for the best ACM performance. Section V.2.2 presents the values which ultimately were used for each structure.

Another key to successfully using an ACM is defining an appropriate edge map. In the edge map, small numbers will attract the parametric curve to the boundary and large numbers will repel it. This is problematic for second-derivative edge strength measures, such as the Laplacian. When the image is convolved with a Laplacian kernel, in the resulting edge map an edge is indicated by a zero crossing, which means there is a high value on one side of the crossing and a low value on the other side. If such an edge map

is used for an ACM model, the evolving curve will be both attracted and repulsed from the boundary, and this does not lead the system to any reasonable convergence.

After experimentation with a variety of edge strength operators, the edge map that performed reasonably well over the Visible Human images was generated in Matlab by:

(1) computing the Sobel operator in horizontal and vertical directions at each pixel and then calculating the root of the sum of these squared values;
(2) normalizing these values across the image to the range [0,1];
(3) logarithmically transforming the edge image to bring up the values of the weaker edges, which improves their influence on the parametric curve in the absence of strong edges.

Comments in source code provided by Demetri Terzopoulos [2002] indicate a similar line of thought in defining an appropriate edge strength operator.

The initial boundary contour was determined interactively by manually defining a "close-enough" boundary around the structure of interest with a series of mouse clicks.

IV.2.3 EXPERT'S TRACING ASSISTANT (ETA)

The background and operation of ETA have already been covered in Chapter III. For this study, a set of 23 features was used as input for the neural network. The features were defined by filters that were centered on the candidate pixel and its neighbors.

Figure IV–9 schematically illustrates the placement of the centers of the input filters. The figure illustrates a boundary-in-progress, indicated by the two shaded pixels and an arrow for the direction in which the boundary is being defined. One of the next pixels to be considered is **C,** and relative to the direction given by the segment in progress, three

left neighbors are labelled **L3**, **L2**, and **L1**, and three right neighbors **R1**, **R2**, and **R3**.

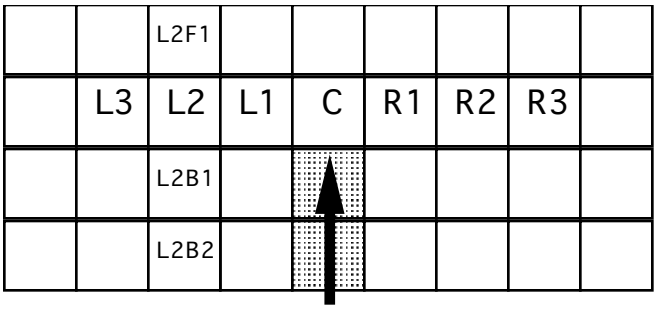


Figure IV–9: The arrow indicates a boundary in progress; a next possible pixel to consider adding to this boundary is the candidate labelled C, shown here with its neighbors labelled accordingly.

The 23 input features x_0 through x_{22} are defined below.

a) The function $g(p)$ takes the channel value G of the RGB value of pixel p and normalizes it to [0,1]. The first five inputs are:

$$x_0 = g(\mathbf{L2}); \quad x_1 = g(\mathbf{L1}); \quad x_2 = g(\mathbf{C}); \quad x_3 = g(\mathbf{R1}); \quad x_4 = g(\mathbf{R2}).$$

b) The function $\boldsymbol{G^3}$ is the 3x3 Gaussian filter with σ=1, where the coefficients have been normalized to sum to one. Since the neural network's output is only used at one significant digit of precision, no more than two significant digits need to be carried through the intermediate calculations. The coefficients beyond the 3x3 center are outside this range of significance, thus a 3x3 size is used rather than the more common 5x5 size for a σ=1 mask. The normalized green channel values g are multiplied by the matrix $\begin{pmatrix} .0751 & .1238 & .0751 \\ .1238 & .2042 & .1238 \\ .0751 & .1238 & .0751 \end{pmatrix}$.

Five 3x3 Gaussian inputs are:

$$x_5 = \boldsymbol{G^3}(\mathbf{L2}); \quad x_6 = \boldsymbol{G^3}(\mathbf{L1}); \quad x_7 = \boldsymbol{G^3}(\mathbf{C}); \quad x_8 = \boldsymbol{G^3}(\mathbf{R1}); \quad x_9 = \boldsymbol{G^3}(\mathbf{R2}).$$

c) The function $\mathbf{B^5}$ averages a 5x1 bar of pixels in a direction parallel to the direction established for the boundary in progress. For example, using the notation established in Figure IV–9:

$$\mathbf{B^5}(\mathbf{L2}) = (g(\mathbf{L2}) + g(\mathbf{L2B1}) + g(\mathbf{L2B2}) + g(\mathbf{L2F1}) + g(\mathbf{L2F2})) \ / \ 5.$$

This *5bar* input is used for five inputs:

$$x_{10} = B^5(\mathbf{L2}); \quad x_{11} = \boldsymbol{B^5}(\mathbf{L1}); \quad x_{12} = \boldsymbol{B^5}(\mathbf{C}); \quad x_{13} = \boldsymbol{B^5}(\mathbf{R1}); \quad x_{14} = \boldsymbol{B^5}(\mathbf{R2}).$$

d) The function $\boldsymbol{G^5}$ is the 5x5 Gaussian filter with σ=1.5, where the coefficients have been normalized to sum to one. The 5x5 size is used, with the same rationale as discussed for $\boldsymbol{G^3}$. The normalized green channel values g are multiplied by the

matrix $\begin{pmatrix} .01442 & .02808 & .03507 & .02808 & .01442 \\ .02808 & .05470 & .06831 & .05470 & .02808 \\ .03507 & .06831 & .08531 & .06831 & .03507 \\ .02808 & .05470 & .06831 & .05470 & .02808 \\ .01442 & .02808 & .03507 & .02808 & .01442 \end{pmatrix}$.

Four 5x5 Gaussian inputs are:

$$x_{15} = G^5(\mathbf{L3}); \quad x_{16} = G^5(\mathbf{L1}); \quad x_{17} = \boldsymbol{G^5}(\mathbf{R1}); \quad x_{18} = \boldsymbol{G^5}(\mathbf{R3}).$$

e) The function $\mathbf{B^7}$ is similar to the function $\mathbf{B^5}$, but it averages a 7x3 bar of inputs instead. Four inputs with this filter are:

$$x_{19} = B^5(\mathbf{L3}); \quad x_{20} = B^5(\mathbf{L1}); \quad x_{21} = \boldsymbol{B^5}(\mathbf{R1}); \quad x_{22} = \boldsymbol{B^5}(\mathbf{R3}).$$

In aggregate, these 23 inputs cover approximately an 11x7 block of pixels centered on **C**.

Since ETA needs to be initially trained before boundary tracing and IS and ACM do not, the expert's second manual trace was sampled to train the network. The performance of this system will then be compared against the ground truth of the expert's first manual trace, against which the IS and ACM boundaries will also be compared. Further empirical discussion of this issue is in Section V.2.4.

IV.2.4 THE EXPERT

Human experts will exhibit some variation when manually tracing a boundary, even with the most visually distinct of borders available. Brahmi, et al., [1999] note that boundaries drawn by experts may display substantial variation, especially in areas where contrast is poor. Karayiannis and Pai [1999] also note an inconsistency of ratings among experts in a complex segmentation problem. In a comparison of an automated system to the expert, the system should thus not be required to exactly replicate an expert's boundary to be successful, but the system-defined boundary should be expected to fall within the range of variation for either a specific user or across a group of experts.

For a ground truth in this comparison, an expert at Visible Productions was asked to manually trace the structures. The expert worked through the image set twice, generating two boundaries, on independent trials, for each structure. This permits a measure of the variation within the expert's manual tracings to be quantified.

It could be argued that this ground truth is not really a truth, but one user's subjective judgement of a structural boundary. The expert user, however, is bringing outside knowledge to bear on the problem, and is dealing with more than simple pixel values when delineating a boundary. For a system to be useful and acceptable as an assistant to an expert, it should replicate what the expert is attempting to do, rather than do

what is dictated by some set of *a priori* assumptions over which the expert has no input or control.

Visible Production's tracing system is very flexible in providing the user the ability to modify a boundary as it is in progress, thus it can be matched exactly to the expert's desires. The user can work at any enlargement level from 2:1 to 20:1; pixels are replicated rather than interpolated when displayed at magnification. In observing experts at their task, it was noted that after quickly scanning the image overall, they worked zoomed in on the image, typically at a factor of 10:1. Also, they had a preferred direction for tracing structures; in this case the structures were always traced clockwise.

IV.3 Measures of Comparison

Since these are all user-guided systems, one comparison is the amount of user interaction required to generate a structure's boundary. One measure considered is the amount of time a user spends defining a boundary. This however, is more a function of the user-interface engineering than it is of the underlying method itself: the best algorithm wrapped in an awkward interface would not fare well in such a comparison. Also, the user's interaction will be different depending on the system, so it will not be directly comparable across systems. However, the general quality and amount of interaction has been tracked and will be discussed.

Similarly, performance measured in actual execution time is not an adequate comparison. The methods in this study were all run on different platforms, and the software implementations varied from interpreted Matlab statements to optimized C code. Once a method is proved to be fundamentally superior, performance engineering can be brought to bear on it to improve efficiency. Computations can be sped up by

more and faster processors and better numerical methods; for example, the field of Bayesian methods is in renaissance now that the computational load of these methods can be handled. However, there are basic algorithmic bounds to note; for example, an iterative relaxation method such as ACM will always be computationally more expensive than a forward pass through a neural network.

Figure IV–10 displays an example of a machine traced boundary (in black) overlaying a manually traced boundary (in white) for a visual comparison. The boundary definitions that result from these methods need to be quantitatively compared, to each other, and to the ground truth of an expert's manual definition. Qualitatively, one could say these are quite close, but how can 'close' be quantified? The remainder of this section develops a methodology to quantify this notion of "closeness".

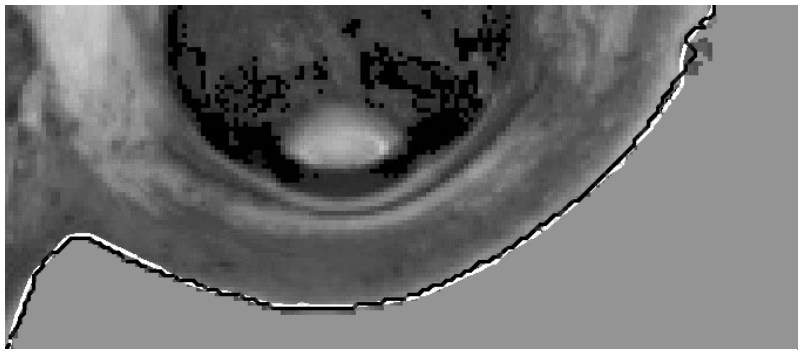


Figure IV–10: A visual comparison of an expert-traced (white) against a machine traced (black) boundary.

IV.3.1 Comparing Boundary Curves

The boundaries these methods produce are sets of ordered points which can be connected by lines or curves or splines to visually bound a region. The comparison then is to measure how far the boundary defined by one set of points is from the boundary defined by the other set of points.

One distance measure is to find, for each point in one set, the distance to the closest point in the other. This measure can be misleading, however. The left side of Figure IV–11 illustrates two sets of points marked with **x** and **o**, both sampled off the same circle. Since they come from the same curve, they should ideally have a distance measure of zero between them. However the minimal distance from any one point to the closest point of other set is 2.61. A better method then "nearest-point" for quantifying this difference would be to measure the perpendicular distance from a point to the polyline created by connecting the points in the other set. This is shown in the right half of Figure IV–11, and the distance measured this way is now 0.34. Though not zero, it is an order of magnitude closer to the desired answer.

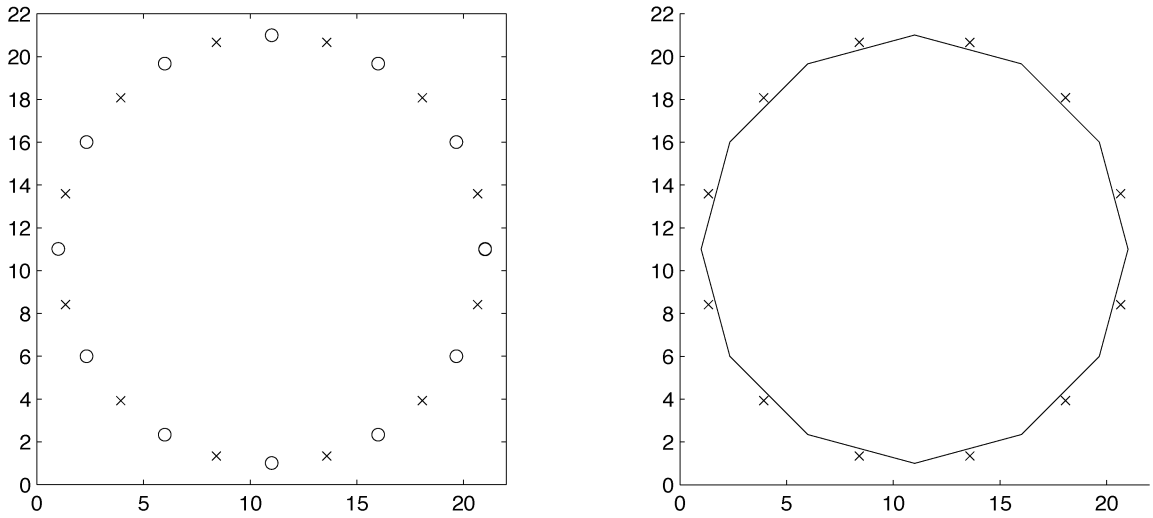


Figure IV–11: LEFT: Two sets of points, marked with **x** and **o**, taken from a circle of radius 10 centered at (11,11). RIGHT: Measuring from one set of points to the polyline of the other set.

More accurate measures could be derived by fitting a higher-order curve to the set of points rather than a piece-wise linear, however in this study such a higher degree of precision is not necessary. The methods studied are set to produce boundary points spaced between one and two pixels apart, thus a worst case misstatement of error in measuring a distance between sets would be half that amount, or less than one pixel.

Reducing that worst error by an order of magnitude to a few tenths of a pixel by measuring to the polyline is sufficient for the comparisons performed herein.

The measure of one curve to another is not simply one number. For each point defining one curve, there is a number that quantifies its distance to the other curve; all these numbers in aggregate comprise a *distance set*. The quantitative comparison of curves thus involves generating and studying these distance sets.

IV.3.1.1 Generating a Distance Set

For a set of points $\mathcal{P}$, what is the distribution of distances of the points in $\mathcal{P}$ from a curve (polyline) defined by another set of points $\mathcal{C}$?

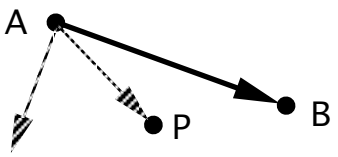


Figure IV–12: Measuring the perpendicular distance from point P to line AB.

The first step is to measure from one point to a line segment of the polyline. As shown in Figure IV–12, for the point P, the distance of P to $\overline{\mathrm{AB}}$ is the projection of $\overline{\mathrm{AP}}$ onto $\overline{\mathrm{AB}}^{\perp}$ *(the perpendicular to segment AB)*, which is $\overline{\mathrm{AP}} \bullet \left(\overline{\mathrm{AB}}^{\perp}\right)$.

To find the closest segments of the polyline to the point, for each point P in $\mathcal{P}$, start by finding the closest point C in $\mathcal{C}$. The minimum distance from P to $\mathcal{C}$ can be found by taking the minimum of the distances of P to the segments on either side of C.

It is not sufficient, though, only to project $\overline{\mathrm{CP}}$ onto the two segments which share C and pick the minimum. A problematic situation is shown in Figure IV–13. In the case illustrated, the minimum distance from P to the polyline is from P to $\overline{\mathrm{CC}^{+}}$, however

the projection of P onto $\overline{CC^-}$ is smaller. This is resolved by noting that since $\overline{CC^-}$ is a directed vector, the projection of $\overline{CP}$ onto $\overline{CC^-}$ is negative in this case. So the distance from P to the polyline created by $\boldsymbol{C}$ is the minimum of:

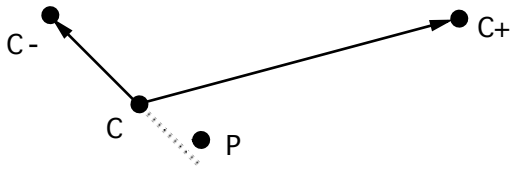


Figure IV–13: Point P is closer to the segment CC+ rather than to CC-.

(1): $\overline{CP} \bullet \left(\overline{CC^-}^{\perp}\right)$, considered only when $\overline{CP} \cdot CC^- > 0$, or

(2): $\overline{CP} \bullet \left(\overline{CC^+}^{\perp}\right)$, considered only when $\overline{CP} \cdot CC^+ > 0$, or

(3): $\| \overline{CP} \|$, the distance from P to C.

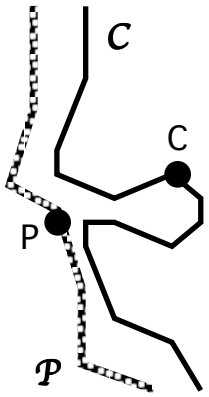


Figure IV–14: The distance from $\boldsymbol{\mathcal{P}}$ to $\boldsymbol{C}$ is not the same as from $\boldsymbol{C}$ to $\boldsymbol{\mathcal{P}}$.

The set of distances for all P gives a collective measure of the distance from $\boldsymbol{\mathcal{P}}$ to $\boldsymbol{C}$. However, the set of distances from $\boldsymbol{\mathcal{P}}$ to $\boldsymbol{C}$ is not the same as the set of distances from $\boldsymbol{C}$ to $\boldsymbol{\mathcal{P}}$. In Figure IV–14, the minimum distance from P to the curve $\boldsymbol{C}$ does not reflect the excursion of the curve out and around point C. The discrepancy of the two curves is only captured by the distance from C to $\boldsymbol{\mathcal{P}}$. Therefore, to adequately quantify the distribution of distances between the curves, the union of the set of distances from $\boldsymbol{\mathcal{P}}$ to $\boldsymbol{C}$ and from $\boldsymbol{C}$ to $\boldsymbol{\mathcal{P}}$ must be studied.

The distance set is thus the ordered collection of distances from all P in $\mathcal{P}$ to $\mathcal{C}$ and from all C in $\mathcal{C}$ to $\mathcal{P}$.

IV.3.1.2 Comparing Distance Sets

Figure IV–15 illustrates several visualizations that characterize this set of polyline-to-polyline distances. The first graph in the upper half of the figure plots distances between the curves (on the vertical axis) as a function of position on the curve (on the horizontal axis). These are called *"position graphs"* in this work. Sections of the curve where there are significant excursions above a threshold of 1.0 are indicated.

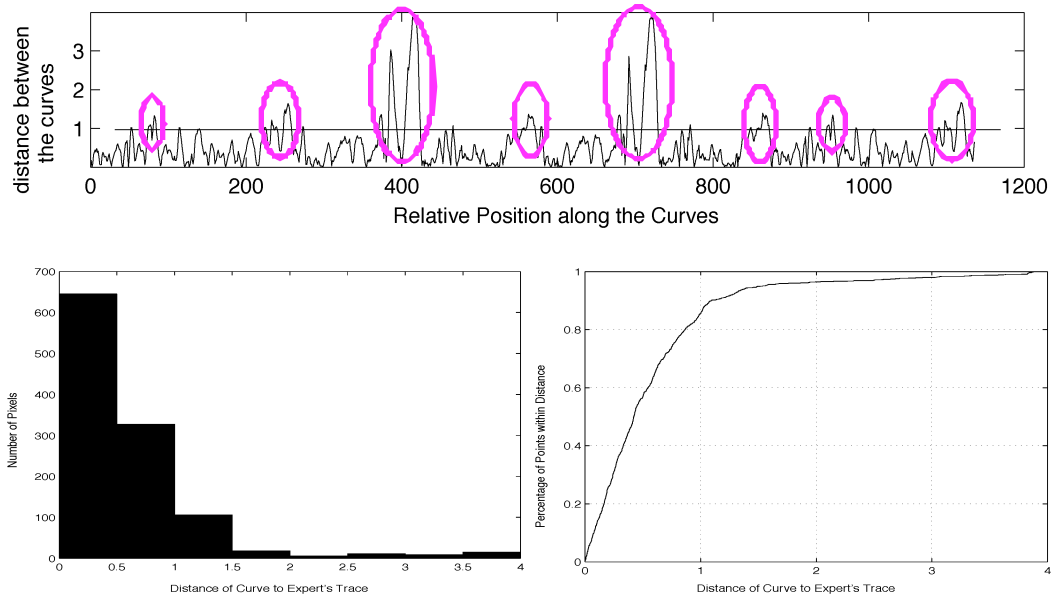


Figure IV–15: The set of distances can be visualized by examining (at top) the distribution of distances between the curves as a function of position on the curve, (lower left) a histogram of the distances overall, and (lower right) the empirical cumulative distribution function (CDF) of the distances. Significant excursions above 1.0 are circled in the upper graphic.

Care must be taken in what statistics are used to quantify the comparison of boundary curves in this study, since depending on one's purpose, some measures of the difference will be misleading,. For example, the Hausdorff measure of the difference between the

two curves is the maximum value of the position graph (Huttenlocher, et al. [1993]). In this case, the Hausdorff distance between the curves is 3.9. In the context of these user-guided systems where the user will override distances of over 1.0, the Hausdorff measure becomes irrelevant, though, since all deviations over 1.0 are equally severe, i.e., they require a user correction regardless of their magnitude. The measure of these systems' success is based on how close they are for the bulk of the pixels, not how far off the single worst difference is.

In this position graph, eight significant excursions above a threshold of 1.0 are highlighted. These eight excursions represent four sections where the two curves are significantly more than 1.0 pixels apart from each other; the number of excursions shown in the position graph is twice number of troublesome curve sections, since this set is a union of the distances measured from the first to the second and then from the second to the first. If the goal was to have the two curves within one pixel of each other, this indicates that there are four places where operator intervention would be required to adjust the curves so as to meet that objective.

The lower left of Figure IV–15 is a simple histogram of the distance set, with the number of occurrences on the vertical axis and distance bins on the horizontal. The lower right of Figure IV-15 is an empirical cumulative distribution function (CDF) over the distance set. The vertical axis measures the fraction of occurrences that are within a tolerance specified on the horizontal axis. The CDF allows quantification of the inter-curve distances by relating a tolerance in pixels to a percentage within that tolerance.

In the next chapter, the boundary methods delineated by the methods in this chapter will be compared on a set of structures in a controlled study. These comparison statistics will quantify the results.

——== Chapter V ==——

Comparative Results

This chapter presents the comparison of the boundary definitions provided by an expert to the semi-automated ACM (Active Contour Model), IS (Intelligent Scissors), and ETA (Expert Tracing Assistant) methods on a set of representative structures from the Visible Human imagery. The discussion categorizes the boundaries as "basic" (where the boundary is represented by well defined image edges), "hard" (where outside knowledge is required to define a boundary), or "intermediate" (somewhere between those two extremes). Additional structural boundaries are explored in MR imagery to illustrate similar results in another imagery domain.

Section V.1 compares the boundaries defined by these three methods. Section V.2 extends the general discussion of IS, ACM, and ETA methods in Chapter IV to the very specific details of how the methods were configured to define adequate boundaries for the specific structures of study. Section V.3 discusses the degree of user interaction required. Section V.4 discusses the reproducibility of the results. Section V.5 extends the imagery into the X-ray CT domain, further exploring issues identified in the Visible Human imagery. And in closing, Section V.6 summarizes the issues identified in this comparison study.

V.1. Comparison of the Boundary Definitions

The measures previously discussed in Section IV.3.1 are presented here for the nine structures identified in Section IV.1. There are five boundaries under consideration for each structure: two boundaries that were manually traced by the expert and one boundary produced by each of the three user-guided methods. Figure V–1, Figure V–2, and Figure V–3 show the five boundaries superimposed on the structures identified in Figures IV–3 through IV–5. The images are presented in greyscale to better observe the behavior of the boundaries shown in color. The color coding across the figures of this chapter will maintain this common meaning:

Yellow: the expert's manually generated ground-truth boundary (GT).

Green: the boundary defined by the Expert's Tracing Assistant (ETA)

Blue: the boundary defined by the Active Contour Model (ACM)

Red: the boundary defined by the Intelligent Scissors (IS)

Black: the second manually traced boundary provided by the expert (M2T).

The expert manually outlined each structure on two independent trials. The first boundary is used as the reference ground truth, and the second boundary is used to provide a measure of the inherent variation a user shows in manually tracing a boundary. A good boundary delineation method need not exactly match any specific expert's boundary definition, but it should be within the range of the expert's variance.

A value of one pixel was selected in advance as a basic criteria for two curves to be adequately "close enough". Values much larger than one pixel will yield less discrimination among the methods, since in some cases all the boundary curves are within two pixels of one another. Values much smaller than one pixel rely on

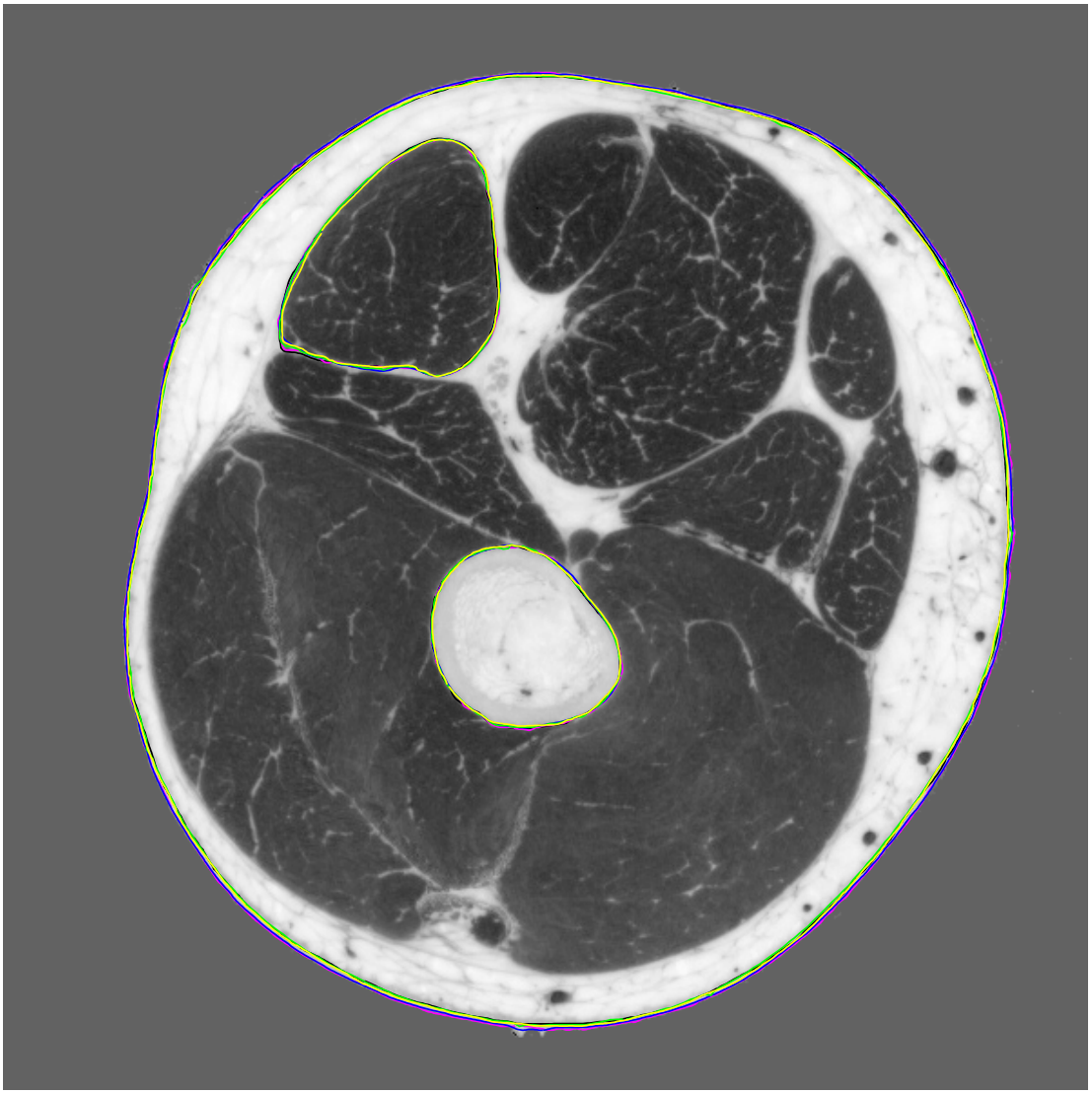


Figure V–1: The boundaries defined on the leg image; skin, muscle, and bone boundaries superimposed on a greyscale image of Figure IV–3. Key: GT is yellow, M2T is black, ETA is green, ACM is Blue, and IS is red.

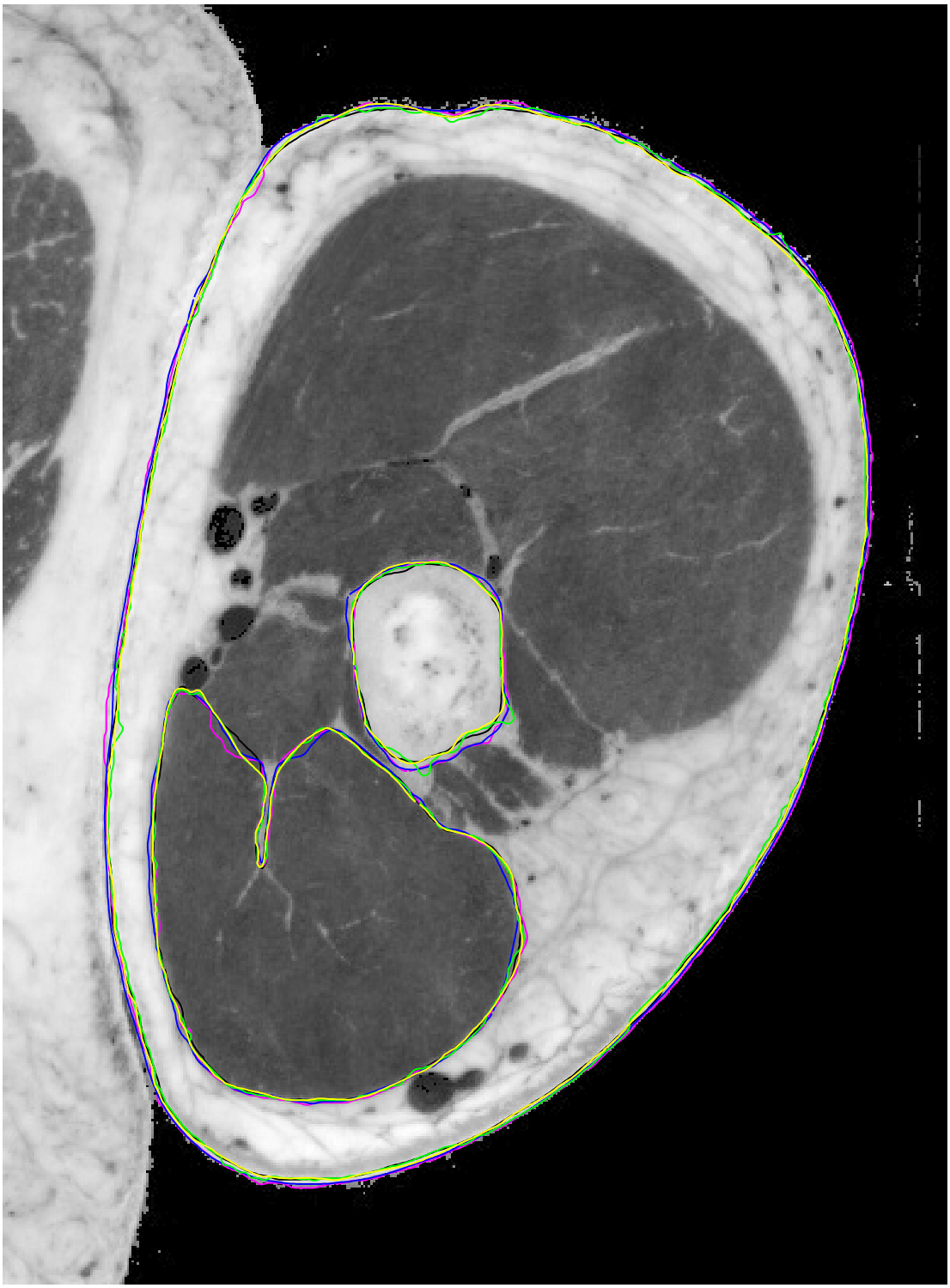


Figure V–2: The boundaries defined on the arm image; skin, muscle, and bone boundaries superimposed on a greyscale image of Figure IV–4. Key: GT is yellow, M2T is black, ETA is green, ACM is Blue, and IS is red

Figure V–3: The boundaries defined on the thorax image; throat, ventricle, and lung's lobe boundaries superimposed on a greyscale image of Figure IV–5. Key: GT is yellow, M2T is black, ETA is green, ACM is Blue, and IS is red.

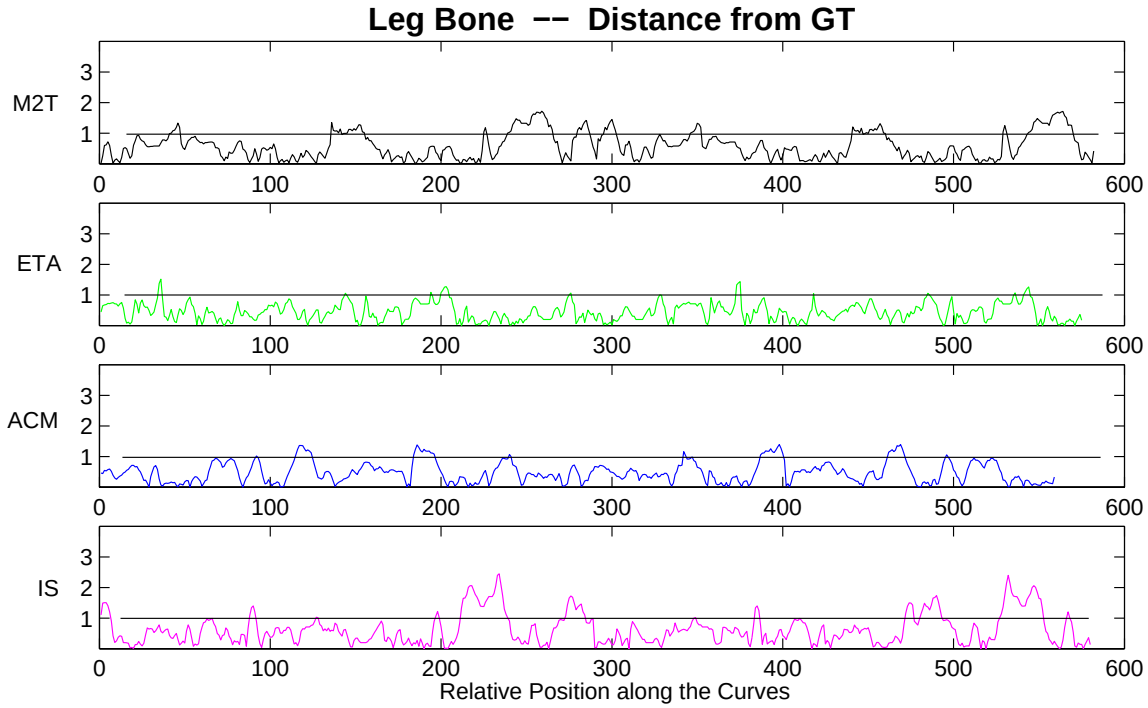
interpolations to sub-pixel accuracy, which may or may not be warranted by the image data. And as a practical matter, as the following empirical results show, the relationships among the methods remain consistent over a range around one pixel. Thus while numeric results would differ were a different threshold chosen, the relative rankings would remain the same.

V.1.1. QUANTIFYING THE COMPARISON

As an example, the graphs used to quantify the comparison for each structure are presented in Figure V–4. The background and justification of the graphics was presented and discussed in Section IV.3.1, the purpose of this section it to walk through the comparison graphics to verify their meaning.

The five graphs of Figure V–4 capture the differences between the ground truth GT and the other four boundaries defined around the leg bone. The top four horizontal graphs are four *position graphs*, plotting the distance of points along a curve to and from GT. The first position graph, labelled M2T, plots the distances from the expert's second manual boundary tracing to and from GT. The second position graph, labelled ETA, plots the distances of ETA's boundary definition to and from GT (ETA was trained on M2T and then compared to GT). The third position graph presents the difference data for ACM, and the fourth position graph presents the difference data for IS. The bottom graph in the figure collectively displays the empirical cumulative distribution functions (CDFs) over these four difference sets, each distinguished by color.

To verify the interpretation of the position graphs, the IS position graph is dissected here and related back to the two source boundaries it encapsulates. The first half (horizontally) of the position graph shows the IS-to-GT distances, followed by the GT-to-IS distances in the second half. Reading the first half of the IS position graph, the



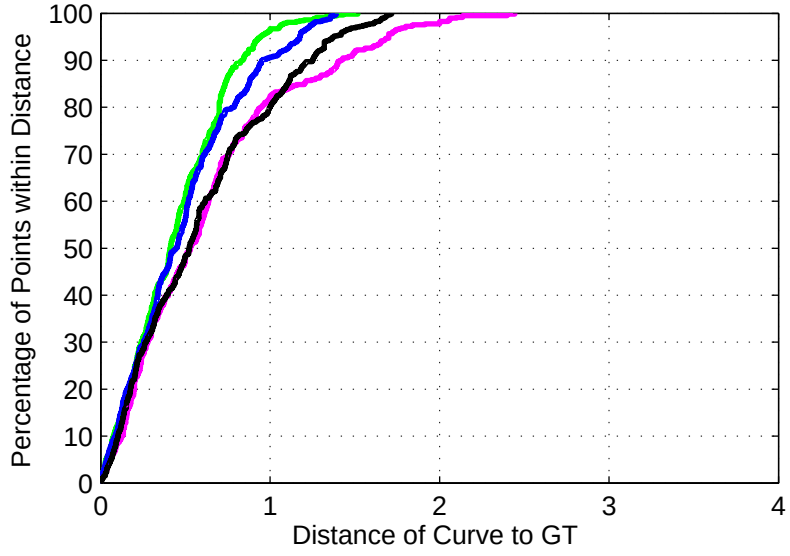


Figure V–4: For the Leg Bone, the top four position graphs measure the distance from the M2T, ETA, ACM, and IS boundaries to the reference GT; the bottom graph displays the CDF of the for differences sets.

horizontal axis records the position **s** along the IS curve for its length of 291 points, with the vertical axis showing the distance to GT. There is an initial excursion above 1.0 immediately starting at ***s=0***, another short excursion at ***s=88***, followed by a long excursion above 1.0 beginning at ***s=212***, and a final excursion beginning at ***s=273***. The second half of the graph starting at ***s=290*** has the distances from GT to IS, that reflects similar, though not exactly the same, excursions at different horizontal placements since GT starts at a different place than IS.

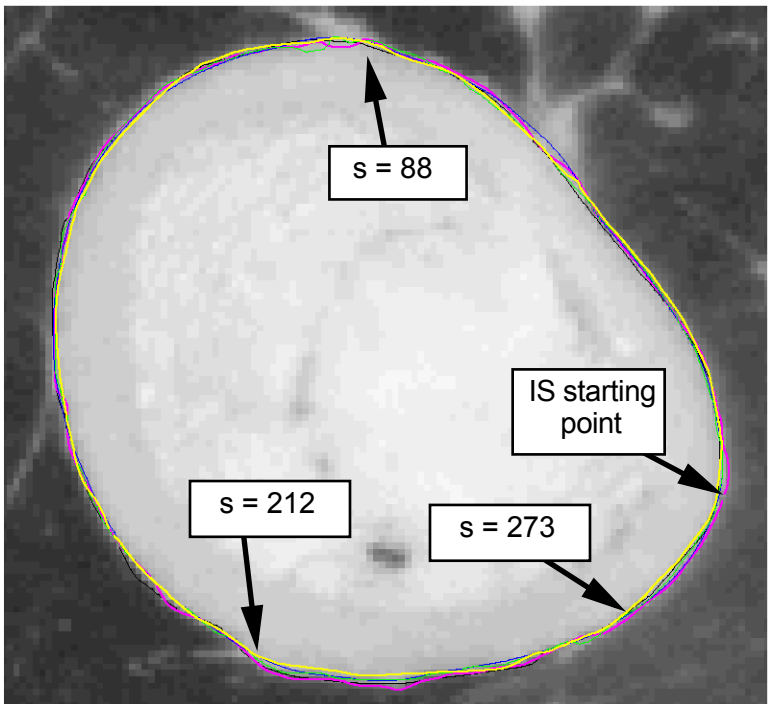


Figure V–5: The Intelligent Scissors (IS) boundary, in red, is annotated where it deviates significantly from the Ground Truth (GT) boundary, in yellow. These deviations can be mapped to the IS position graph of Figure V–4.

Figure V–5 shows in detail the leg bone boundaries super-imposed on the image. The IS boundary, in red, is annotated where it deviates significantly from the GT boundary, in yellow, to visually confirm the behavior seen in the position graph.

The gap in the red boundary is the IS boundary starting point, and this boundary was created counter-clockwise. In this case, the user clicked to start the IS contour one pixel away from the GT reference, and the initial few pixels of the boundary show up as the initial blip to a value of 1.5 at the start

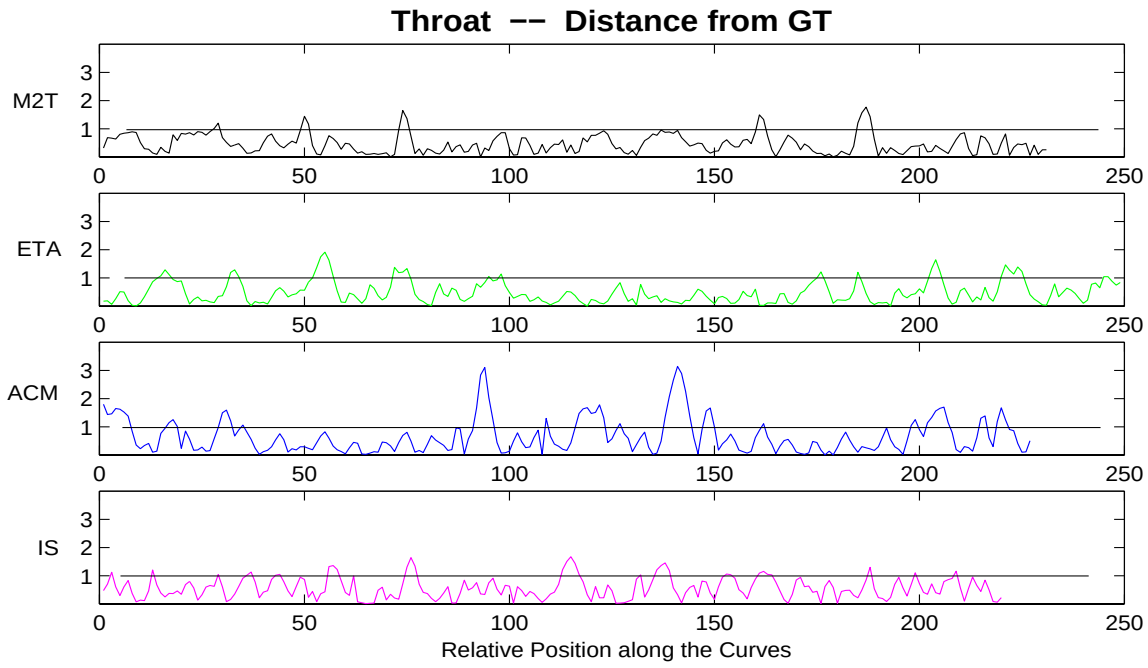
of the IS position graph. The excursions beginning at ***s=88***, at ***s=212***, and at ***s=273***, can be observed as the red IS curve pulls away from the yellow GT curve at those places.

V.1.2. Basic Cases - All Methods Essentially Agree

The boundaries for the leg bone were presented in Figure V–4. The overall agreement among all four boundaries is strong, since image edges adequately represent the boundary. There are a few excursions above the threshold line at 1.0 in the four position graphs, though none of the excursions pass 2.0. The differences arise from the fine points of how the connective tissue, which is of a similar visual characteristic to the bone, is segmented away from the bone. From the CDF graph, all the boundaries, excepting the upper tail of IS, are within the range of intra-expert variation represented by the black CDF, thus, all three could be thus judged to be as good as the expert.

The throat, leg muscle, and ventricle show similar agreement of boundaries to within the range of the intra-expert variability. The position graphs and CDFs for the throat, leg muscle, and ventricle, are shown, respectively, in Figures V–6, V–7, and V–9. The boundaries are in agreement for most of their lengths; they would each require minor tweaks in only a few short spots to bring them totally within one-pixel of the expert. However since their overall variation is within that of the expert, they may be good enough to stand without correction. The expert is typically self-consistent, however Figure V–8 shows detail of the leg muscle where this is not the case. The large diversion of the expert is evident in the lower-left where the yellow and black curves diverge as the expert chose to include slightly different pieces in the marbled area of muscle mass in different trials.

The large diversion in the IS position graph of Figure V–9 is evident in Figure V–10 in the lower right, where user-supplied points include an area of the ventricle which the expert



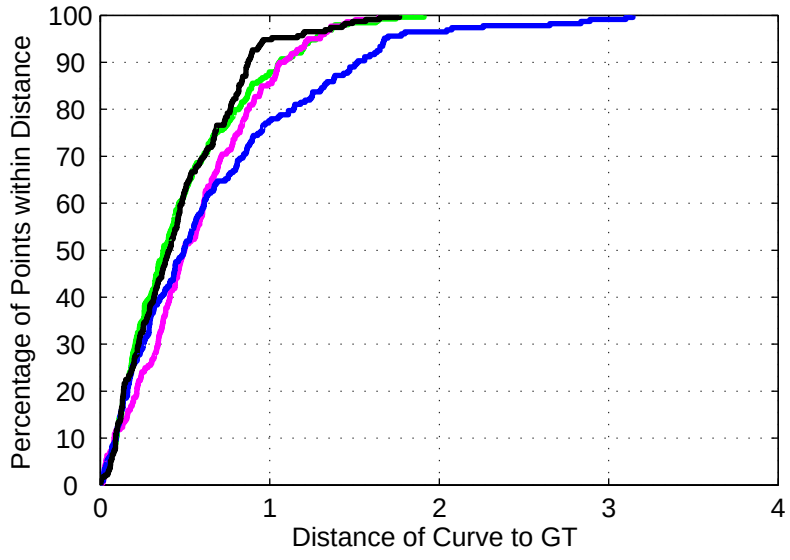
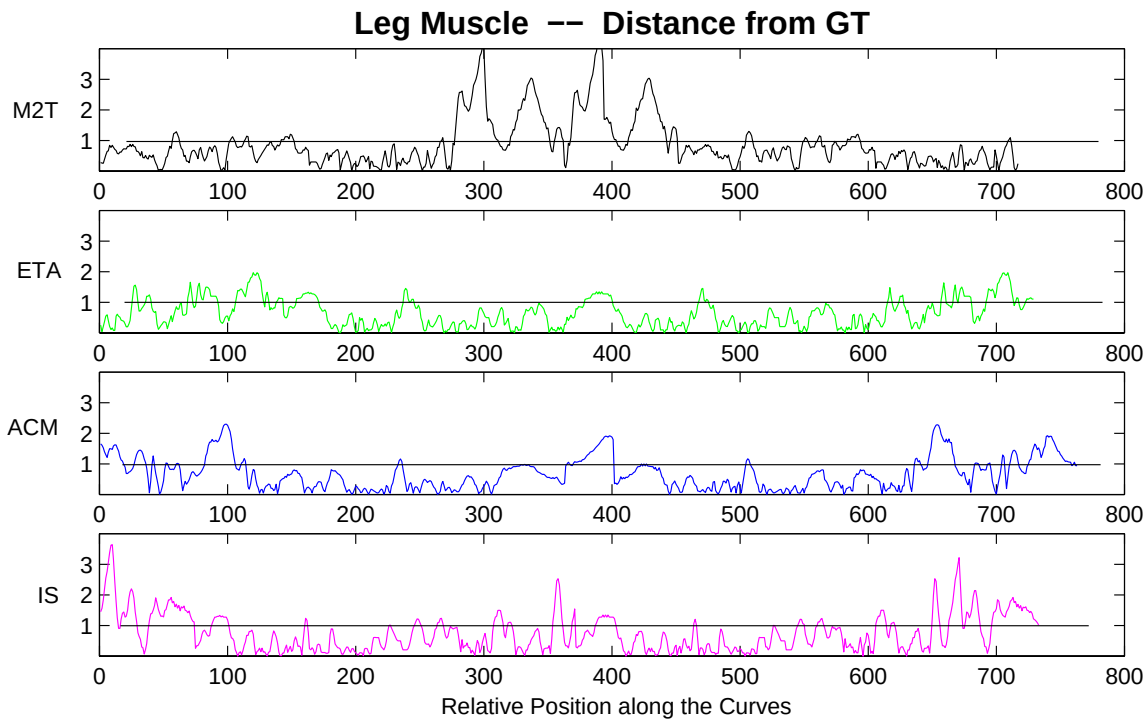


Figure V–6: For the Throat, the top four position graphs measure the distance from the M2T, ETA, ACM, and IS boundaries to the reference GT; the bottom graph displays the CDF of the for differences sets.



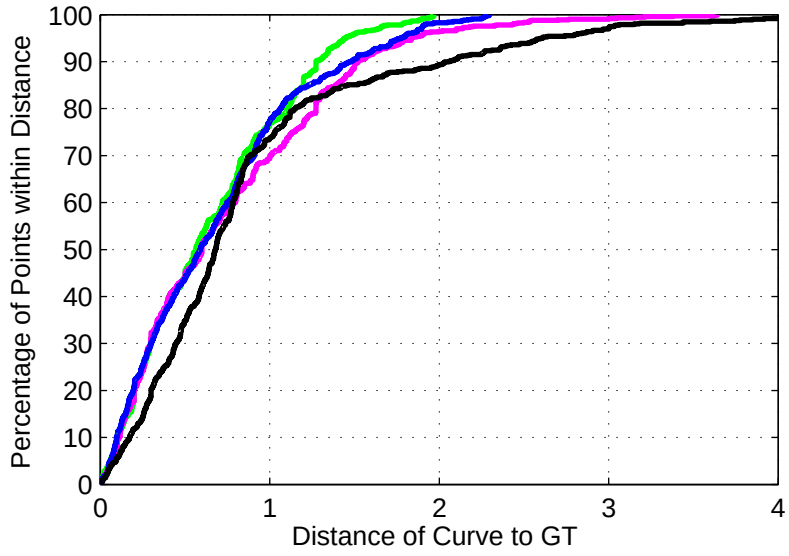
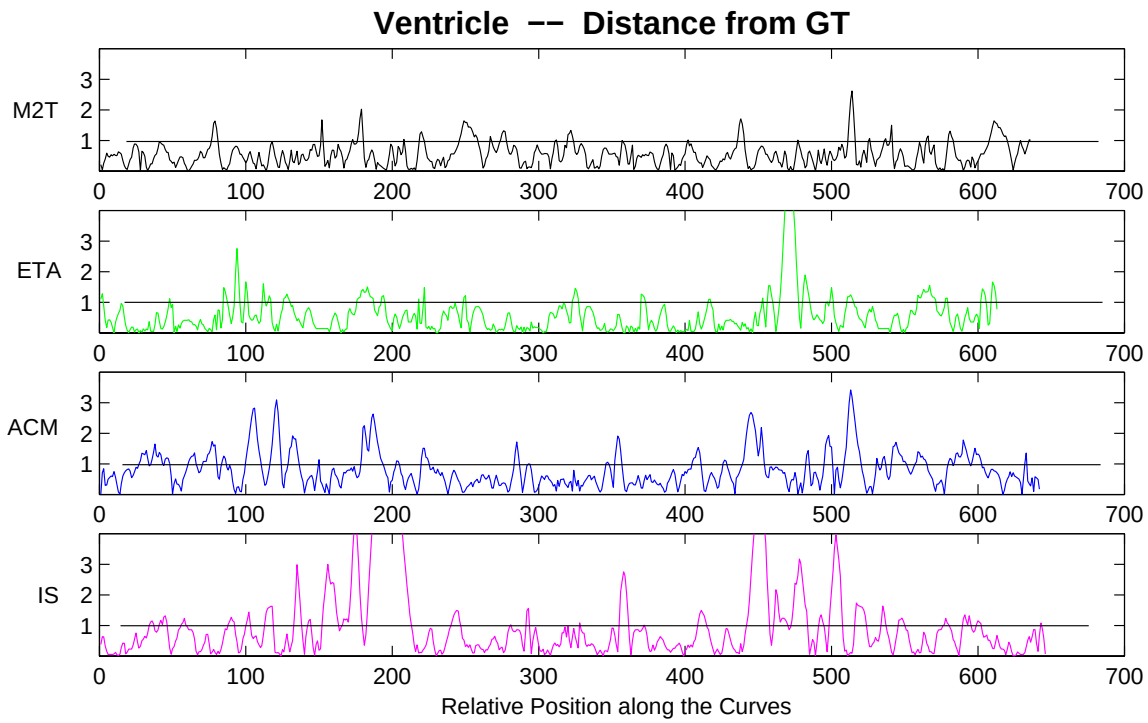


Figure V–7: For the Leg Muscle, the top four position graphs measure the distance from the M2T, ETA, ACM, and IS boundaries to the reference GT; the bottom graph displays the CDF of the for differences sets.

Figure V–8: The five boundaries drawn over the leg muscle; note the difference in the lower left between the expert's two manualy defined contours (yellow and black).



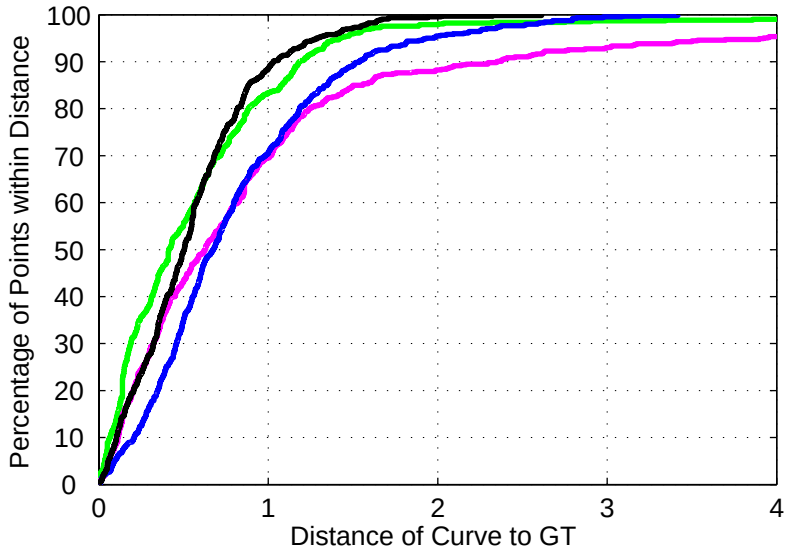


Figure V–9: For the Heart's Ventricle, the top four position graphs measure the distance from the M2T, ETA, ACM, and IS boundaries to the reference GT; the bottom graph displays the CDF of the for differences sets.

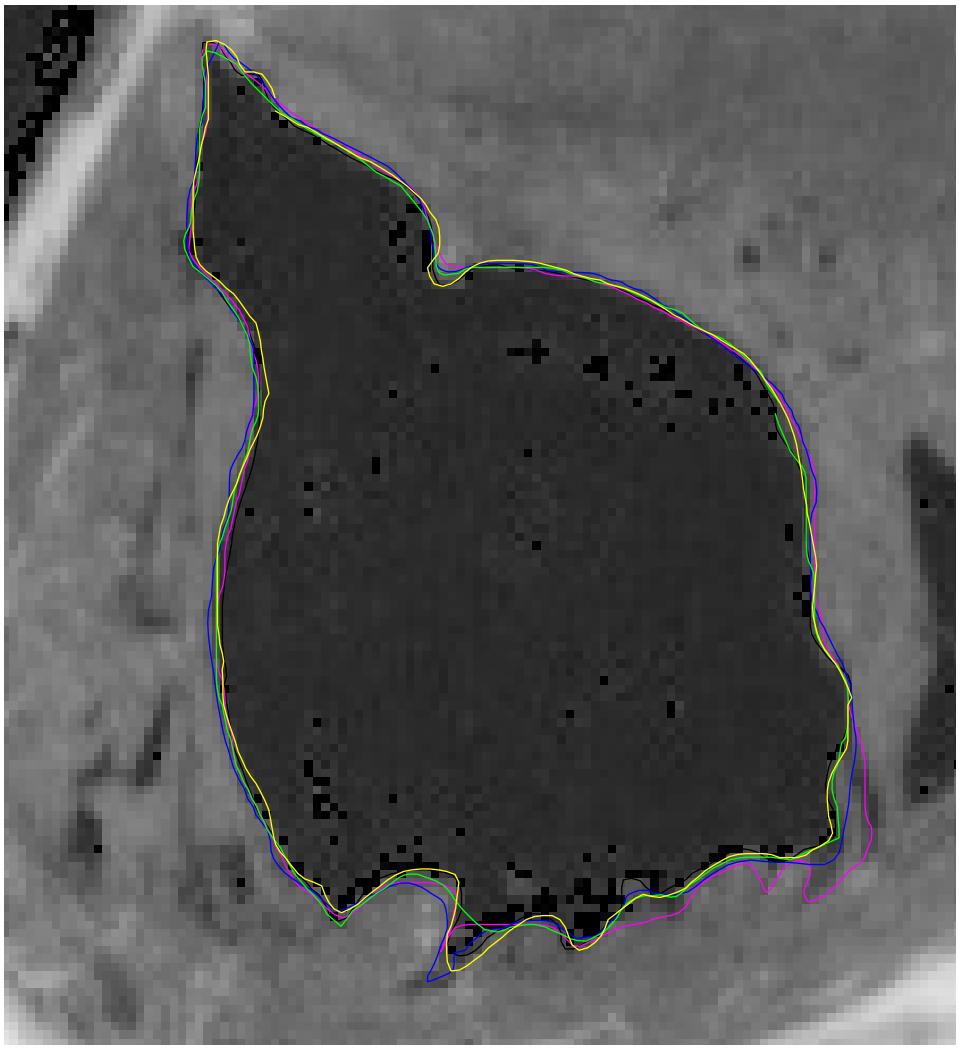


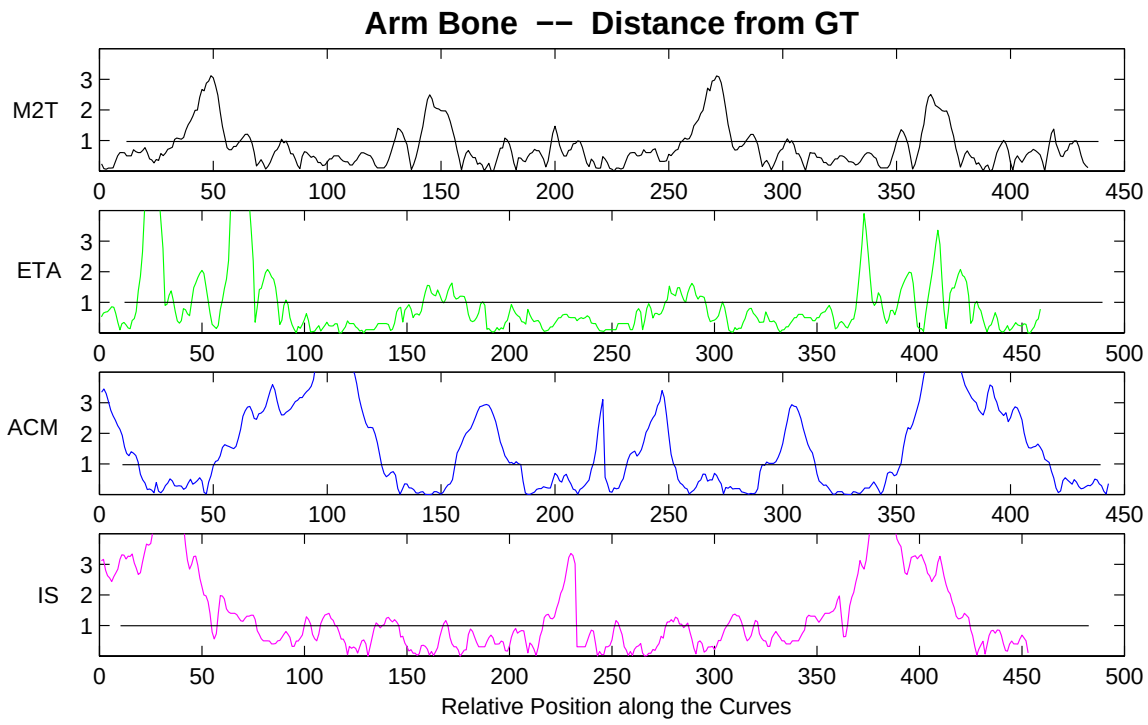
Figure V–10: The IS boundary (in red) diverges from the other boundary definitions in the lower right due to mis-placed user control points.

chose not to include. This diversion, which shows up strongly at *s=200* in the IS position graph (in Figure V–9), still implies only one user correction of about 30 pixels. The point to re-emphasize in this case is that it doesn't matter whether the boundary is wrong by three or thirty pixels, for when it's wrong it's still counted as just one user correction. It could even be argued that it's better to be off by more rather than less when wrong, since the user intervention required is more obvious when the error is extreme.

V.1.3. Hard Cases - All Methods Have Trouble

The statistics shown in Figure V–11, Figure V–12, and especially Figure V–13 show evidence of larger and sustained inconsistencies in the boundaries. These cases indicate significant trouble spots for the methods.

The arm bone and the arm muscle (see the boundaries on Figure V–2 or the raw image in Figure IV–4) have difficult boundary distinctions. The general problem with bone is that the connective tissue attaching muscle to bone is visually very hard to distinguish from the bone itself. In the best images, the fine texture of the outer bone surface may be slightly visible as a distinguishing surface. In the lower part of the arm bone, two patches of connective tissue are evident. While the expert has traced this bone-tissue boundary, both the IS and ACM methods are attracted to the much stronger tissue-muscle boundary. The large excursions from GT evident in the position graphs of Figure V–11 result from this problem. The ETA begins to follow this false boundary while tracing, but once corrected by the user and put back on the bone-tissue boundary, it continued closer to the expert. The CDFs in these three figures show ETA tracking the expert's variability, while both IS and ACM performed more poorly.



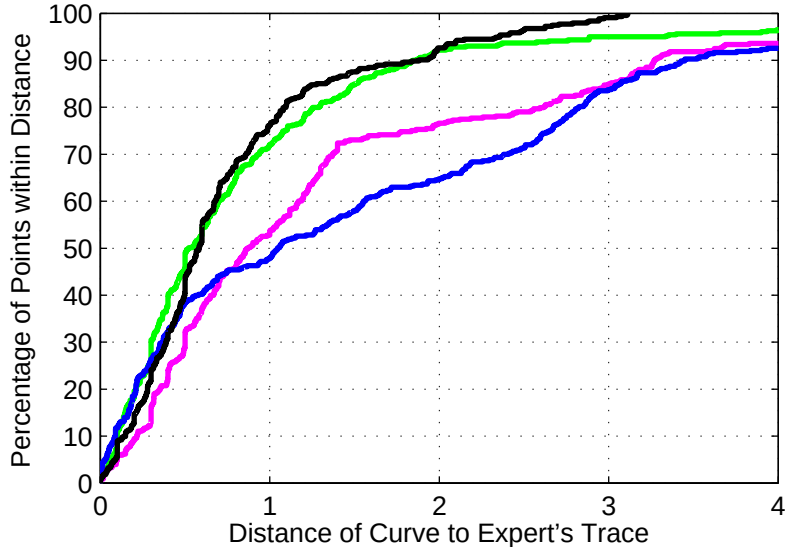
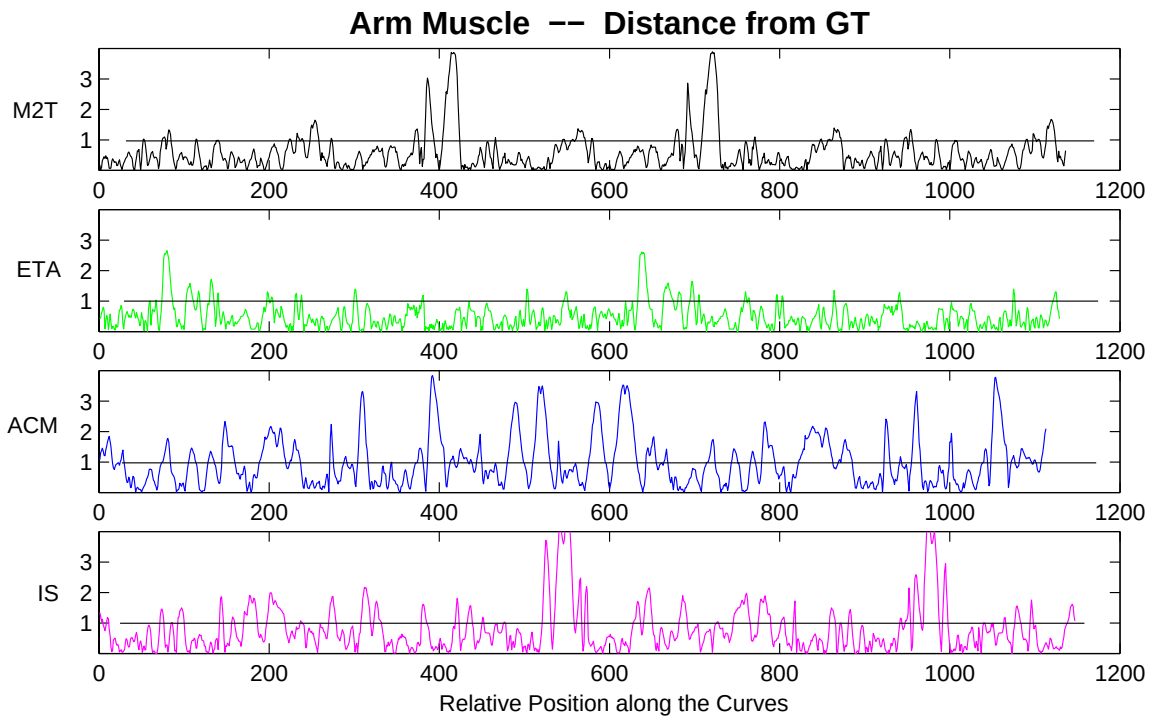


Figure V–11: For the Arm Bone, the top four position graphs measure the distance from the M2T, ETA, ACM, and IS boundaries to the reference GT; the bottom graph displays the CDF of the for differences sets.



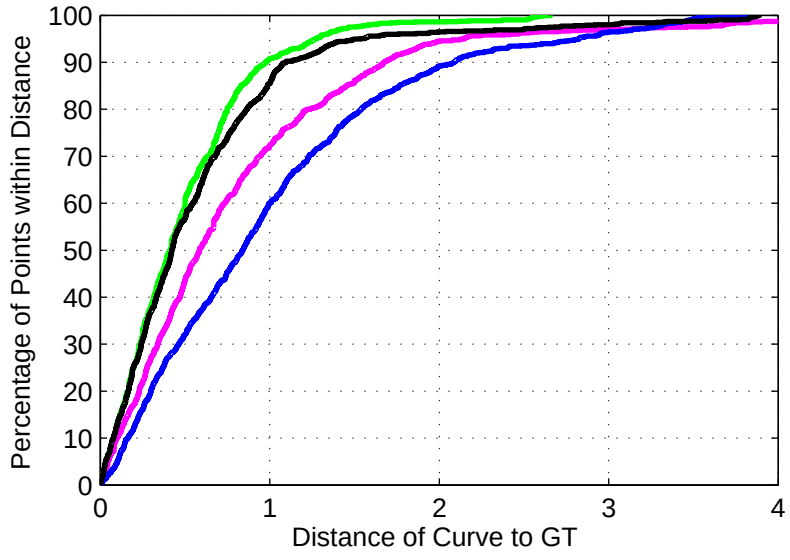
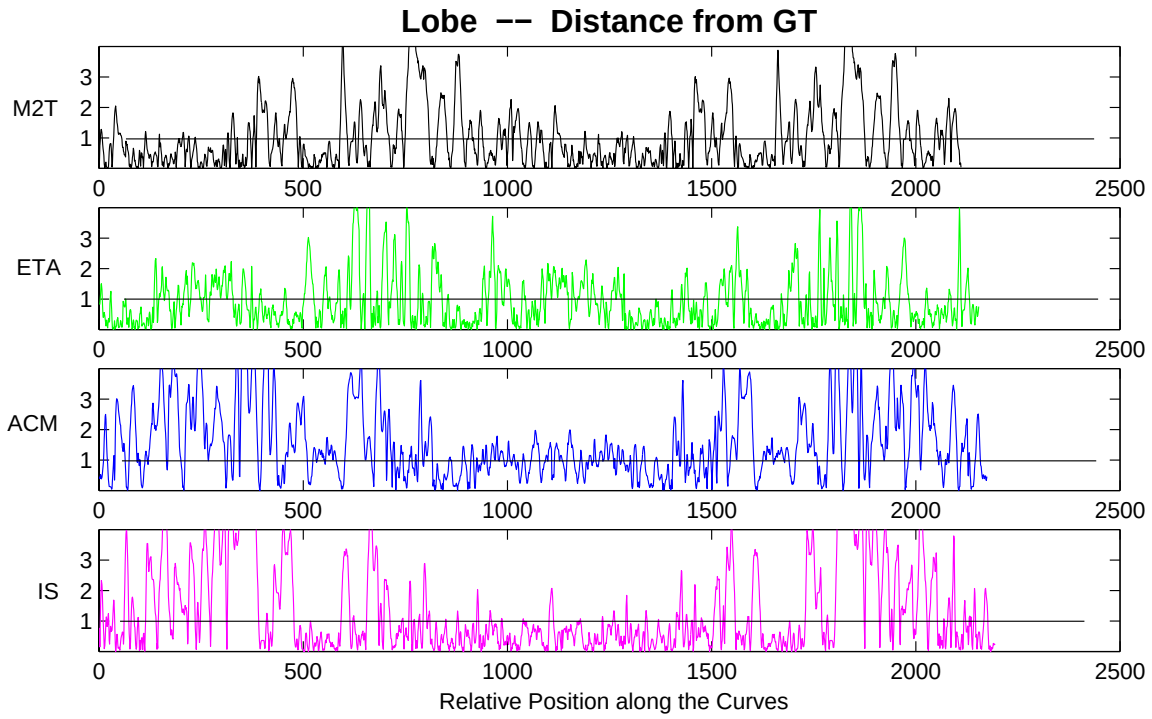


Figure V–12: For the Arm Muscle, the top four position graphs measure the distance from the M2T, ETA, ACM, and IS boundaries to the reference GT; the bottom graph displays the CDF of the for differences sets.



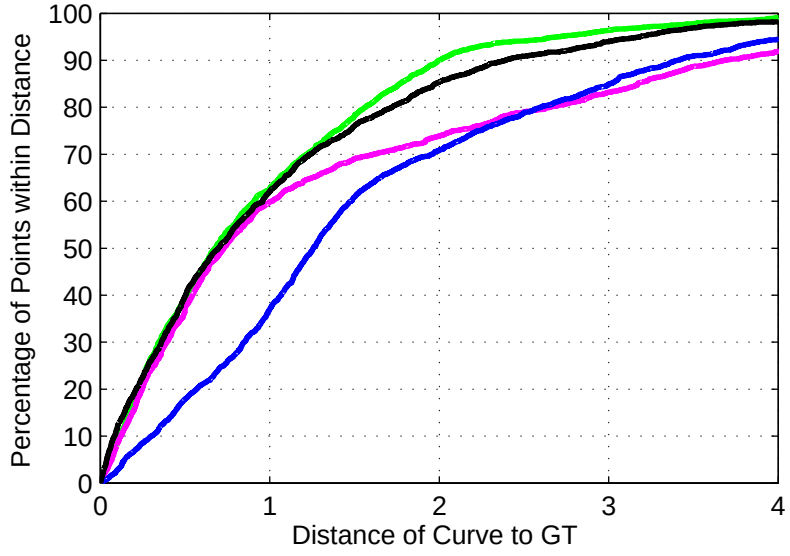


Figure V–13: For the Lung's Lobe, the top four position graphs measure the distance from the M2T, ETA, ACM, and IS boundaries to the reference GT; the bottom graph displays the CDF of the for differences sets.

The arm's muscle tissue presents several problems: first is a totally indistinct border along the top-left of the muscle; second is a deep fissure particularly troublesome to ACM; and third is the narrow channel between muscle and bone where the strong bone edges heavily influence edge-weighted ACM and IS methods. All five boundaries wander on somewhat different paths in the vaguely defined top-left region. The texture along the edge of the muscle seems to result in poorer performance of ACM and IS, as evident on the position graphs of Figure V–12.

The lobe of the lung (Figure V–13) is the worst-case scenario where all the boundaries radically differ; details are shown in Figure V–14. On the inner concave boundary, where the lung abuts the heart tissues, all methods are in general agreement. However, on the outer surface (the lower-right side of the image) where lung meets protective tissue meets ribs, there is no consistency, even between the expert's manual traces.

V.1.4. Intermediate Cases - Improved Learned Boundaries

ETA produces dramatically better results than IS and ACM in following the leg skin and the arm skin boundaries. The position graphs of Figure V–15 show ETA well within the one-pixel tolerance band, while IS and ACM are wildly out-of-bounds. Figure V–16 illustrates what is occurring: the expert judgement of the skin boundary is inside what an edge-defined boundary would be; note that both IS and ACM are agreeing on where the boundary lies, and *a priori* this appears to be a sensible boundary to draw. In this case, though, the body was encased in a gel before freezing, and the expert is accounting for both gel effects and the image pre-processing in locating the skin's boundary. The expert is consistent in this judgement, and ETA has learned this behavior and replicated it. Figure V–17 shows similar results for the skin of the arm. ETA is not simply learning an offset, since the arm skin also abuts the thorax as shown in Figure V–2.

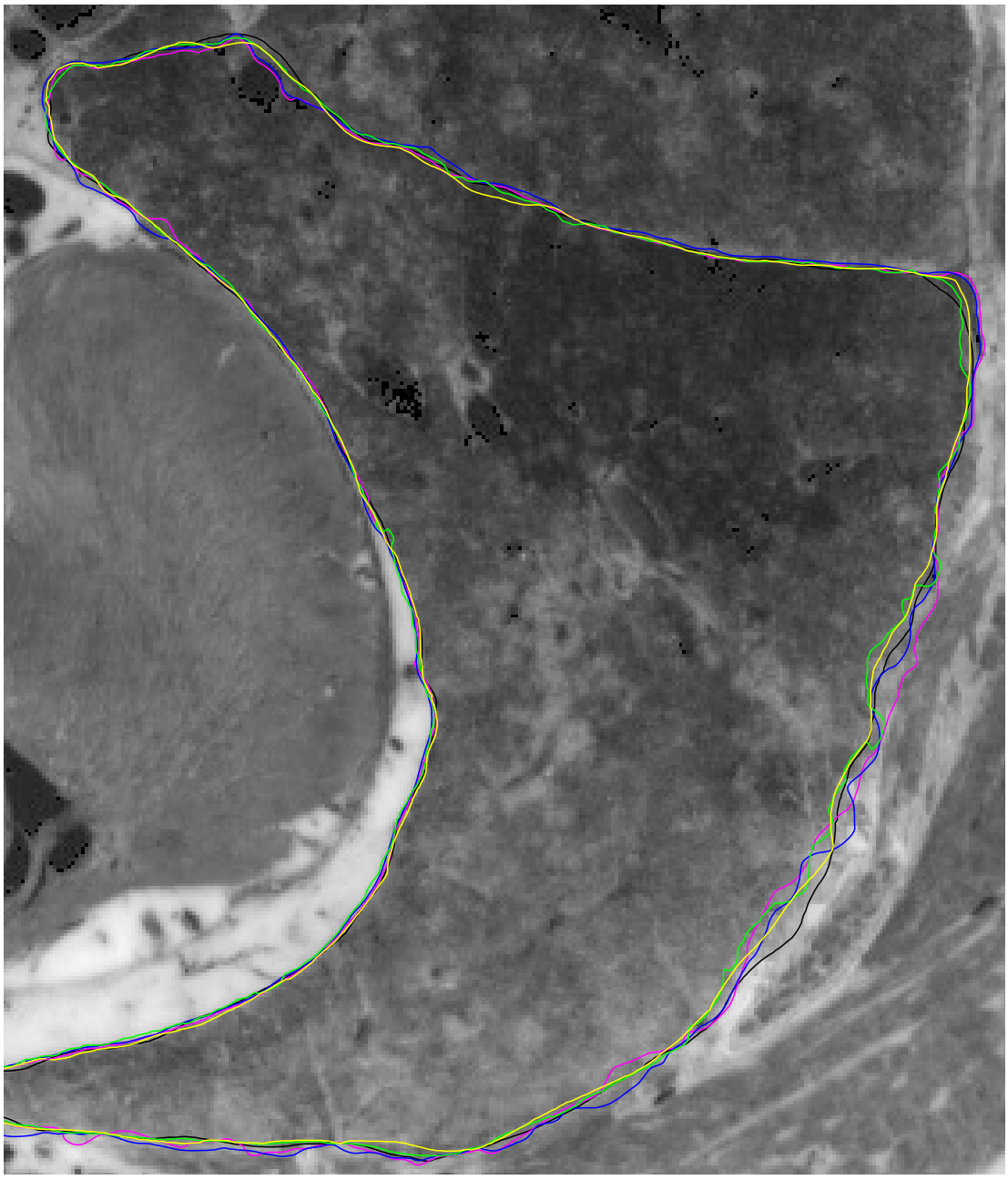
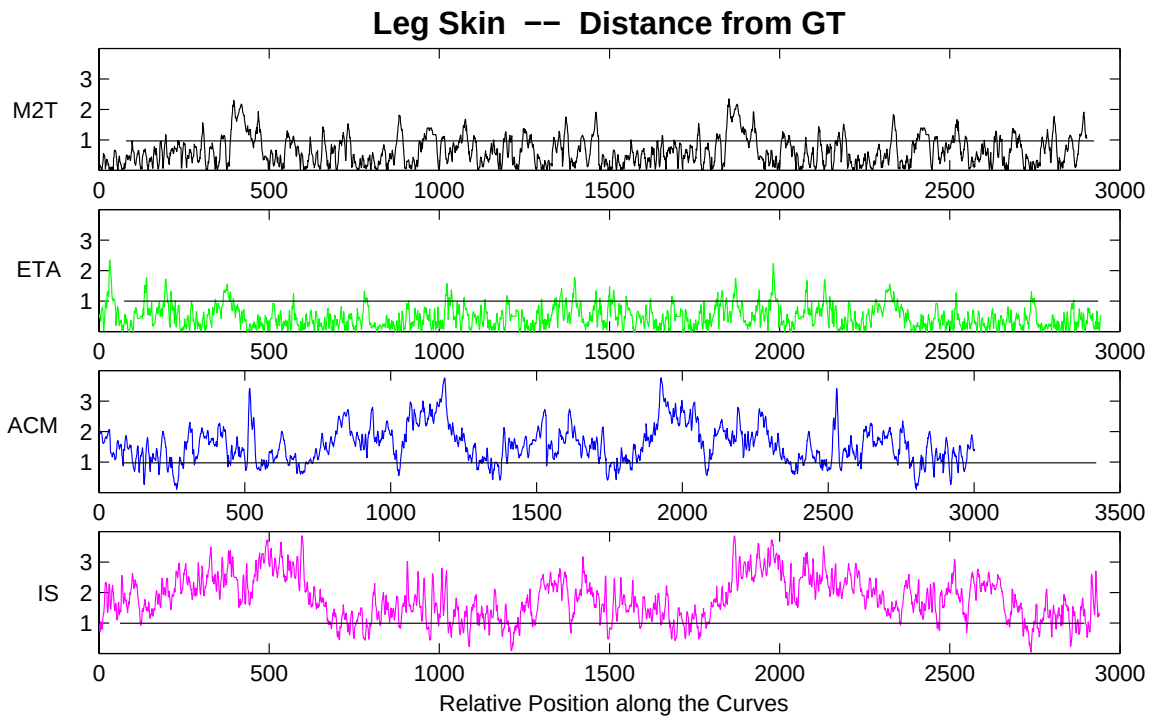


Figure V–14: The outer boundary of the lungs, on the right, against rib and protective tissue, is a particularly troublsome case for all the boundary definition methods.



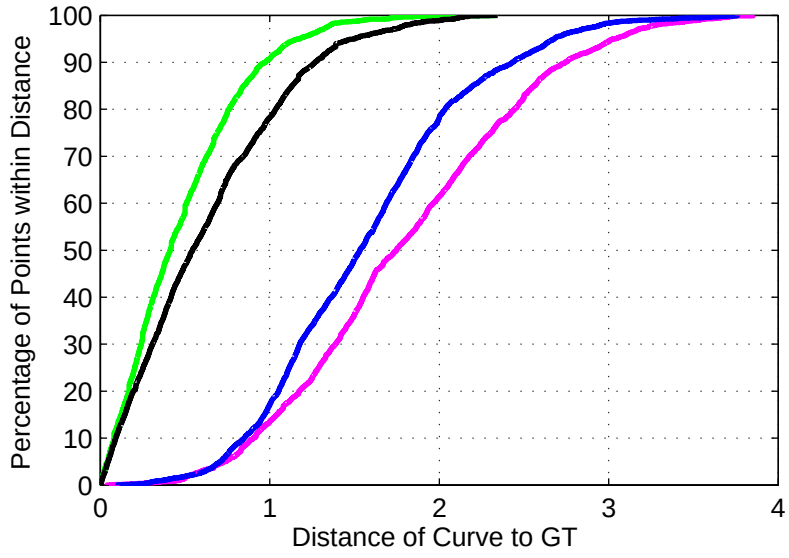


Figure V–15: For the Leg's Skin, the top four position graphs measure the distance from the M2T, ETA, ACM, and IS boundaries to the reference GT; the bottom graph displays the CDF of the for differences sets.

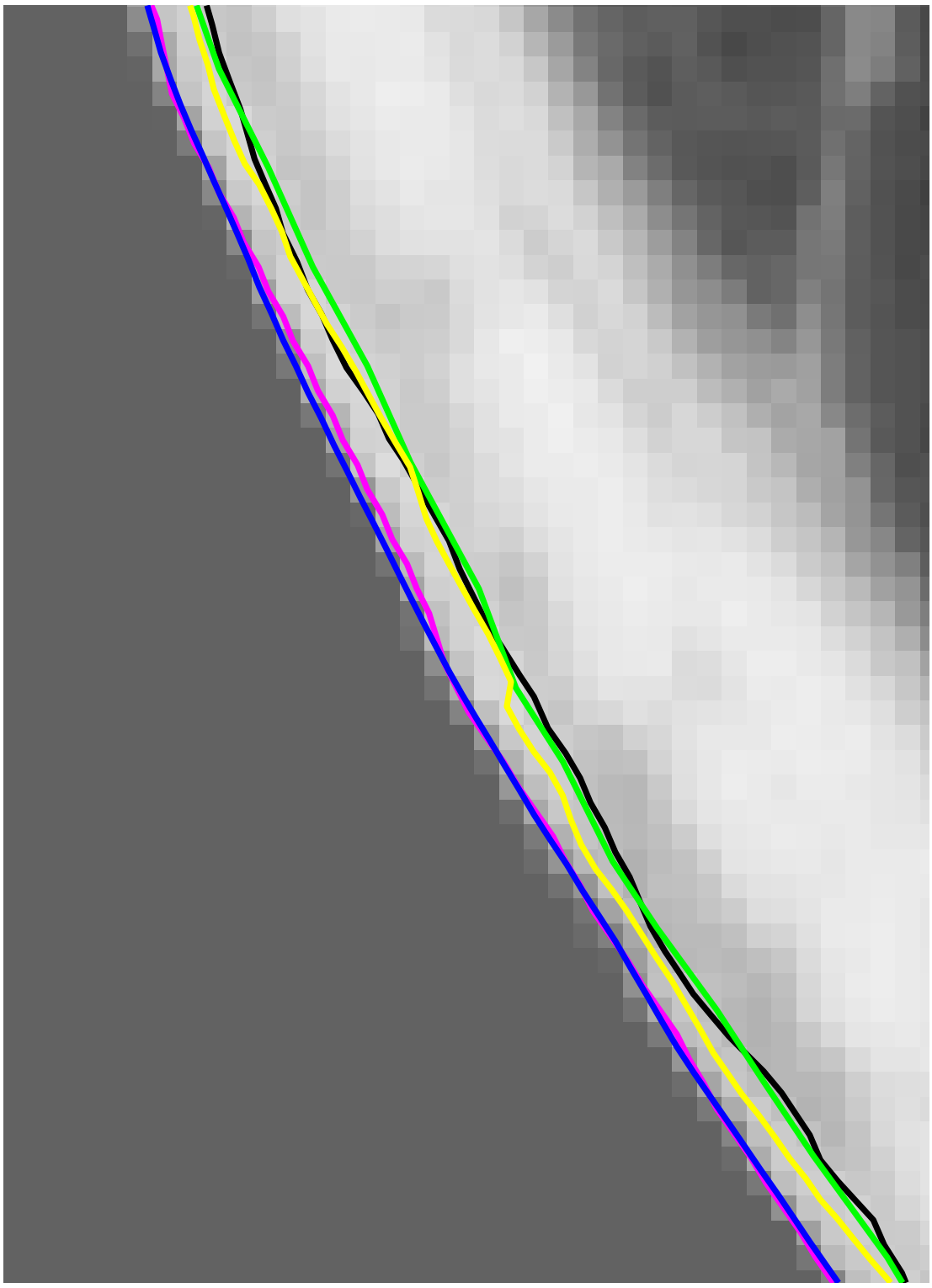


Figure V–16: In this detail of the five boundaries on the leg's skin, ETA is seen to have learned to follow the expert who consistently defined the boundary (in yellow and black) slightly to the inside of what would be classically expected.

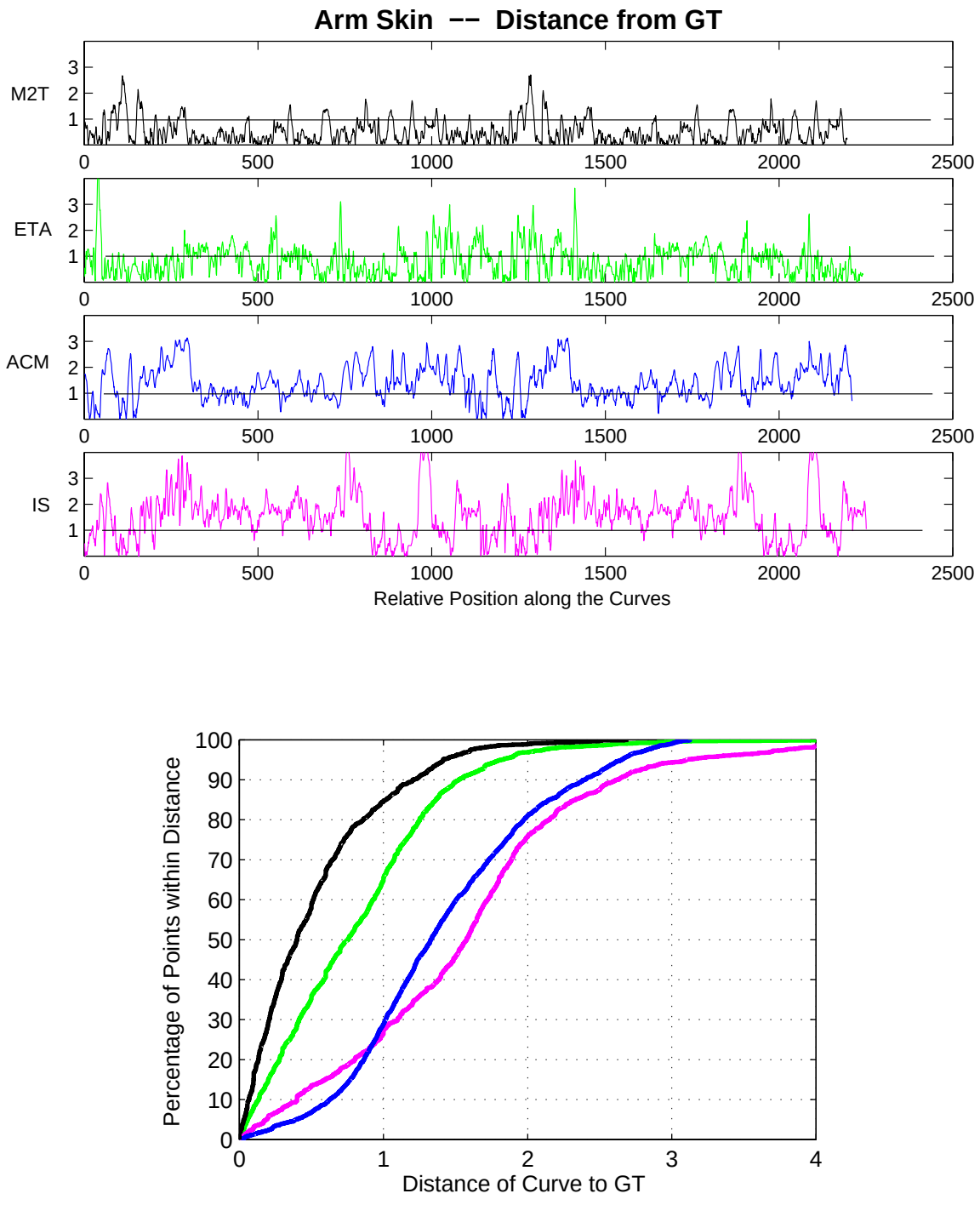


Figure V–17: For the Arm's Skin, the top four position graphs measure the distance from the M2T, ETA, ACM, and IS boundaries to the reference GT; the bottom graph displays the CDF of the for differences sets.

V.1.5. SUMMARY DATA

Table V–1: Summary statistics for the IS, ACM, and ETA methods compared to the baseline intra-user variance.

		% of Curve within 1 pixel of GT				90th percentile of distance to GT			
		M2T	ETA	ACM	IS	M2T	ETA	ACM	IS
BASIC	Leg Bone	80%	96%	91%	82%	1.25	.84	.95	1.42
	Throat	95%	88%	78%	86%	.87	1.06	1.52	1.07
	Leg Muscle	73%	77%	77%	70%	2.10	1.27	1.48	1.52
	Ventricle	89%	84%	71%	70%	1.03	1.18	1.54	2.35
BASIC average		**84%**	**86%**	**79%**	**77%**	**1.31**	**1.09**	**1.37**	**1.59**
INT	Leg Skin	78%	91%	18%	13%	1.27	.97	2.43	2.74
	Arm Skin	84%	65%	29%	27%	1.21	1.52	2.40	2.62
INT average		**81%**	**78%**	**24%**	**20%**	**1.24**	**1.25**	**2.42**	**2.68**
HARD	Arm Bone	75%	71%	48%	53%	1.89	1.86	3.45	3.27
	Arm Muscle	86%	91%	60%	73%	1.09	.97	2.09	1.67
	Thorax Lobe	62%	62%	37%	60%	2.39	2.00	3.41	3.69
HARD average		**74%**	**75%**	**48%**	**62%**	**1.79**	**1.61**	**2.98**	**2.88**

Table V–1 summarizes the data for the nine structures studied. For the four structures grouped as BASIC, where structural boundaries align with well defined image edges, the

average values are all reasonably close. In these well-defined cases, all the methods are basically equivalent. For the three HARD cases, ETA is more consistent with the intra-user M2T. The clearest and most dramatic difference among the semi-automated methods is for the INTERMEDIATE cases: ETA matches M2T, and both are within the one-pixel tolerance 80% of the time, while the other two methods are within tolerance less than 25% of the time.

Not only are the statistics out-of-tolerance for the INTERMEDIATE cases, but there is no adjustment to get the IS or ACM methods to adequately represent what the expert has chosen for the boundary. Since the boundary is NOT coincident with image edges, IS and ACM will always do poorly. ETA however has learned the pattern the expert follows. ETA learned to offset the skin boundary from the apparent image edges when the skin is set against the gel background, and to not offset the skin boundary when following skin-skin edges such as in the armpit area.

V.2. Parameters of the Implementations

The IS, ACM, and ETA methods have been discussed only in generality in previous chapters. This section describes the particulars of how each method was configured and any appropriate parameter settings that were used to generate the comparative boundaries.

For the ACM and ETA methods, in some cases additional minor tuning was made for a particular structure, depending on the boundary traced. While this extra effort might not reflect realistic practice, a principle of this study is to compare each method at its best, rather than with generic average settings that perform only adequately.

V.2.1 INTELLIGENT SCISSORS SETUP

The IS boundaries were generated using the software created by Eric Mortensen [1999]. This software, which captured five years of evolution of IS research, represented the state-of-the-art for the field. The only advance setup required by the system was deciding the various scale factors at which the the image features would then be computed. The Laplacian and Gaussian kernels used were at scales of 5x5, 7x7, 9x9, and 11x11. Since noise was not an issue with this imagery, the larger scale 13x13 and 15x15 kernels which are most useful in noisy environments were not used. These kernels were used to calculate the Laplacian zero-crossings, gradient magnitude, and gradient direction features that feed into the local cost function, as detailed in Section IV.2.1.

Before each image was handled, there was a pause of some tens of seconds while all the features and edge weights were pre-computed. This underpinned the responsiveness of the live wire since all the heavy calculations were done in advance, but it also implied extra overhead for each image. For a several megapixel image, this cost could be significant. This issue was addressed in Photoshop's implementation by restricting the range the cursor can move from the most recently placed control point, and only computing the edge costs within that more limited area.

The method was surprisingly robust in finding paths through indistinct areas, such as the lung's lobe or the arm's skin in this imagery. This robustness arises from the calculation of gradients at several scales and then independently selecting the maximal value among these at each edge pixel. This allows for a semi-automatic adaptation to the appropriate scale of the problem at that point.

V.2.2 ACTIVE CONTOUR MODEL SETUP

For the ACM software, there are five continuous-valued parameters to specify (detailed in Section IV.2.2). This presents a daunting five-dimensional parameter space in which to search for a "best" fit. In this particular implementation, the parameters are set at one value for the entire boundary contour. However, in more general systems and models, these parameters can be specified independently at each control point on the contour thus adding further dimensions to the problem and solution spaces. Additionally, the number of iterations can be used as yet another control parameter, stopping the evolution of the curve when it meets a visually appropriate criteria, rather than computing the curve to convergence. The specification of the initial contour also influences the final state of the contours.

Figure V–18 demonstrates how the evolution and final convergence state of the ACM is sensitive to its starting contour. The figure shows an initial contour in green, a series of contours in yellow each of which was drawn at intervals of five iterations, and a final 30^{th} iteration shown in red. The dark inner structures are muscles, and the light outer structure is the surrounding fatty tissue and skin. Using the default set of initial ACM parameters, the contour can both expand and contract to the nearest and/or strongest image edge attractor.

One initialization problem is that when points on the starting contour are closer to the muscle than the outer edge, they are drawn to that interior muscle as it evolves, and when they are closer to the skin the contour evolves toward the skin, resulting in the ill-formed boundary shown. Also note the similar attraction to the small dark structures, arteries and veins, in this lighter tissue. A useful initialization rule-of-thumb is to keep

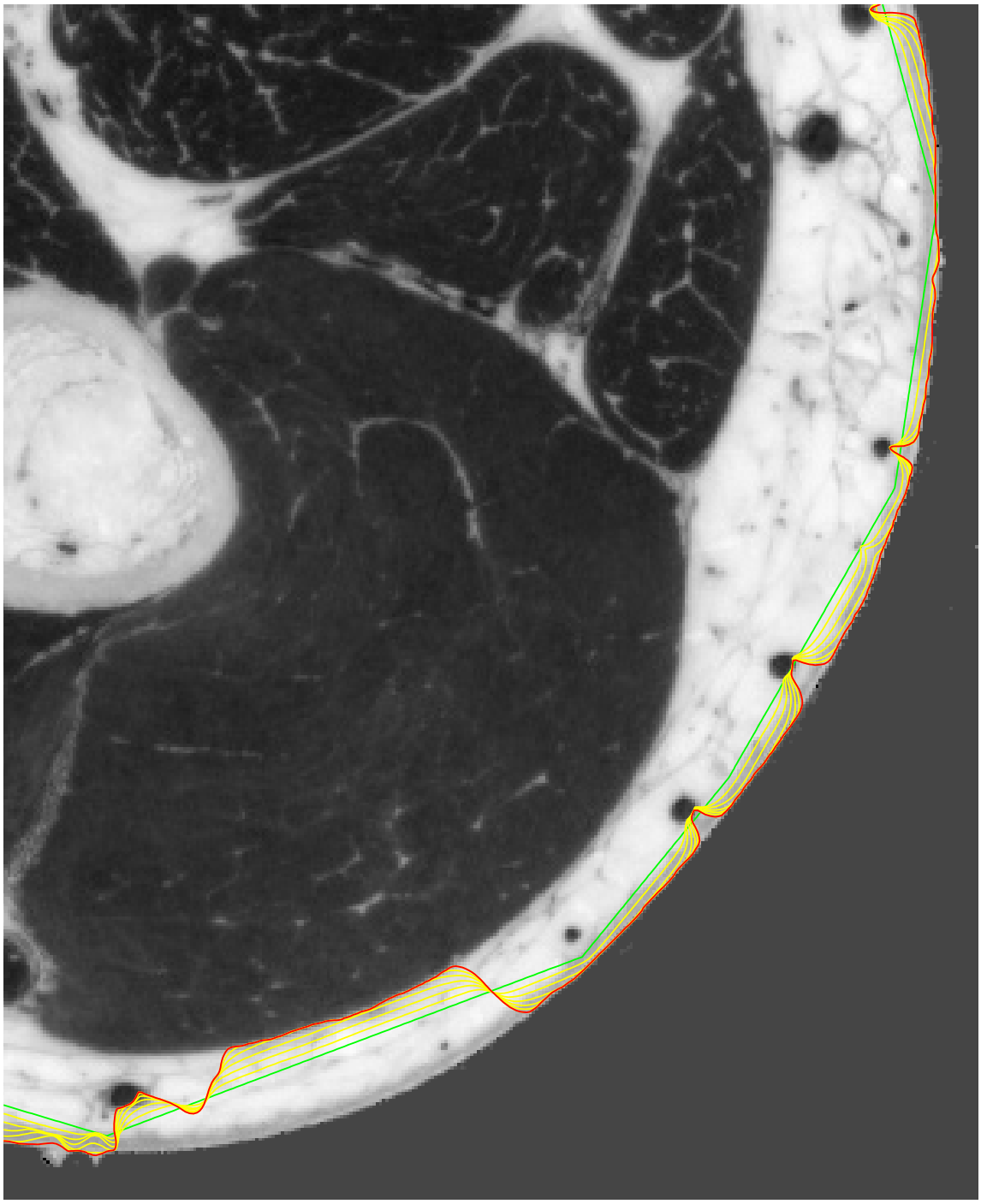


Figure V–18: The ACM runs into problems when the initial parametric curve is not close enough to the boundary of interest. The green line shows the initial boundary segment, the red line shows the state of the ACM after 30 iterations, and the sequence of yellow lines shows the contour evolution at five iteration intervals. Note that when the initial boundary is closer to a dark structure than to the skin, the ACM is attracted to that closer structure as it evolves.

the initial contour closer than half-way to the desired boundary when other strongly attracting structures are present. In the case of the skin, the contour could be initialized slightly outside the skin all the way around, and the contour would then contract without interference to the skin boundary. Conversely, many structures are consistent internally, thus they can be initialized from the inside and will then evolve outward to the boundary.

Settling of the ACM to some final convergence state can be visually observed and qualitatively monitored graphically. Note in Figure V–18 that as long as the progression of contours is evenly spaced, the contour is still actively evolving; as progressive contours crowd together, the active contour is converging to its final resting state.

Figure V–19 shows the ACM as applied to the arm bone. The upper half of this figure shows how the ACM is attracted to the strong light/dark boundary. However, this is incorrect, since the light area includes both bone and connective tissue; the desired boundary is more like what is shown on the lower half of this figure. To get the desired result, several parameters of the ACM can be modified: (1) the "tension" α can be increased, which acts to reduce the overall length of the snake; (2) the sensitivity of the model μ can be decreased, thus the contour is not attracted to more distant features; and (3) the boundary can be initialized close to the desired solution.

The parameter values used for the nine structures of this study are given in the following Table V–2. Initially, the default set of parameters was used, based on the study of their overall effect on the ACM in this imagery, as detailed in Section IV.2.2. When the resultant boundary was inadequate, the parameters would then have been changed based on the particular characteristics of the structure and its surround and the ACM restarted.

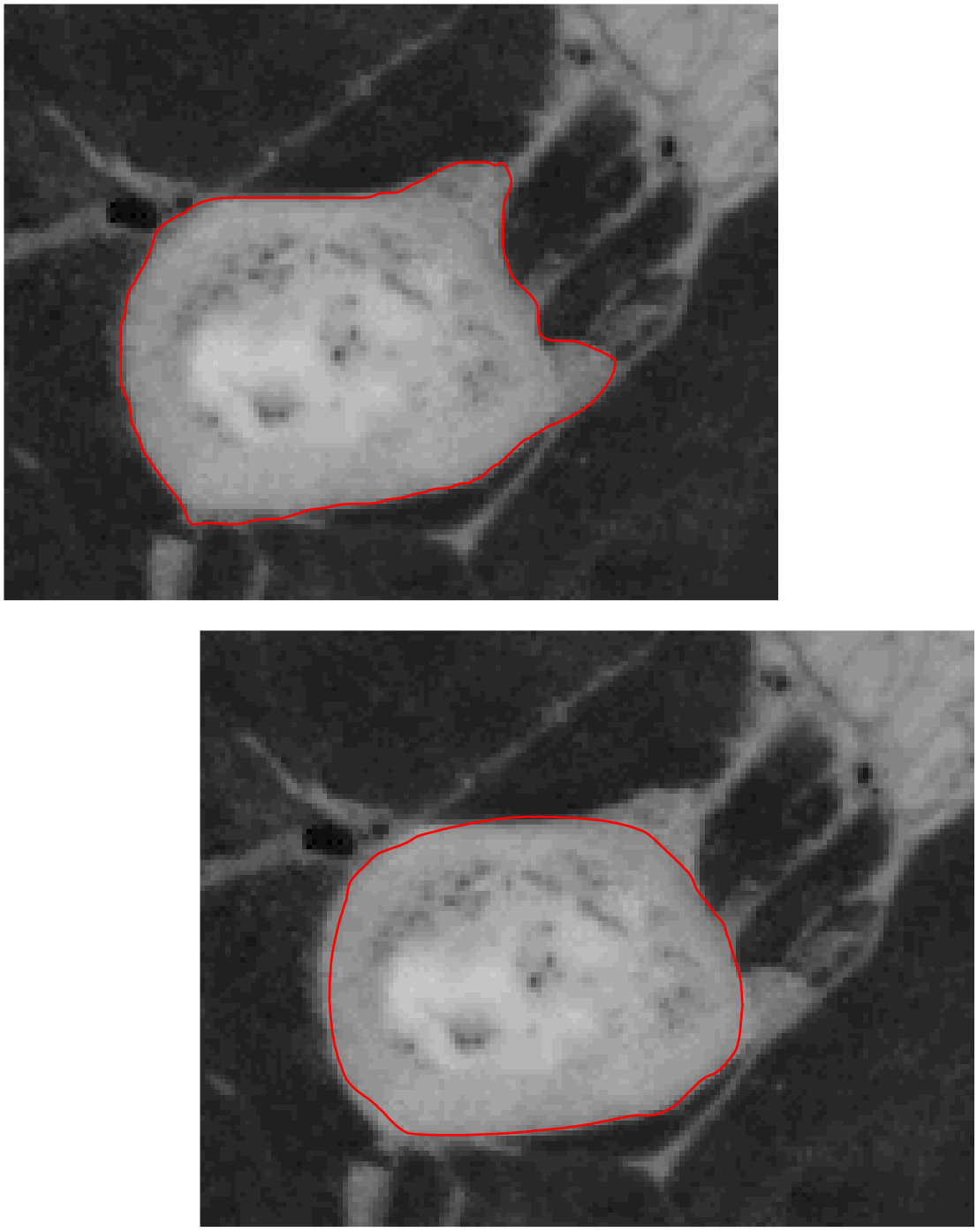


Figure V–19: When the ACM parameters are set for a loose fit (above), the final boundary includes both bone and similar connective tissue. When the ACM parameters are set for a tight fit (below), the final boundary does not truly represent the structure – note the rounded part of the top of the bone.

Table V–2: Summary statistics for the IS, ACM, and ETA methods compared to the baseline intra-user variance.

	α	β	γ	κ	μ	# **of iterations**
Arm – Bone	1.0	0	4.0	0.5	0.1	50
Arm – Skin	10.0	0	2.0	0.5	0.02	20
Arm - Muscle	1.0	0	1.0	0.5	0.1	30
Leg – Bone	15	0.1	3.0.	1.0	0.1	50
Leg – Skin	9.0	0.1	3.0	1.0	0.1	25
Leg – Muscle	5.0	0	3.0	1.0	0.1	25
Thorax – Throat	0.3	0	2.0	0.5	0.1	30
Thorax – Ventricle	0.3	0	2.0	0.3	0.05	15
Thorax – Lobe	1.0	0	4.0	0.3	0.05	50

The arm skin (raw image shown in Figure IV–4) proved particularly troublesome for the ACM. In the upper-left where the arm is touching the chest in the arm-pit area, the skin boundary becomes a faint line, with strong distracting structures (arteries and veins just below the skin) nearby. With the same rationale as for the arm bone, an acceptable boundary solution was found by increasing α, decreasing μ, and initializing the boundary close to the desired solution The fundamental problem is that the underlying edge map is weak, even non-existent, in the troublesome area. A consideration to improve performance here would be to supplement the edge map with information from a line-sensitive kernel.

V.2.3 EXPERT TRACING ASSISTANT SETUP

The ETA input configuration used a feature set that has demonstrated robust performance, as discussed in Section III.5.3. The learning rates were 0.25 for the weights in the output layer and 0.5 in the hidden layer, with a common momentum of 0.5. These values historically proved robust on the visible human imagery in general.

To create the training data, boundaries or subsections of boundaries were sampled. The leg's skin, for example, has a fairly consistent character all the way around, so in this case, the expert's second manual trace (M2T) of approximately 1500 points was sampled at 200 evenly spaced points. Each sample point generates one positive exemplar and three or four negative exemplars, depending on orientation; each negative exemplar was given a 30% chance of being selected, thus keeping the positive and negative examples in approximate balance (as discussed in Section III.3). From this pool of approximately 400 samples, 25% are randomly assigned to the validation set leaving the remaining 75% for the training set. To avoid testing this method on its training set, ETA is trained on the M2T boundary, and compared in this study to GT, the expert's first manual trace. This issue is given further discussion in Section V.2.4.

The arm's skin, in contrast to the leg, has a boundary with two distinctly different aspects. A part of the boundary is similar to the leg's skin, but another part is in the arm pit, where arm skin touches chest skin. If the M2T boundary were evenly sampled, the part of the boundary in the armpit area would have fewer samples than the longer skin-grey boundary, and under-represented things in the training set are not learned quickly. In this case, two different segments of approximately 350 pixels were manually traced to represent the two different boundary aspects. These two boundary examples were then sampled at 100 points each, creating a pool of approximately 400 samples

that was handled as before. As a general practice, when a boundary varied in character along its length, this procedure was used to maintain balanced and representative training and validation sets.

With the training set established, the system was trained through 2500 epochs, and if there is a minimum error across the validation set in that interval, the weights of that epoch were chosen. For this study, a cross-check was made by re-training the network varying only the number of hidden units to see if any significantly new weight behavior emerged. Figure V–20 shows the weights for several hidden unit configurations when learning the leg skin boundary. For each of these configurations, the leg skin was traced out, and the CDFs corresponding to the performance for these different hidden unit cases are in Figure V–21.

The three rows of Figure V–20 show the weights for networks with five, three, and one hidden unit(s). The network was trained for 2,500 epochs. In the first two rows, the left set of weights represents the network state at the validation set minimum (approximately 1,000 epochs in both cases), and the right set of weight represents the network's state at 2,500 epochs. The pattern of the weights, which are almost identical early in training (at 50 epochs) have evolved into two slightly different patterns. A third, minor pattern appears to be emerging with training beyond the minimum validation set error. The third row shows the weights with only one hidden unit, and the validation set error was still declining at 2,500 epochs when training was stopped.

Figure V–21 shows the CDF curves of these five cases against the reference GT boundary, along with the CDF of the intra-user variation. All five boundaries show roughly the same behaviors. The case of only one hidden unit is slightly the worst of the five. The CDF for the 2500 epoch, five-hidden unit case is an improvement over the best validation case epoch CDF, but this data represents the curve after the operator has

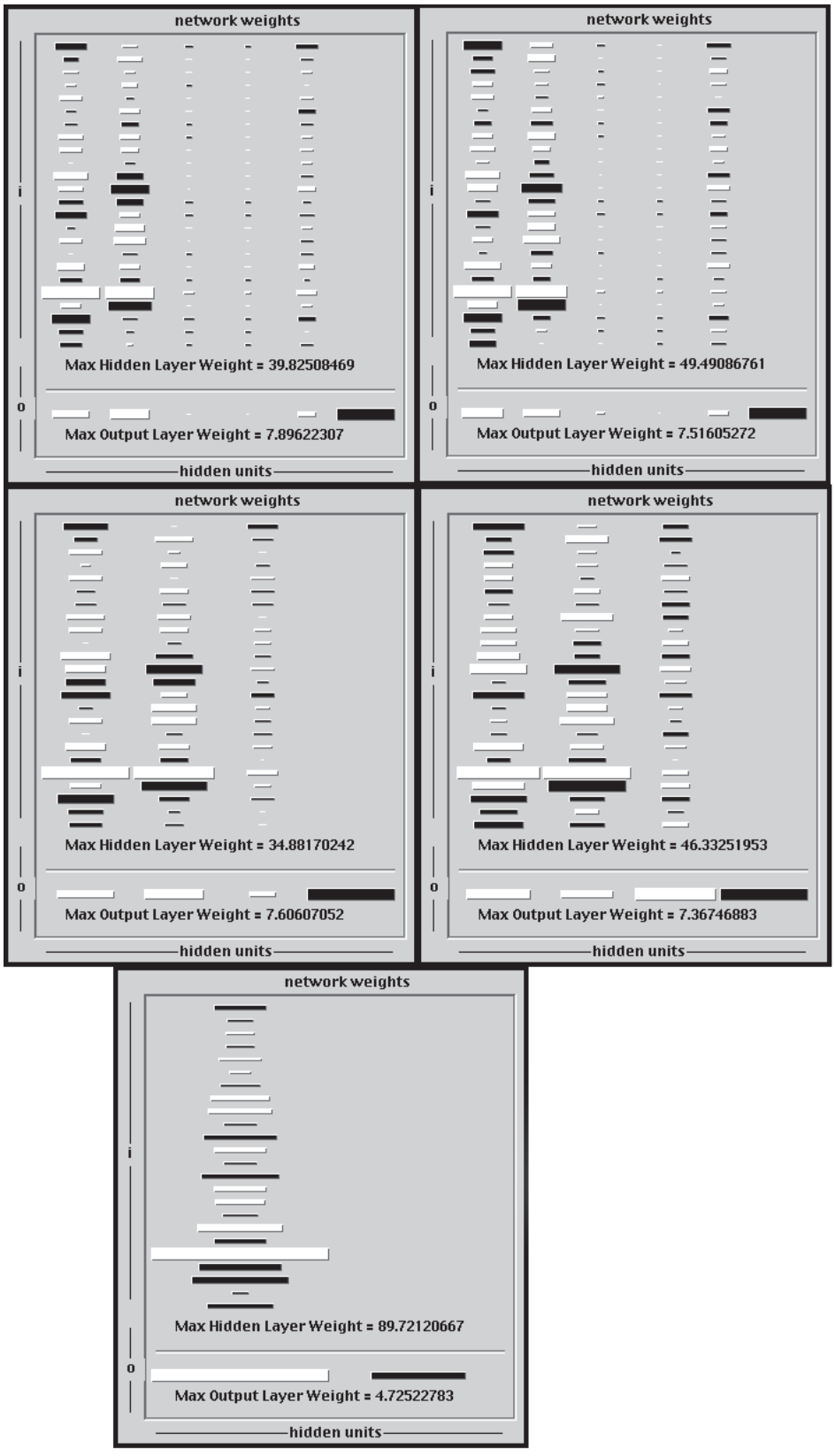


Figure V–20: The neural network weights for five hidden units (top), three hidden units (middle) and one hidden unit (bottom) in learning the leg's skin boundary.

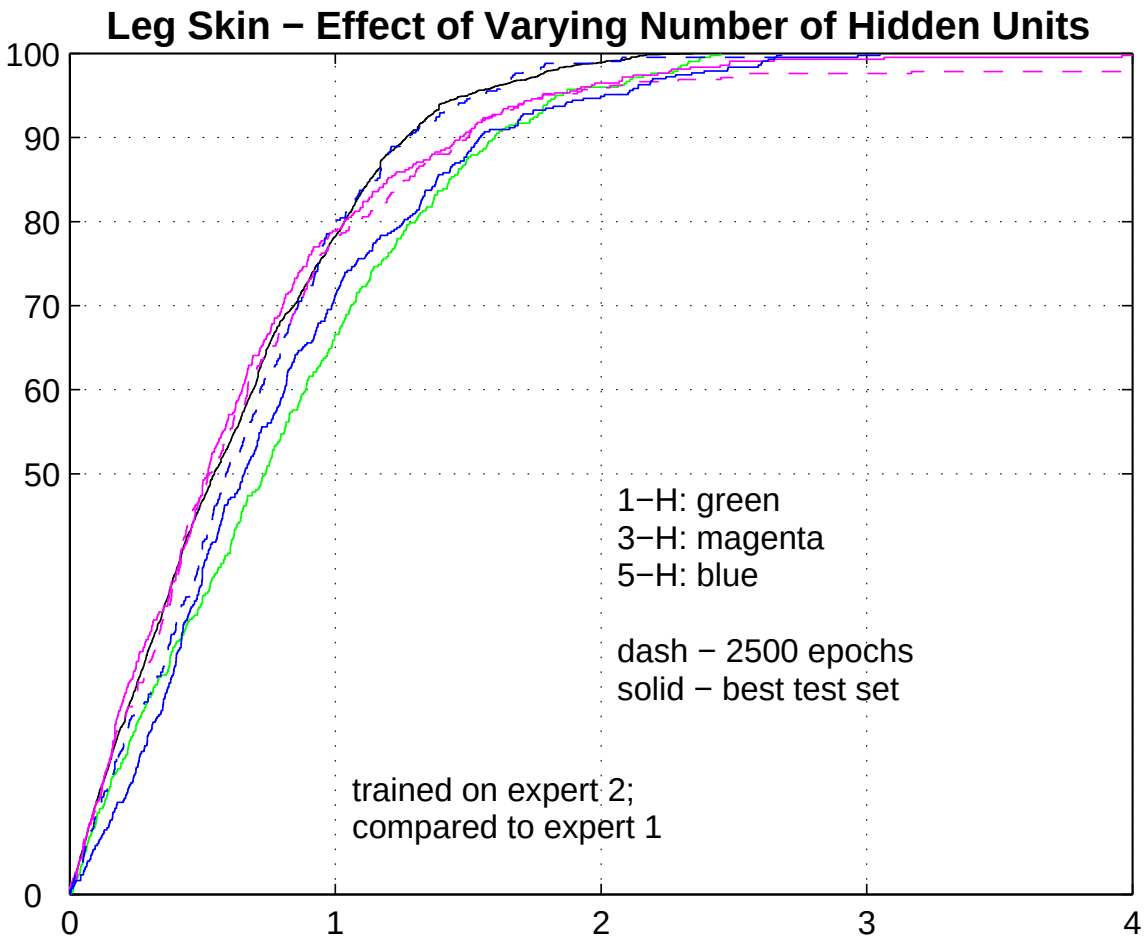


Figure V–21: CDFs for the five network variation, using hidden layer size of one, three, and five units, and using training epochs of either 2500 epochs or training to the minimum of the validation set error.

made ten corrections to the trace. Both of the three hidden unit cases and the intra-user variability are within one pixel 80% of the time. This comparison illustrates the cost associated with over-fitting the data: increased user intervention without any significantly improved boundary quality.

Additionally, this example can be used to study the effectiveness of the early stopping criteria, that is, to stop training when the error on the validation set reaches a minimum. Training beyond this point is referred to as *over-training*. In each of these five cases, the ETA system was used to trace the skin. Figure V–22 shows a particular error behavior that arose in over-training. The black curve shows the boundary which was sampled to create the training set. The white line shows an uncorrected error being made while tracing. With three or five hidden units and when trained to 2500 epochs (well past the validation set minimum), the ETA tracings had these sorts of errors which would require operator correction when they occur. The over-trained network with five hidden units exhibited this behavior nine times in the course of tracing the skin, while the network with three hidden units exhibited seven of these behaviors. The three scenarios where training stopped at the validation set minimum did not exhibit these behaviors at all. All five scenarios required an operator intervention at the bottom of the leg skin image, where there is an actual blip in the skin caused by wires used to tie down the body during freezing.

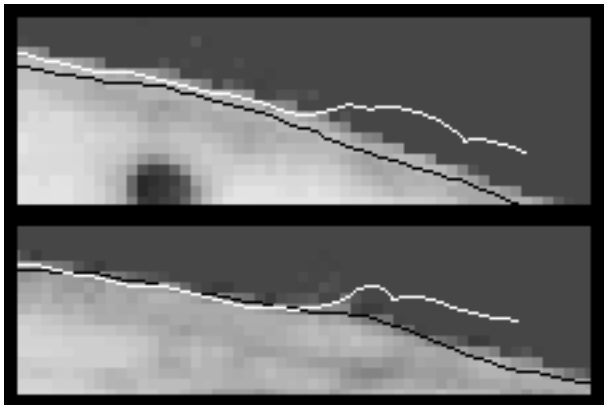


Figure V–22: These two examples are characteristic of mistakes made by ETA when over-trained; the white curve is the uncorrected ETA trace in progress, shown for comparison against the black curve used for training.

The heuristic for initially determining the number of hidden units is to provide at least one hidden unit for each different boundary character that is to be learned. For example, the boundary of the lung's lobe (shown in Figure V–14) has four different characteristic areas: the lobe-lobe boundary (on the top), the lobe-heart boundary (middle left), the curving lobe-fatty tissue boundary (lower left) and the lobe-rib boundary (at right). Thus ETA was configured with four hidden units in this case. Fewer hidden units in a model is preferable to more from a computational perspective: fewer hidden units implies fewer free parameters in the model and thus requiring less training data, and both fewer weights and less training data implies less computation in each training epoch and thus faster training.

V.2.4 Separating Training and Testing

In the operational framework of ETA, a boundary on one or some few images is learned and then replicated on other similar images. This has been observed to work well in practice. Due to shortcomings of the experimental procedure, this working model is not exactly replicated in the comparison study, however. In the study, all the analysis was done on one image per structure, where the expert manually traced each structure on two independent trials. Thus there is a potential problem in that the results are contaminated by generating boundary traces on the same image that was used to train the system.

The expert manually generated two boundaries, GT (*ground truth*) and M2T (*manual second trace*). ETA used some portion of M2T to learn the boundary character, then a boundary **B** was generated by ETA on the same image. **B** generally follows M2T, though with variation. It may be argued that the only reason **B** follows M2T is because that is the boundary upon which the system was trained. This concern is largely mitigated by

how the experimental results are measured and by the sampling of M2T, detailed further in this section. In addition, the leg skin boundary was analyzed to empirically study the sensitivity of the results to the inclusion or exclusion of training neighborhoods from **B**. The leg skin boundary was analyzed since that case most dramatically supports this dissertation's premise and the empirical differentiation between learned boundaries and *a priori* boundaries.

The problematic issue of testing a system on its training data usually arises in the context of a system that, trained on M2T, generates boundary **B** which is then compared to M2T as a measure of success. In the boundary comparisons studied here, however **B** is not being measured against M2T but rather against GT, a boundary on which it was not trained. Thus, as a first consideration in this issue, strictly speaking, the experimental procedure here thus does not measure a trained boundary against its training set.

However, as **B** is generated in this experimental setup, the system will likely encounter some pixels and neighborhoods on which it was trained. Because of the design of the training, though, this will be infrequent. The skin boundary was used with a 23-3-1 network, which has 76 free parameters (the weights). The training set should thus have in the vicinity of 200 data vectors – this is an empirical balance between too few exemplars (which leads to memorizing the data and thus usually poor generalization) and too many redundant exemplars (which implies more computation than necessary). So the M2T boundary of roughly 1,400 pixels should be sampled at approximately 100 points, since each point generates two exemplars, one positive and one negative. The boundary **B** in this case, also of 1,400 points, could thus possibly have at most 100 "contaminated" points, i.e., points on which the system was trained. The remaining 1,300 of B's points, 93% of the total, were not in the training set.

The above numbers are estimates, but they drive the point that even if the data is potentially contaminated by testing-on-training, the contamination can be kept to a minimum through appropriately sizing and selecting the training set. A sensitivity analysis was run to verify this hypothetical argument in practice on the leg skin boundary. The analysis was performed on both scenarios potentially used for selecting the training set. The first scenario is *sampled training*: training points are evenly spaced around the training boundary. If 100 samples are needed from a 1,400 point boundary, every 14^{th} point is taken as a sample point to generate the training exemplars. The second scenario is *segment training*: the training points are taken from a small, continuous segment rather than an entire boundary. In this scenario, the initial 10% of a boundary is used for training, and the system could be tested against the remaining 90%. Segment training is appropriate when the chosen segment is representative of the boundary as a whole. Multiple, approximately equal segments are useful for learning a multi-characteristic boundary.

Applying the sampled training scenario to the leg skin, 160 points were selected around M2T which generated 324 exemplars; these were partitioned into a training set of 242 exemplars and a validation set (used for early stopping) of 82 exemplars. The ETA-generated boundary **B** contains 1465 points. The points are real-valued rather then integer-valued, and due to rounding and smoothing in the boundary tracing process, there is no exact match between any of the 160 training points and the 1465 points of **B**. Table V–3 summarizes the distances from the points of **B** to the nearest point in the training set.

To filter out likely contaminants, a subset **B1** was created by removing from **B** the 218 points that are within one pixel of a training point. To further study the possible influences not only of training points but also of their neighborhoods, subset **B2** was

Table V–3: Summary of distances from boundary to training points.

Number of points defining boundary **B** ...	... that are within this distance (measured in pixels) of a training point
218	< 1.0
328	>= 1.0 and < 2.0
312	>= 2.0 and < 3.0
296	>= 3.0 and < 4.0
280	>= 4.0 and < 5.0
31	>= 5

created by additionally removing the 328 points that are within two pixels of a training point, and subset **B3** was created by additionally removing the 312 points within three pixels of a training point.

Since **B1**, **B2**, and **B3** represent incomplete boundaries, their distance set to GT was calculated by measuring only the one-way distance of those points to the GT polyline. Figure V–23 shows the CDFs of these distance sets. As before, the intra-user CDF is shown in black and the CDF for **B** is shown in green. The CDFs for **B1**, **B2**, and **B3** are superimposed in red, magenta, and blue, respectively. The CDFs for **B1** and **B2** are essentially the same as for **B**, thus the effect of possible training set contamination does not influence this overall result. The CDF for **B3** is slightly different, and actually improved, but not with any statistical significance.

Applying the segment training scenario to the leg skin, the first 220 points of M2T were chosen as the training segment, from which 109 training points were selected (alternate

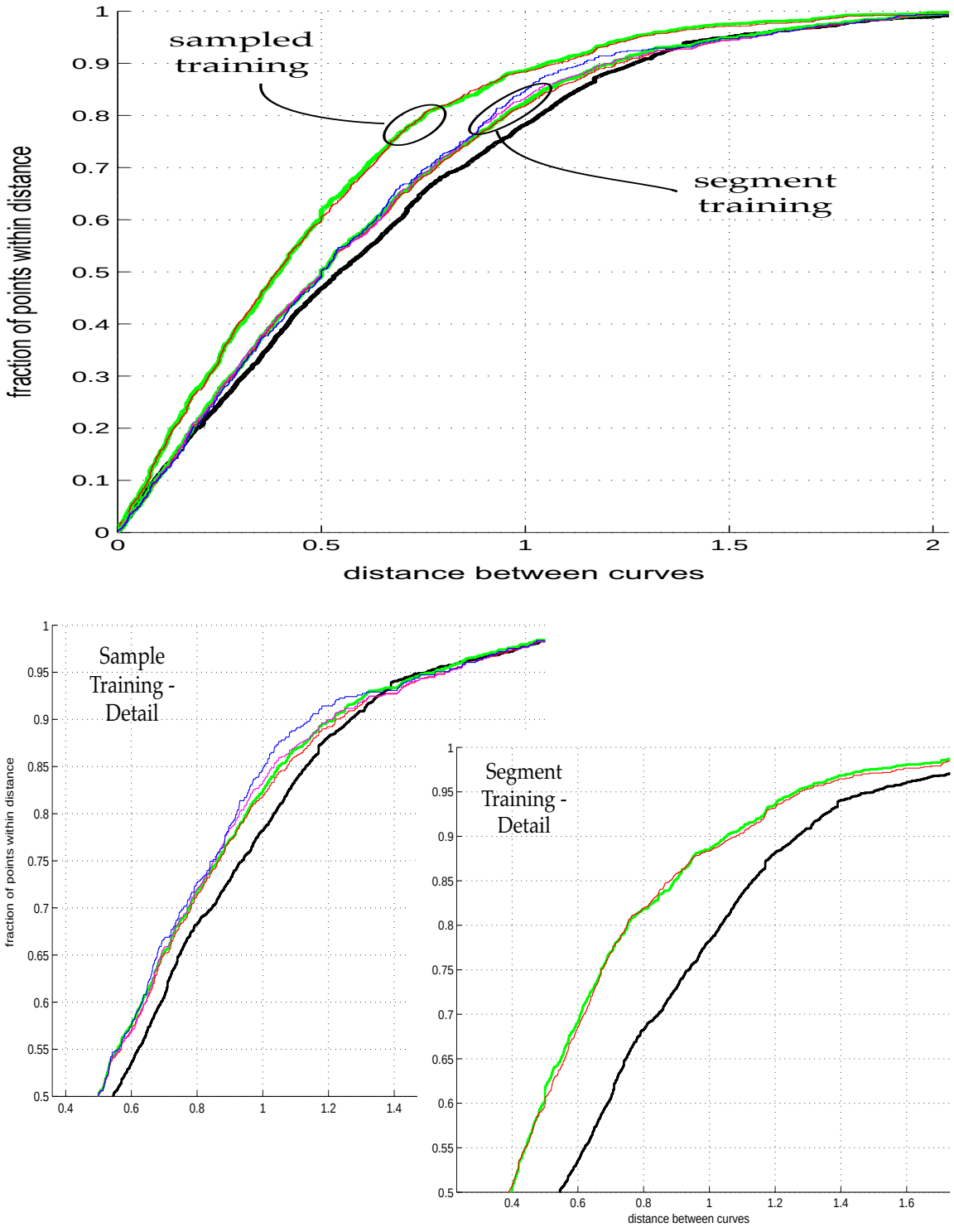


Figure V–23: Two studies of possible training data contamination, for the leg skin boundary; black represents the intra-expert variation. Top: combined results. Lower left, detail for sampled training: green is all points; red excludes points within 1 pixel of any training data; magenta excludes points within 2 pixels of any training data; blue excludes points within 3 pixels of any training data. Lower right, detail for segment training: green is the full boundary; red is the partial boundary, excluding the training segment.

points of the segment are used to avoid the high correlations of immediately adjacent points). These 109 points were used to generate 226 exemplars that were partitioned into a training set of 169 and a validation set of 57. The ETA-generated boundary **Bw** contains 1463 points. After removing the initial training segment from **Bw**, the subset **Bwo** contains 1242 points. The CDFs of the distance sets for **Bw** and **Bwo**, measured as discussed above and shown in Figure V–23, are essentially the same.

The comparison experimentation would have more accurately reflected ETA's operational framework with expert traces on more than one image. Since these were not available, dual expert boundaries on a single image were used. Figure V–23 summarizes the sensitivity analysis discussed in this section, and shows the issue of contamination by testing-on-training-data has minimal, if any, impact on the overall results.

V.3. Required User Interaction

The user interacts with these systems in two distinct ways. The first user interaction is in setting up the system to begin with, deciding on the parameters discussed in the previous section. The second way the user interacts with the systems is by correcting and revising the boundaries as the system creates them. This section looks at the interaction required by the IS, ACM, and ETA methods, in that order.

V.3.1 Intelligent Scissors

Intelligent Scissors (IS) is an inherently user-guided approach. The user must specify an initial point on the boundary of interest, and then sweep the cursor along a path roughly following the boundary until the curve closes back on it's starting point. The closeness with which the user must follow the boundary and the number of control points required depend on the proximity of nearby confounding structures.

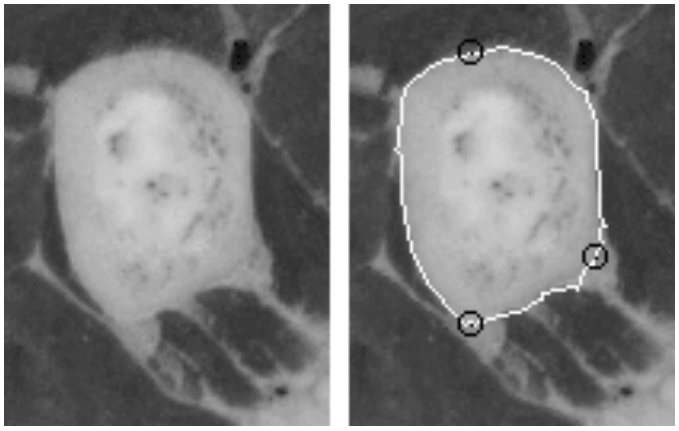


Figure V–24: Raw image (left) shows strong edges around both bone and white tissue; control point placement (circled) forces IS to the bone boundary.

Figure V–24 illustrates how a user corrects an IS contour which will not naturally lie on a desired border. The raw image shows a bone and its connective tissue, both of which are visually similar. The IS contour is naturally attracted to the strongest boundary, but the user can force the curve to a weaker boundary by placing a control point there, as the circles on the right-side image indicate.

There are two situations which will always cause problems for the IS method. These are the Narrow Channel and the Peninsula situations, illustrated in Figure V–25. The peninsula problem arises because the cost of the long excursion around the peninsula exceeds the cost of the shortcut across the peninsula's neck, thus the live wire will always be drawn to this incorrect shortcut. The operator corrects this behavior by placing an extra control point or two at the end of the peninsula to pull the boundary to its full extent. In the Narrow Channel case, when tracing the boundary of A, in the tight channel the IS boundary will be drawn to the shortest path, cutting across the channel to follow the boundary of B before returning back to A.

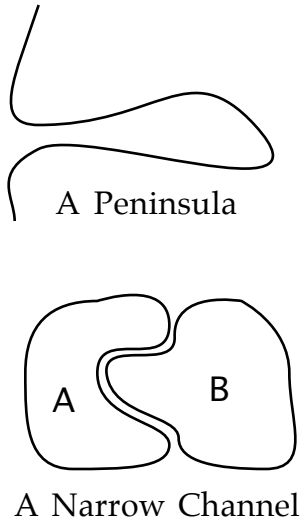


Figure V–25: These structures will always be problematic boundaries for IS.

The control points used in tracing these nine Visible Human structures are shown as red circles on the boundaries shown in Figure V–26.

V.3.2 Active Contour Models

For Active Contour Models, an initial contour close to the boundary of interest is required. This will sometimes be supplied by the user, but for imagery sets representing a sequence of images (e.g., sectional imagery or movie frames), the final boundary from one image may well work as the initial image for the neighboring images. If not, the user will be required to initialize the contour.

In the `GVFsnake` implementation, the user controls the curve by specifying appropriate parameters. Figure V–27 shows the influence of this selection, for a fixed initial contour. The green line is the initial user-specified contour, yellow lines represent the contour after groups of five iterations, and the red line shows a final contour after 50 iterations. In the upper part of the figure, the contour is seen to be extending up and to the left toward other structural boundaries. The large spacing of the contours shows that it would continue to evolve in that direction with further iterations. By increasing the internal tension parameter α from 0.3 to 1.0 and increasing the viscosity parameter γ from 2.0 to 4.0 to slow its evolution in time, the curve converges and comes to rest on an appropriate boundary.

If the boundary is wrong, the user corrects it by respecifying the parameters and restarting the model. This is expensive computationally since the ACM relies on iterative computations to converge. The user must also have a solid understanding or intuition in regard to how the parameters influence the contour evolution in their imagery domain, such as that shown in Figures IV–7 and IV–8.

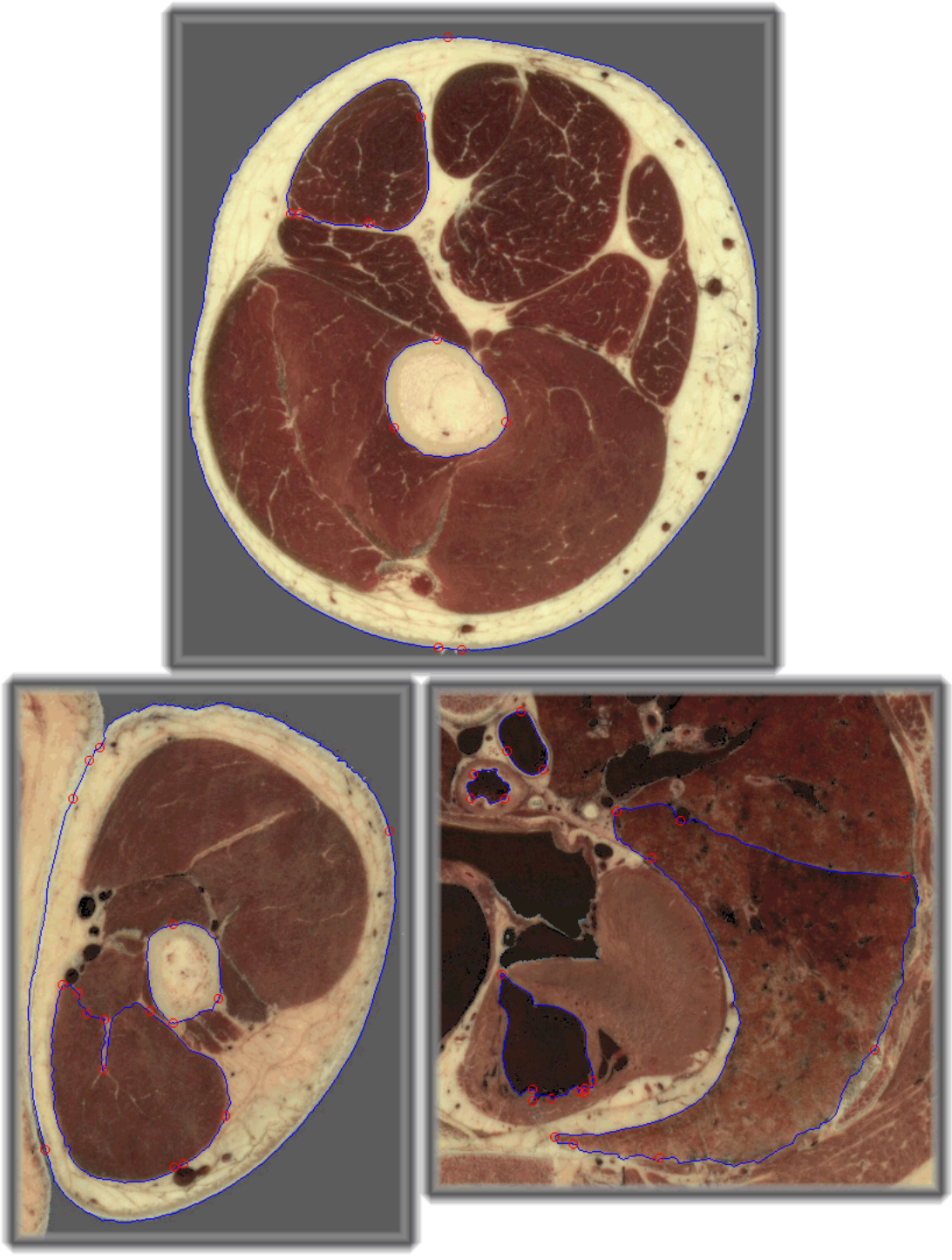


Figure V–26: The nine boundaries defined by IS are shown in blue, and the control points on the boundary required to define them are circled in red.

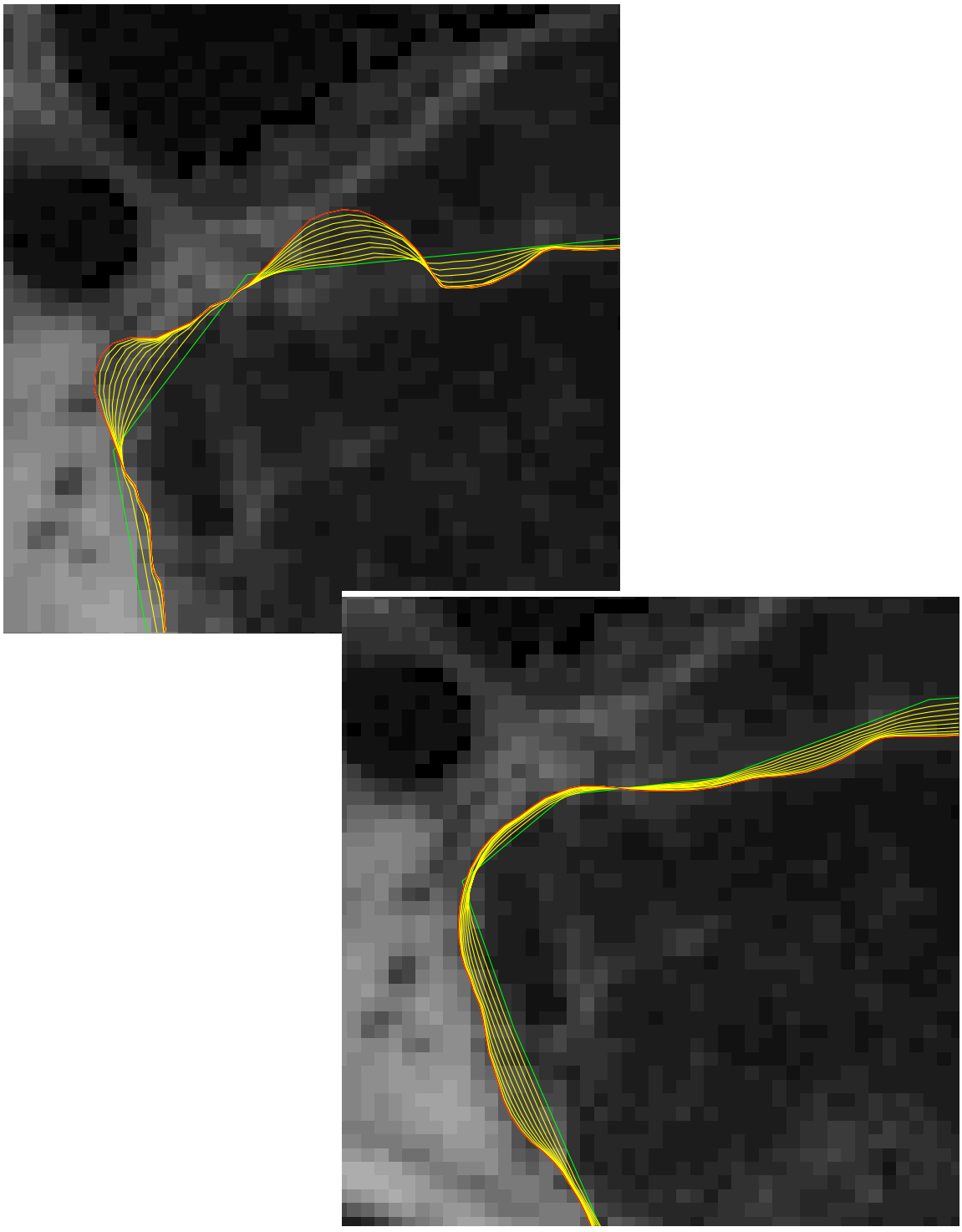


Figure V–27: Two alternative evolutions of an ACM are shown above. The green line is the initial user-specified contour, yellow lines represent the contour after groups of five iterations, and the red line shows the final contour after 50 iterations. In the upper picture, α=0.3 and γ=2.0; note the contour is seen to be continuing up and to the left toward other structural boundaries. By increasing α to 1.0 and increasing γ to 4.0 to slow its evolution, the curve converges to the appropriate boundary.

V.3.3 EXPERTS TRACING ASSISTANT

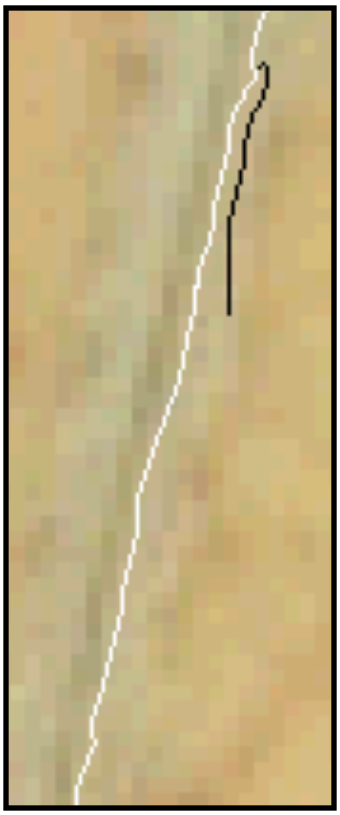


Figure V–28: An ETA trace, after catching a distracting texture, turns around and begins tracing the opposite direction.

Following lines and tight channels can also be problematic for ETA. A line through a consistent background, from a local perspective, will look the same in either direction. Similarly, the wall on one side of a narrow channel will look similar to the opposing wall if travelling in the opposite direction. In these circumstances, should some noise or texture patch happen to turn the trace around, it will continue along in the wrong direction. Figure V–28 illustrates this behavior when tracing the fine line of the arm skin through the armpit. The white line is the ETA-traced boundary, which proceeded from bottom to top, and the small black extension shows where the ETA trace turned itself around and continued on in the opposite direction. The user took control, backed up over the error, and manually traced through the trouble spot until the system could adequately take over again..

The user monitors the ETA as a boundary trace is laid down, and overrides it when it strays from the desired boundary. The goal for this system is to automate the routine parts of the boundary tracing, allowing the user focus their expertise on the confusing parts. In this set of imagery, the most severe user interventions were in tracing the lobe of the lung and the vague portions of the arm muscle. The images are shown here in Figure V–29. In the lobe image, the two expert traces are shown in green and ETA in magenta. In the muscle, ETA is shown in magenta and the expert's boundary in blue.

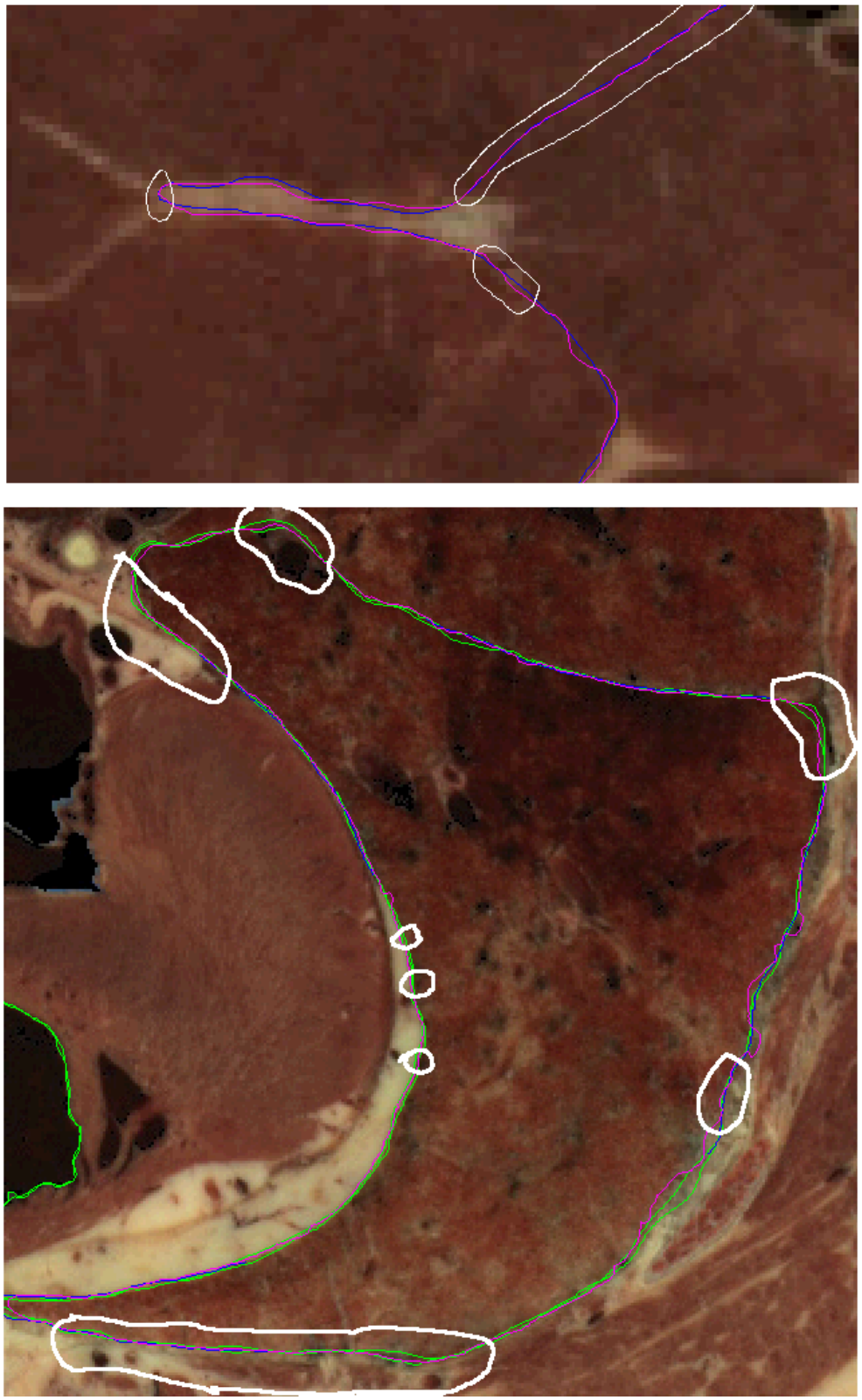


Figure V–29: For ETA, segments requiring user intervention are circled in white. The upper figure is the arm muscle; blue is GT and magenta is ETA. The lower figure is the lobe of the lung; green boundaries are the two expert traces and magenta is ETA.

The sections where ETA required human intervention are circled in white. The arm muscle shows a case where essentially the boundary (upper-left) is interpolated by eye. There is little boundary evidence in this area, and any boundary definition borders on fiction through this part of the image. The lung lobe presented a problematic case for all the methods, and ETA relied on fairly significant manual intervention to trace the structure at all. This case is particularly hard, and no method proved successful on this structural boundary.

V.4. Reproducibility of the Boundaries

People tracing a structural boundary in an image are known to exhibit a variance, both across different users and across the same user at different times (Brahmi, et al., [1999]; Karayiannis and Pai [1999]). In the images studied here, typically 80% of the two boundaries traced by the same expert at different times differed by less than one pixel distance. This user variability provides a reference point with which to evaluate how well the user-guided methods performed.

With IS, the selection of a set of scales determines at what scales the features are computed and then algorithmically combined to weighted edge costs. If the set of scales and features remains the same, a cost-minimal path computed by this methodology will remain constant. Any variation in the IS boundaries will be dictated by the user's interactive placement of seed points.

ACMs are well known to have initialization problems (Mao, et al., [1999]; Gill, et al., [1999a]). Given that the real-valued parameters of the model remain constant, the final boundary will vary depending on the definition of the initial contour by the user, and the number of iterations through the ACM are made.

With the ETA, given a fixed set of input features, the learned trace is dependent on all the training parameters of the neural network. Given a fixed training set, in practice the network settles to similar solutions across a range of reasonable parameter choices. Creating a representative training set, however, is key to this method's success. ETA-derived boundaries will range from poor to excellent, depending on the initial choice of training set. Poor performance, however, can be improved by additional learning, using corrected cases which were initially erroneous.

V.5. Indistinct Boundaries in CT Imagery

The Visible Human imagery is essentially free of any noise. The only complications arise from artifacts of the freezing and sectioning processes. This section will briefly explore the application and comparison of ETA and IS to lower resolution, less clean CT imagery. The comparison here is only to IS, since ACM tracked IS closely in its behaviors.

The base image, shown in Figure V–30, is a CT through the legs of a dog. The contrast range has been selected for the visualization of muscle tissue. Three muscle masses are numbered: note the indistinct boundary between 2 and 3, and where 1 and 2 run along the skin.

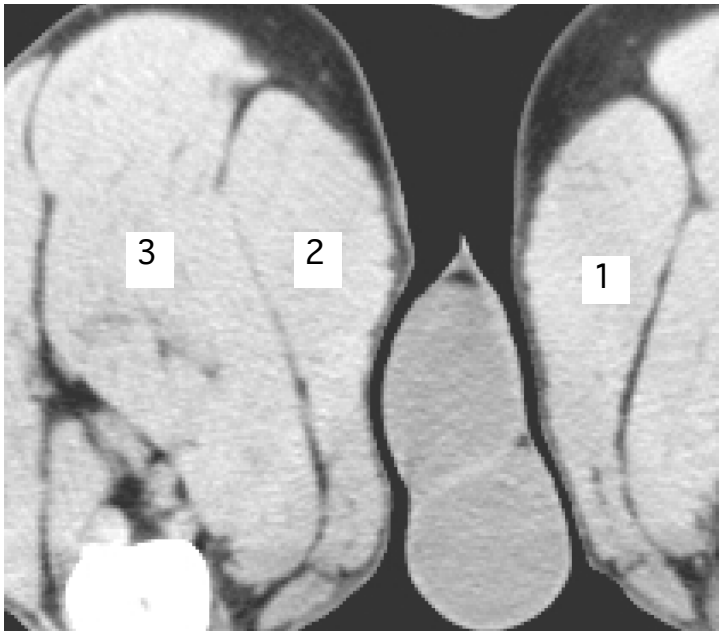


Figure V–30: Three muscles are labelled in this CT image from a dog's legs.

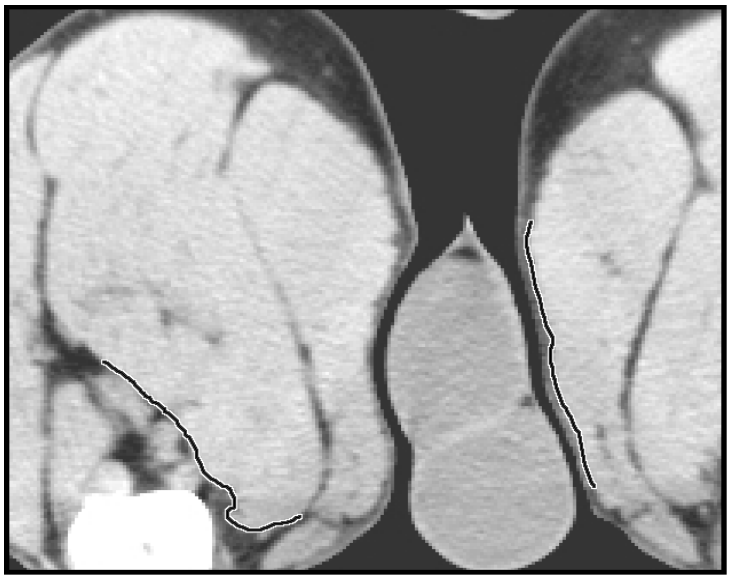


Figure V–31: Two representative segments, shown in black bordered by white, were used to generate a training set.

Figure V–31 shows two muscle boundary segments, shown in black bordered by white, chosen as representative samples for training. One aspect of this study was to explore the effectiveness of a limited number of simple inputs. Five 5–bar inputs were used, centered on the candidate pixel, its two left neighbors, and its two right neighbors.

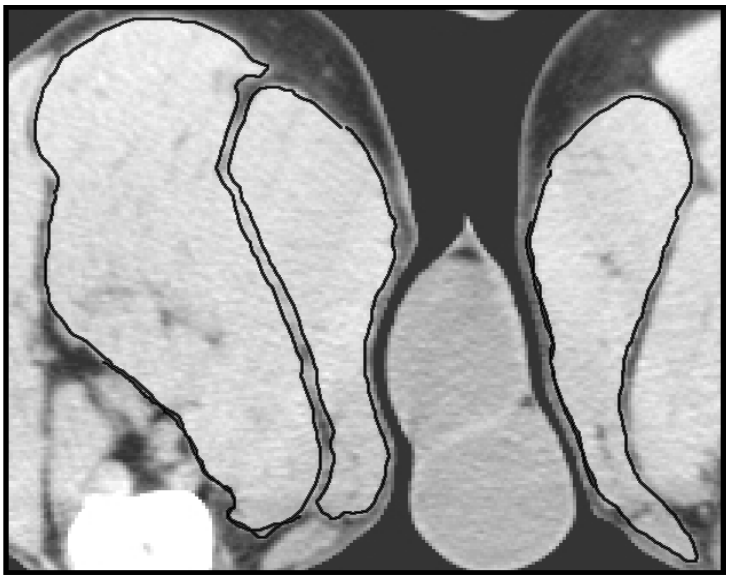


Figure V–32: After training, ETA cleanly traced the three muscle boundaries.

Figure V–32 shows the results of outlining the three muscles with the trained system, superimposed on Figure V–31. User intervention was required only twice, for 5% of the total boundary. Note the clear separation of muscles 2 and 3, and the clean tracking between skin and muscle even though the much stronger skin-to-air edge is only two to three pixels away.

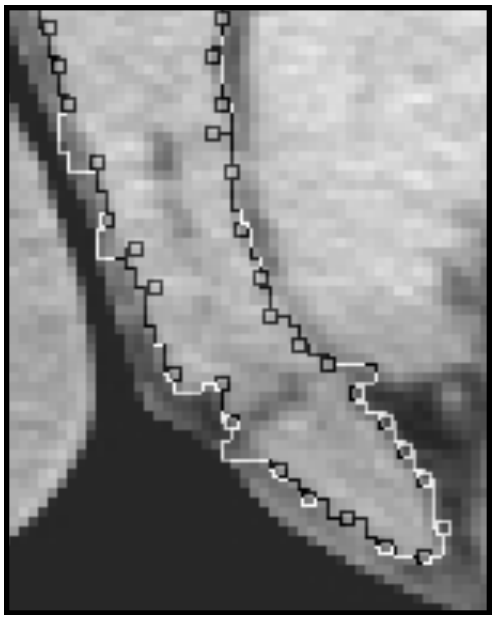


Figure V–33: Detail from an IS defined boundary; the squares represent automatically placed control points.

Figure V–33 shows a detail of an IS defined boundary on the lower portion of muscle 1. The IS tool used to generate this boundary was the magnetic drawing tool from Photoshop, which is Adobe's implementation of Barret and Mortensen's IS work. The contrast sensitivity of the tool is set very low so that the weak muscle-skin edge will be recognized. The distance parameter is set small, which means the boundary definition stays close to the cursor as the user loosely traces out the boundary. The boundary is generally traced well, however as seen in this detail, the boundary jitters when a strong edge runs near the weaker edge of interest.

Figure V–34 shows the same image detail, this time with the ETA boundary. A section of the smoother training segment is visible along the upper left of the boundary. The ETA defined line is jagged, since the boundary points have not yet been smoothed and are placed at pixel centers. Note the boundary stays consistently within one pixel of its training exemplar when it traces over it. In this case, ETA effectively learned to follow only the weak muscle-skin

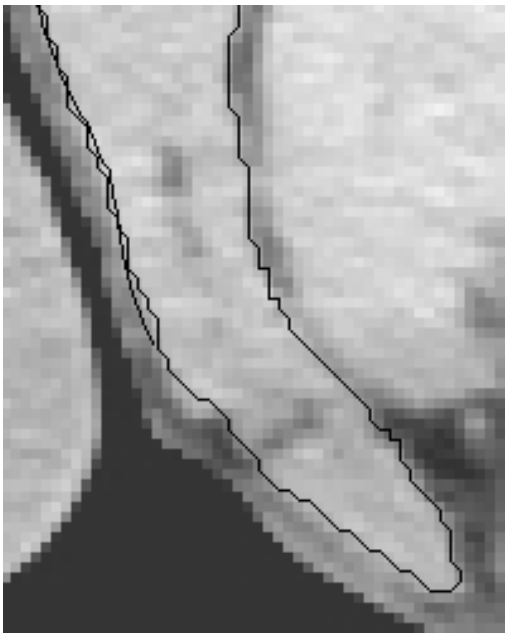


Figure V–34: A detail of the ETA boundary corresponding to the previous IS boundary detail.

boundary and to ignore the stronger outer skin boundary. Both the ETA and IS boundaries can be improved by smoothing, as shown in Figure V–35, however artifacts from the IS edge jitter still remain.

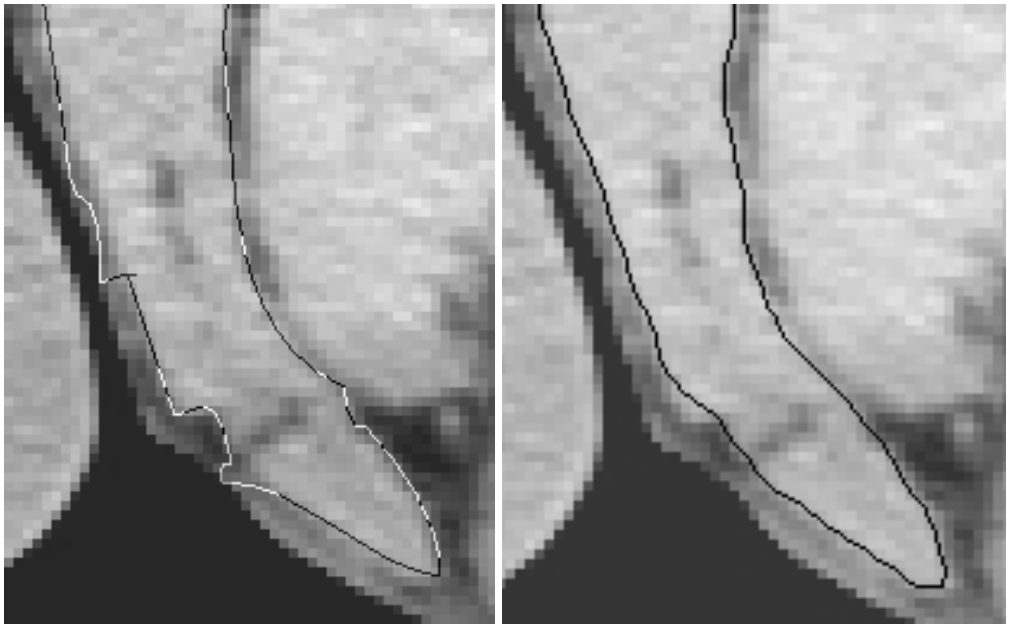


Figure V–35: Smoothed versions of the IS (left) and ETA (right) boundaries.

V.6. Comparison Summary

Studying the empirical cumulative distribution functions of the distances between the ground truth and the boundaries defined by the ACM, IS, and ETA methods revealed several key trends. Figure V–36 shows collectively all nine CDFs for the nine boundaries compared in this chapter. The IS and ACM errors are seen to parallel each other closely, and the ETA usually parallels the intra-expert difference (M2T) closely. In five cases the IS-ACM pair was consistent with the ETA-M2T pair, while in four cases the ETA boundary difference tracked the intra-expert difference and the IS and ACM differences were significantly worse. In the dramatic case of the skin boundaries, ETA was able to much better replicate the expert than either IS or ACM methods.

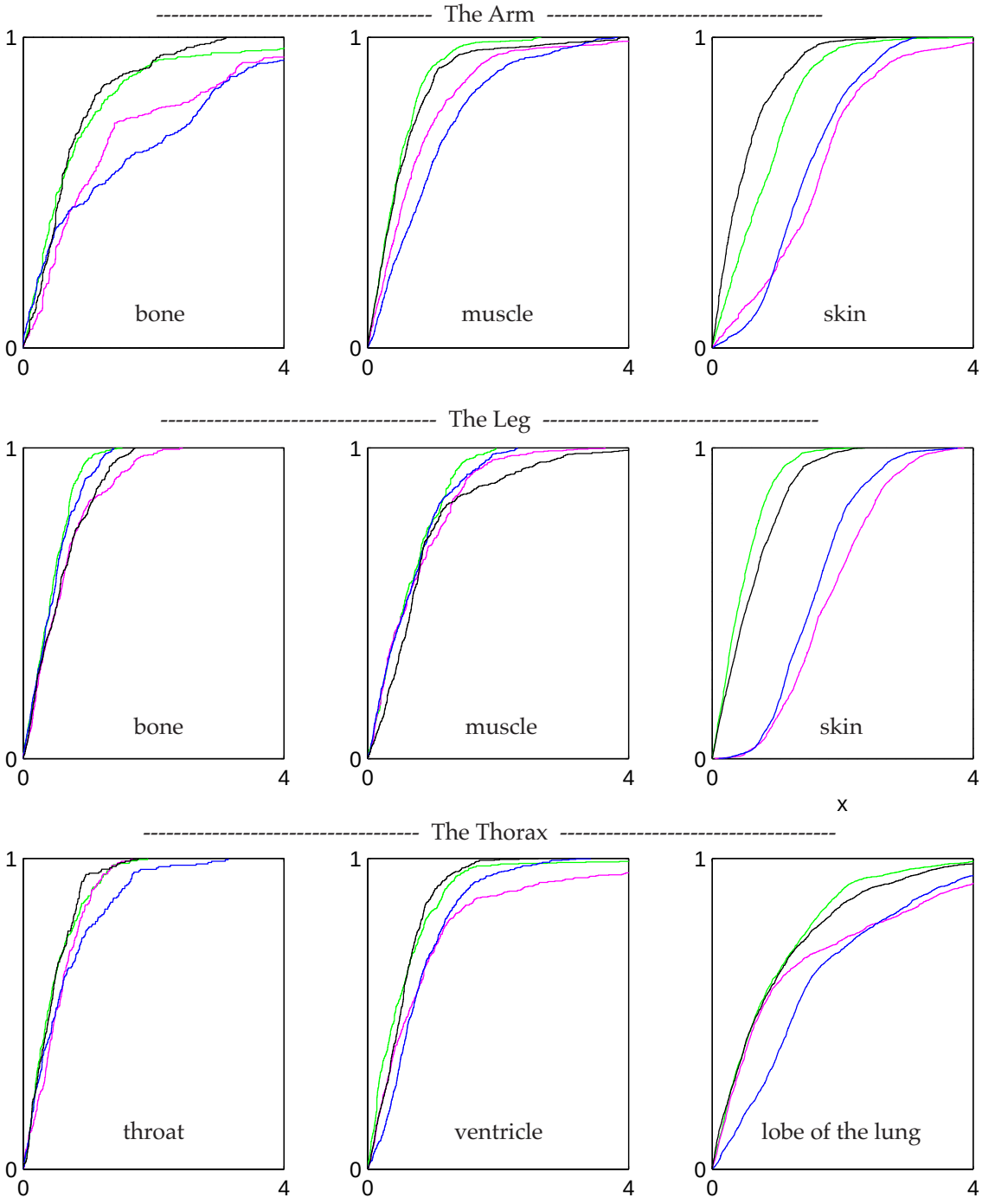


Figure V–36: Nine CDFs for the nine boundaries compared in this chapter.
The IS and ACM CDFs are seen to track each other closely,
and the ETA usually tracks the intra-user difference (M2T) closely.
Color Key: green is ETA; blue is ACM; red is IS; and Black is the intra-user variation M2T.

ETA was the only method to have sufficient flexibility to mimic the expert's behavior in tracing the skin. This flexibility arises from the placement of a variety of input filters at at several offsets, and the learning mechanism that ultimately determines which are important. For the basic skin, the network can simply learn the boundary as a simple offset from image edges. However the network learns more than a simple offset from the background edge, as seen when tracing the arm's skin through the skin-skin boundary of the armpit (in Figure V–2), and as seen in the dog's leg CT imagery (in Figure V–32).

IS at its core relies on *a priori* edge strength measures for defining the cost functions on the arcs between pixels. To an extent, weights on these measures can be adaptively moderated for IS to follow weaker edges, but IS still has little flexibility in the mapping of edge to boundary. To attract a contour, ACM models rely on a pixel feature map that is typically implemented as some variation of an edge map. The feature map could be more flexibly defined over other features of interest, however this flexibility comes at the cost of understanding and defining a custom feature map for every different situation of interest.

ACM guarantees a continuous, closed boundary result since the method starts with a closed parameterized curve and only modifies the parameters though its iterations. IS guarantees a continuous boundary since the core algorithm finds a path from the designated starting point to a designated finish. ETA is myopic by comparison, since at any one point, the extension of the boundary in progress looks ahead only to the extent that the input filter set is sized to do so.

ACM does provide a real-valued set of boundary points at sub-pixel resolution, while both IS and ETA produce boundary points placed at integer pixel centers. IS and ETA rely on post-processing of the integer-valued points to smooth their boundaries.

The main computational burden of IS is in computing, at each pixel, the features on each of the eight arcs connecting a pixel to its eight neighbors. Once this data is available, the user can outline structures very quickly, requiring only a few seconds for simple geometries with strong edges. The feature computations need to be done once for a specific image, and then multiple boundaries can be defined quickly over that image. When changing to a new image, these costs must be borne again. These features could be pre-computed and saved along with the image data to speed future boundary definition on the image. Another option is to reduce the overall computational burden by computing the arc-costs only in the immediate vicinity of the cursor as the cursor moves. This may slow down the ability to define long boundaries quickly, but for large images it focuses the computation only where it is likely to be needed. This is the strategy chosen by Photoshop and PaintShop when integrating this tool into their image editing packages.

The main computational burden of ETA is in the training of the neural network for a specific structure's boundary. Once the boundary definition is learned within an image set, no more computation is required for different images in that set, thus the computation required by training can be paid back only when there are a number of similar boundaries to be traced. One method to speed the learning process is to add each boundary definition to a library as it is created, and quickly run through the library before starting training anew with each new structure. The feed-forward neural network can extend boundaries very quickly, faster than a user can verify its accuracy.

The computational burden of ACM is in the iterations necessary for the initial parametric curve to converge to its final state. This iterative computational burden will be necessary for each boundary defined on each image, which makes ACM the most

computationally expensive of all three methods. For small structures, this may converge quickly, but for large structures this can be a long, costly convergence.

All three methods require a user to start the boundary in an appropriate place. The user thus tackles the difficult problem of sorting through the overall image context and isolating the structure of interest. IS requires the user to place an initial point on the boundary, and ETA requires the user to draw a segment of a few pixels on it. ACM requires the user to fully sketch out an initial, close-enough curve; the closer the initial curve, the less resulting iterations are needed thus speeding the ACM process.

Regarding necessary user interventions, IS has some geometric situations that will always require user guidance, namely the peninsula and narrow-channel shapes discussed along with Figure V–25. Without any adaptive learning of boundary statistics, IS requires close guidance on weak boundaries, especially when neighboring strong boundaries are present. ETA can learn these weak boundary definitions, and has a confidence measure associated with the edge extension so when it strays in unlearned domains, the system can automatically pause and wait for user guidance. When the system continues erroneously on its own, the user needs to backtrack over the false boundary and restart the boundary on its proper course. The user interventions associated with ACM involve the initial parameter selections that are based on general guidelines and specific experience; some ACM systems allow the user to tweak specific points along the curve to pull it into the full extent of sharp concavities which it will not otherwise track.

Intelligent scissors has the best user interface among the methods. The boundary can be defined quickly, and is easy to back up when correcting what is wrong. Once the user is satisfied with a piece of the boundary contour, a mouse click freezes that segment in place preventing accidental changes. These are characteristics that would be useful to incorporate into the user interface of ETA.

——== Chapter VI ==——

Overall Conclusions

VI.1 Summary

Most image processing work addressing boundary definition tasks embed the assumption that an edge in an image corresponds to the boundary of interest in the world. In straightforward imagery this is true, and a wealth of edge-detection research can be applied to the task. However this is not always the case. There are images in which edges are indistinct or obscure, images which can only be segmented by an expert.

This dissertation addresses the range of imagery between these two extremes, the straightforward and the horribly difficult. The premise is that by freeing systems of *a priori* edge definitions and building in a mechanism to learn boundary definitions as needed, systems can do better and be more broadly applicable. This dissertation presents the construction of such a boundary-learning system and demonstrates the validity of this premise on real data.

This work was motivated by existing problems in fully automatic and fully manual identification of boundaries in image processing tasks, such as the analysis of biomedical imagery. Solving these problems to some degree would significantly assist experts in the task of delineating structures of interest in digital imagery. A framework was created for the task in which expert-provided boundary exemplars are used to

create training data, which in turn are used by a neural network to learn the task and replicate the expert's boundary tracing behavior. This is appropriate over large and repetitive image sets, where a small, representative subset of the imagery can be used to learn a boundary representation that can then be exploited on the remainder of the imagery. This is the framework for the Expert's Tracing Assistant (ETA) system.

As an example of the effectiveness of this framework, a version of ETA was used at Visible Productions to trace for a second time the skin of the Visible Male across 1800 high-resolution images. The first tracing was done totally manually, and the inconsistencies in the original manual tracing motivated the re-tracing. The task time was reduced by at least 80%, from approximately three staff-weeks to three staff-days, and the overall set of boundaries was both more accurate and globally more consistent.

The issues addressed in this dissertation arose in the context of this application. The neural network used for this task is a mathematical abstraction, and several issues center around representing and interpreting the parts of this abstraction. Experiments with interpretations of the neural network's output proved that a feature-detector output interpretation, where the output unit goes high (or low) in the presence of a feature and opposite in the feature's absence, was superior to a continuous-valued interpretation suggested by control theory. In the network's layer of hidden unit(s), a natural interpretation arises of that layer as a filter derived from the inputs, which in turn is used at the output unit in a boundary decision criteria. Experimentation with the network's inputs demonstrated that for a fixed number of inputs, preprocessing the input data can bring about faster learning than simply presenting raw pixel data as inputs to the network.

In this application, there is no lack of training data. An expert can create a representative training segment of 200 pixels in length in less than one minute. At each

point one on-boundary exemplar (a correct boundary continuation) and several off-boundary exemplars can be constructed, thus generating a pool potentially of 1,000 training exemplars. Different classes of exemplars may be disproportionately represented, however, and this was shown to skew the results of training when the negative exemplars outweigh the positive. At a 4:1 ratio of negative to positive exemplars, the correct responses were inadequately learned. By randomly choosing only 25% of those negative exemplars to include in the training set, both positive and negative cases were adequately learned, as was measured by the clustering of responses about their desired targets. This issue also arises, for example, when a structure is bounded by a light background most of the time and a dark background only a small percentage of the time. The boundary sampling needs to represent the differing boundary character approximately equally for the neural network to learn both boundary characteristics adequately.

While the ETA system proved useful in practice, an experimental comparison to other user-guided boundary definition methods was performed to explore its strengths and shortcomings. For a representative set of nine structures in the Visible Male cryosection imagery, ETA was compared and contrasted to two other state-of-the-art, user guided methods – Intelligent Scissors (IS) and Active Contour Models (ACM). Each method was used to define a boundary, and the distances between these boundaries and an expert's ground truth were compared. There is a natural variation between independent boundary traces made by a human, and the three semi-automated methods were compared to this intra-user variance.

Studying the empirical cumulative distribution functions of the distances between the ground truth and the boundaries defined by the ACM, IS, and ETA methods revealed several key trends. The IS and ACM errors usually paralleled each other closely, and the

ETA errors usually paralleled the intra-expert difference (M2T) closely. Across the nine structures compared in Chapter V, in five cases the IS-ACM pair was consistent with the ETA-M2T pair, while in four cases the ETA boundary difference paralleled the intra-expert difference and the IS and ACM differences were significantly worse. In the dramatic case of the skin boundaries, ETA was able to much better replicate the expert than either IS or ACM methods. In this case, where the expert's judgement was most called into play to bound the structure, ACM and IS could not adapt to the boundary character the expert used while ETA could.

Researchers have observed that experts have resisted using automated systems that are available, as noted in Section I.1.2. This may result from many factors, from inappropriate system behaviors to a poorly designed user interface. One reason may be that the system is perceived as doing its own thing rather than what the expert is deciding upon. A selling point for the adoption of systems which learn a boundary from examples is that the system will be taught to match the expert, and thus its results should be viewed more acceptably from the expert's perspective.

VI.2 Limitations

The learned boundary was shown experimentally to succeed in many cases. Uniformly bounded structures are the most successfully sampled, learned, and automatically traced. In some structures the boundary character varies, for example the arm's skin where the boundary was either skin-against-background or skin-against-skin in the armpit area. Multi-characteristic boundaries such as this can be successfully learned when the training set has approximately equal representation from boundary segments representing each characteristic. And in structures where an expert's boundary judgement is at variance with traditional image edge definitions, the implicit expert's

boundary definition can be successfully learned by example. This was demonstrated by the results in defining the boundary of the Visible Human's skin and the muscles on the CT image of the dog leg. Visually similar, though still slightly distinct, structural boundaries can be learned, such as the bone-ligament boundary analyzed in the Visible Male imagery. In this case, a minor operator intervention is usually required to set the system onto the proper boundary, but then the tracing proceeds well. The results of both these cases is detailed in Section V.1.

One limitation of the experimental method used in comparing the boundary methods was reliance on the ETA user to understand the implications of a multi-characteristic boundary situation. Ideally, the user would recognize the situation beforehand and provide roughly equal training segments for each characteristic. When this isn't anticipated, the recognition that a boundary characteristic is inadequately learned comes when the automatic tracing fails miserably in one section. The user then simply backs over the problematic segment and manually traces in the proper boundary. The user has the option of providing separate training segments for this region and then retraining the network. The learning can only be as good as the chosen exemplars.

Several situations were isolated where the learned representation proved troublesome. The "thin channel" situation is one, where a narrow structure is bounded on either side by similar image characteristics; the narrow structure is said to have close, anti-parallel edges. In this situation, a slight perturbance caused by noise or a small nearby structure may cause the boundary trace to change direction, and the system picks up tracing the structure's opposing side in the opposite direction. Additionally, small and strongly distinct structures or unrepresentative boundary anomalies can kick the boundary tracing off-course. These situations require operator intervention to correct. And some tasks are resistant to any accurate and consistent bounding, such as the case presented

by the lobe of the lung, when even the intra-user variance was exceptionally high. For such exceptionally hard cases, an expert will always be needed.

The learning approach used in the ETA framework as discussed in this dissertation was static in the sense that the training set was defined and then never changed. The way to further improve a learned representation for a boundary is to analyze the errors made initially and learn the correct responses for them, however additions to the training set were not used during the experimentation in this work.

The raw data used for each pixel's input was one eight-bit channel of information. Sometimes eight-bit greyscale is the norm, for instance with MRI imagery. However X–ray CT imagery typically is captured on one channel of 12 bits, and the Visible Human dataset is captured on the three RGB channels of 8 bits each. When working with 12-bit CT imagery, the user tuned the displayed image to best highlight the structure of interest, and the 12-bit image was saved in a reduced 8–bit range spanning the appropriate image intensities. In the three channel RGB imagery from the Visible Male, the 8-bit green channel was used since it provided the best visual distinction among structures overall. While these are both reasonable choices for deciding upon a significant eight bits if data, they are still user-dependent, relying on an appropriate user choice.

Another shortcoming of the experimental methodology is that there was no attempt to normalize the imagery for differences in image intensity or inconsistency. When the Visible Human imagery was captured, the lighting conditions were well-controlled, and thus the images are consistent both across the full extent of each image and between images in the set. Other imaging modalities are not as easily usable, however. MR imagery, for instance, typically has both significant inter-slice and intra-slice intensity

variation. This could confound a learned boundary definition as the user moved among these areas of intensity variation.

Choosing neural networks as the learning mechanism in this system did have some overall implications. The backpropagation method can be slow, especially when several distinct characteristic segments are used with multiple hidden units to learn a multi-characteristic boundary. There is no proven way to select a best network size, and trial and error was used to select the learning parameters. Neural networks are often problematic since the representation they learn is unclear, however in this application the hidden layer has been shown to have a straightforward interpretation, at least with simple inputs, as a compound filter.

While it rarely proved problematic in these experiments, the learned boundary is directionally dependent. If the training segments were traced clockwise, the boundary continuation would be learned and preferentially extended in that direction. When defining several training segments for a multi-characteristic boundary, segments must be defined consistently either clockwise or counter-clockwise, since mixing directions will likely create a contradictory training set.

Each result presented was checked against a few variations in the random weight initializations and number of hidden units. No comprehensive study was made of the overall robustness of results to variations in learning parameters, weight initializations, image noise levels, or input or hidden unit configurations.

VI.3 Future Directions of Study

The learning approach used in the ETA framework as discussed in this dissertation was static in the sense that the training set was defined and then never changed. However,

the interaction of system and user in ETA framework leads to a natural synergy. The simple fact that the user has taken back control indicates the system has erred, and by collecting exemplars as the user backs up over a mistake and redraws a boundary correctly, the system acquires exactly the data needed to improve the learned representation.

With a larger training set of exemplars, a naive approach would be to start over again and retrain a new network from scratch. This is inefficient, since it implies relearning what was initially learned as well as the new data. Given these new exemplars, the issue is now how to change the learned representation to incorporate the new data without losing the information already assimilated from earlier rounds of training.

Only a single eight-bit value was used for the raw data associated with each pixel. For RGB imagery, using three values per pixel would triple the number of inputs, triple the associated number of weights from input to hidden layers, triple the number of free parameters thus increasing the needed number of training exemplars, all of which imply a dramatic increase in required training computation. Given the benefit of using only a single channel of input data per pixel, what should be used when multi-channel data is available? The choice of using the green channel was made for Visible Human imagery, however the best distinction in the data may not lie in simply one channel. RGB may be transformed into other color spaces, such as HSI or Lab, whose dimensions may better distinguish the boundaries. Another option would be to pre-process a sample of the imagery and find the single dimension of the high information content through a principal component analysis of the sample.

The user was responsible for providing a training set that is both representative and balanced. Balancing the training set, providing roughly equal number of samples for distinct variations in boundary conditions, can be reasonably done at a gross level. But

there may be fine structural variations that are hard to distinguish. One possible approach to addressing the issue of balance would be to implement some fast clustering of the exemplars, then sample equally from each cluster to create the training set. A rough and approximate clustering is all that is required, since this is used not to predict a response but to select the training set used to learn the representation.

The success of ETA in the comparison experimentation demonstrated that the selected input features were sufficient for the task, but this says nothing about the necessity of the features used. Future experimentation could study the adequacy of the boundary representation learned as a function of features in the input set. The weights on the features can be indicators of which are most useful in the boundary definition.

One important source of information which was untapped in this work is the data from neighboring images. These neighboring images may come from a third spatial dimension, as in a stack of images, or from a temporal dimension, as in neighboring frames within a movie. Using neighboring image planes would allow the generation of 3D input features.

The key result of this dissertation is showing the benefit available through the use of learned boundary representations. The framework is quite general, and other learning mechanisms could be used to learn the appropriate responses given the exemplars. Support vector machines, for example, may produce a learned representation much more quickly than the iterative error-backpropagation algorithm. This application could provide a platform for studying the efficiency and effectiveness of different learning mechanisms.

REFERENCES

[Airault, et al., 1994] Airault,S., Ruskone,R., and Jamet,O., "Road detection from aerial images: a cooperation between local and global methods", *Image and Signal Processing for Remote Sensing* (Proceedings of the SPIE, Vol #2315), pp. 508-18, 1994.

[Anderson, 1999] Anderson,C., "Semi-automated Boundary Tracing of Medical Images for Three-Dimenional Model Development", (CASI-TR-99-01), Final report of a CASI FY98 Technology Transfer Grant, *Colorado Advanced Software Institute*, CSU, Fort Collins, CO.

[Bajcsy and Tavakoli, 1976] Bajcsy,R., Tavakoli,M., "Computer Recognition of Roads from Satellite Pictures", *IEEE Transactions on Systems, Man, and Cybernetics*, v.SMC-6, #9, Sept. 1996, pp.623-637.

[Brahmi, et al., 1999] Brahmi,D., Cassoux,N., Serruys,C., Giron,A., LeHoang,P., and Fertil,B., "Correlation between variability of hand-outlined segmentation drawn by experts and local features of underlying image: A neuronal approach", Proceedings of SPIE, v 3647: *Proceedings of the 1999 Applications of Artificial Neural Networks in Image Processing IV*, January 1999, pp.164-172.

[Brahmi, et al., 2000] Brahmi,D., Serruys,C., Cassoux,N., Giron,A., Lehoang,P., and Fertil,B., "Segmentation of virus-infected areas in retinal angiograms using a learning-by-sample approach", *Proceedings of the International Joint Conference on Neural Networks*, July 2000, pp.158-162.

[Canny, 1986] Canny, J., "A Computationl Approach to Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v.8, #6, pp. 679–697, Nov. 1986.

[Carlbom, et al., 1994] Carlbom,I., Terzopoulos,D., and Harris,K., "Computer-Assisted Registration, Segmentation, and 3D Reconstruction from Images of Neuronal Tissue Sections", *IEEE Transactions on Medical Imaging*, v.13 #2, 1994, pp.351-362.

[Chellappa, 1992] Chellappa,R., ed., *Digital Image Processing*, IEEE Computer Society Press, Los Alamitos, CA, 1992.

[Cootes and Taylor, 2001] Cootes,T.F., and Taylor,C.J., "Statistical models of appearance for medical image analysis and computer vision", Proceedings of SPIE #4322: *Medical Imaging: Image Processing, 2001*, pp. 236-248.

[Cootes, et al., 1994] Cootes,T.F., Hill,A., Taylor,C.J., and Haslam,J., "The Use of Active Shape Models for Locating Structures in Medical Images", *Image and Vision Computing*, v.12 #6, July 1994, pp. 355-366."

[Crawford-Hines and McCracken, 2002] Crawford-Hines,S., and McCracken,T., "A System for 3D Surface Models from 2D Sectional Imagery", *Proceedings of the International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences* (METMBS'02), June 2002, v.I, pp.120-124.

[Crawford-Hines, 2000] Crawford-Hines,S., "Learning Expert Delineations in Biomedical Image Segmentation", *Intelligent Engineering Systems Through Artificial Neural Networks*, v.10, Proceedings of ANNIE 2000, Nov 2000, pp.657–662.

[Crawford-Hines and Anderson, 1997a] Crawford-Hines,S., and Anderson,C., "Using Visual System Preprocessing Models in Dynamic Learning of Boundary Contours", *Proceedings of the 2nd International Conference on Computational Intelligence and Neuroscience*, March 1997, pp.47–49.

[Crawford-Hines and Anderson, 1997b] Crawford-Hines,S., and Anderson,C., "Neural Nets Facilitate Boundary Tracing Tasks in Medical Images", *Neural Networks for Signal Processing VII (Proceedings of the 1997 IEEE Workshop)*, 1997, pp. 207-215.

[Crawford-Hines and Anderson, 1994] Crawford-Hines,S., and Anderson,C., "Interactive Region Bounding with Neural Nets", *NNACIP'94 -- International Workshop on Neural Nets Applied to Control and Image Processing*, November 1994, pp. 58-61.

[Dickson, et al., 1996] Dickson,S., Mackeown,W.P.J., Thomas,B.T., and Goddard,P., "Computer vision applied to the detection and localisation of acoustic neuromas from head MR images", *IEE Proceedings on Vision, Image and Signal Processing*, v.143 #4, August 1996, pp.265–272.

[Duda and Hart] Duda,R.O., and Hart,P.E., *Pattern Classification and Scene Analysis*, John Wiley and Sons, 1973.

[Durikovic, et al., 1998] Durikovic,R., Kaneda,K., and Yamashita,K., "Imaging and modelling from serial microscopic sections for the study of anatomy", *Medical and Biological Engineering and Computing*, v.36 #3, May 1998, pp. 276-284.

[Falcao, et al., 1998] Falcao,A.X., Udupa,J.K., Samarasekera,S., Sharma,S., Hirsch,B.E., and Lotufo,R.A., "User-steered image segmentation paradigms: Live wire and live lane", *Graphical Models and Image Processing*, v.60 #4, July 1998, pp. 233-260.

[Fenster and Kender, 2000a] Fenster,S.D., and Kender,J.R., "Training snakes to find object boundaries and evaluating them", Proceedings of SPIE, v4050: *Automatic Target Recognition X*, April 2000, pp. 245-258.

[Fenster and Kender, 2000b] Fenster,S.D., and Kender,J.R., "Comparative technique and performance results on novel learned snakes in two dissimilar medical domains", *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2000*, pp. 706-713.

[Freeman and Adelson, 1991] Freeman,W.T. and Adelson,E.H., "The Design and Use of Steerable Filters", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v.13 #9, September 1991, pp. 891-906.

[Gill, et al., 1999a] Gill,J.D., Ladak,H.M., Steinman,D.A., and Fenster,A., "Development and evaluation of a semi-automatic 3D segmentation technique of the carotid arteries from 3D ultrasound images", Proceedings of SPIE, v.3661,n.I: *SPIE Conference on Image Processing*, Feb 1999, pp. 214-221.

[Gill, et al., 1999b] Gill,J.D., Ladak,H., Steinman,D.A., and Fenster,A., "Accuracy of a semi-automatic technique for segmentation of the carotid arteries from 3D ultrasound images", *Proceedings of the International Conference of the IEEE Engineering in Medicine and Biology*, v.2, Oct. 1999, p.1146.

[Hamarneh, et al., 2001] Hamarneh,G., McInerney,T., and Terzopoulos,D., "Deformable Organisms for Automatic Medical Image Analysis", *Proceedings of Medical Image Computing and Computer-Assisted Intervention, MICCAI 2001*, Utrecht, The Netherlands, October 2001, pp. 66-75.

[Haralick 1985] Haralick,R.M., "Second Directional Derivative Zero Crossing Detector Using the Cubic Facet Model", *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition , CVPR 1985*, pp. 672-677

[Heinonen, et al., 1998] Heinonen,T., Dastidar,P., Eskola,H., Frey,H., Ryymin,P., and Laasonen,E., "Applicability of semi-automatic segmentation for volumetric

analysis of brain lesions", *Journal of Medical Engineering and Technology*, v.22, #4, Jul-Aug 1998, pp.173-178.

[Hohne , et al., 1996] Hohne, Pflesser, Pommert, Reimer, Schiemann, Schubert, and Tiede, "A 'Virtual Body' Model for Surgical Education and Rehearsal", *Computer*, v.29,#1, 1996, pp.25-31.

[Huertas and Medioni, 1986] Huertas,A. and Medioni,G., "Detection of Intensity Changes with Subpixel Accuracy Using Laplacian-Gaussian Masks", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v.8 #5, 1986, pp. 651–664.

[Huttenlocher, et al., 1993] Huttenlocher,D.P., et al., "Comparing Images Using the Hausdorff Distance", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v.14, #9, Sept 1993, pp.850-863.

[Hyde, et al., 1995] Hyde,S., Coppola,H., Rickler, and Weinberger, "Cerebral morphometric abnormalities in Tourette's syndrome: a quantitative MRI study of monozygotic twins", *Neurology*, 1995, 45:1176-82.

[Ivins and Porrill, 1994] Ivins,J., Porrill,J., "Active Region Models for Segmenting Medical Images", *Proceedings of the International Conference on Image Processing*, Nov. 1994, pp. 227-231.

[Johnson, et al., 1995] Johnson,C., MacLeod,R., and Schmidt,J., "Software Tools for Modeling, Computation, and Visualization in Medicine", *CompMed 94 Proceedings*, World Scientific, 1995.

[Kandel, et al., 1991] Kandel,E., Schwartz,J., and Jessell,T., *Principles of Neural Science*, Third edition, Appleton and Lange, Norwalk CT., 1991, Ch 28 and 29, pp.400-439.

[Karayiannisand Pai, 1999] Karayiannis,N.B., and Pai,P., "Segmentation of magnetic resonance images using fuzzy algorithms for learning vector quantization", *IEEE Transactions on Medical Imaging*, v.18, #2, 1999, p 172-180.

[Kass, et al., 1987] Kass,M., Witkin,A., Terzopoulos,D., "Snakes: Active Contour Models", *First International Conference on Computer Vision*, 1987, pp.259-268.

[Kim, et al., 1999] Kim,K.I., Jung,K., Park,S.H., Kim,H.J., "Supervised texture segmentation using support vector machines", *Electronics Letters*, v.35, #22, 1999, pp. 1935-1937

[Kohonen 1997] Kohonen,T., *Self-Organizing Maps*, Springer-Verlag, Berlin, 1997.

[Konishi and Yuille, 2000] Konishi, Scott, and Yuille,A.L., "Statistical cues for domain specific image segmentation with performance analysis", *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR '2000, p 125-132.

[Ladak, et al., 2000] Ladak,H.M., Mao,F., Wang,Y., Downey,D.B., Steinman,D.A., and Fenster,A., "Prostate segmentation from 2D ultrasound images", *Proceedings, 22nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Jul 2000, pp. 3188-3191.

[LeCun, 1997] LeCun,Y., "Neural Networks and Gradient-Based Learning in OCR", Invited lecture at Neural Networks for Signal Processing VII conference, September 1997.

[Malik and Perona, 1992] Malik,J., and Perona,P., "Finding Boundaries in Images", Neural Networks for Perception, Academic Press, 1992, pp.315-344.

[Mao, et al., 1999] Mao F., Gill,J., and Fenster,A., "Technique for evaluation of semi-automatic segmentation methods", Proceedings of SPIE, v.3661, n.II: *SPIE Conference on Image Processing*, Feb 1999, pp. 1027-1036,

[Marr and Hildreth, 1980] Marr,D., Hildreth,E., "Theory of Edge Detection", *Proceedings of the Royal Socety London*, v.B207, 1980, pp.187-217.

[McInerney and Terzopoulos, 1996] McInerney,T., Terzopoulos,D., "Deformable Models in Medical Image Analysis: A Survey," *Medical Image Analysis*, v.1 #2, 1996, pp. 91-108.

[McCracken, 2001] McCracken,T., "Complete Software for 3D Surface Modelling of Anatomy from 2D Sections", NSF Award #9901806, June 2001.

[McCracken, 1998] McCracken,T., "SBIR Phase I: Complete Software System for 3D Surface Modelling of Anatomy from 2D Sections", NSF Award #9761590, July 1998.

[McKeown and Delinger, 1988] McKeown,D., and Delinger,J., "Cooperative Methods for Road Tracking in Aerial Imagery", *Proceedings of Computer Vision and Pattern Recognition*, Ann Arbor, 5–9 June 1988, pp.662–672.

[Mortensen, 1999] Mortensen,E., technical exchange visit, April 2000.

[Mortensen and Barrett, 1998] Mortensen,E.N., and Barrett,W.A., "Interactive Segmentation with Intelligent Scissors", *Graphical Models and Image Processing*, v.60, 1998, pp.349-384.

[Mortensen and Barrett, 1995] Mortensen,E.N., and Barrett,W.A., "Intelligent Scissors for Image Composition", *Computer Graphics (SIGGRAPH '95)*, Aug. 1995, pp. 191–198.

[Mudry, et al., 2003] Mudry,K.M., Plonsey,R., Bronzino,J.D., *Biomedical Imaging*, CRC Press, Boca Raton FL, 2003.

[Nevatia and Babu, 1980] Nevatia,R., Babu,K.R., "Linear Feature Extraction and Description", *Computer Graphics and Image Processing*, v.13, July 1980, pp.257–269.

[Nitzberg, et al., 1993] Nitzberg,M., Mumford,D., and Shiota,T., *Filtering, Segmentation and Depth*, Springer-Verlag, Berlin, 1993.

[NLM, 2001] National Library of Medicine, "The Visible Human Project", as of Jan 2001: http://www.nlm.nih.gov/research/visible/visible_human.html

[Ozkan, et al., 1993] Ozkan,M., Dawant,B., and Maciunas,R., "Neural-Network-Based Segmentation of Multi-Modal Medical Images: A Comparative and Prospective Study", *IEEE Transactions on Medical Imaging*, v.12,#3, 1993, pp.534-554.

[Quam, 1978] Quam,L.H., "Road Tracking and Anomaly Detection in Aerial Imagery", *Image Understanding Workshop, Proceedings*, May 1978, pp.51-55.

[Raya, 1990] Raya,S., "Low-Level Segmentation of 3D Magnetic Resonance Brain Images - A Rule-Based System", *IEEE Transactions on Medical Imaging*, v.9,#3, 1990, pp.327-337.

[Reichenbach, et al., 1990] Reichenbach,S.E., Park,S.K., Alter-Gartenberg,R.A., "Optimal Small Kernels for Edge Detection", *Proceedings: 10th International Conference on Pattern Recognition*, Vol. II, Atlantic City, June 1990, pp. 57–63.

[Ripley, 1997] Ripley,B., in open discussion at the NATO Advanced Study Institute, *Generalization in Neural Networks and Machine Learning*, Cambridge, UK, August 1997.

[Rumelhart, et al., 1995] Rumelhart,D.E., Durbin,R., Golden,R., Chauvin,Y., "Backpropagation: The Basic Theory", Chapter 1 in *Backpropagation : Theory, Architectures, and Applications*, Lawrence Erlbaum Assoc, March 1995

[Schalkoff, et al., 1999] Schalkoff,R.J., Carver,A.E., Gurbuz,S., "Image segmentation using trainable fuzzy set classifiers", Proceedings of SPIE - v3716: *1999 SPIE Conference on Visual Information Processing VIII*, April 1999, pp. 9–17.

[Spitzer and Whitlock, 1998] Spitzer,V.M., and Whitlock,D.G., *Atlas of the Visible Human Male*, Jones and Bartlett Publishers, Inc., Sudbury, MA., 1998.

[Terzopoulos, 2002] Terzopoulos,D., personal communication, October 2002.

[Tesauro, 1995] Tesauro,G., "Temporal Difference Learning and TD-Gammon", *Communications of the ACM*, v.38 #3, March 1995, pp.58–68.

[Vasudevan, et al., 1988] Vasudevan,S., Cannon,R.L., Bezdek,J.C., "Heuristics for Intermediate Level Road Finding Algorithms", *Computer Vision, Graphics, and Image Processing*, v.44 #2, Nov 1988, pp. 175–90.

[Wang, et al., 1998] Wang,W., Wee,W., and Wang,X., "Volume Segmentation of the Visible Human CT Data Set", *The Second Visible Human Project Conference Proceedings*, 1998, published online as of April 2003 at `http://www.nlm.nih.gov/research/visible/vhpconf98/MAIN.HTM`

[Wang and Jenkin] Wang,Z., and Jenkin,M.R.M., "Using Complex Gabor Filters to Detect and Localize Edges and Bars", *Advances in Machine Vision*, World Scientific, 1992, pp. 151–170.

[Xu and Prince, 1997] Xu,C., and Prince,J.L., "Gradient Vector Flow: A New External Force for Snakes", *IEEE Proceedings of Conference on Computer Vision Pattern Recognition (CVPR'97)*, June 1997, pp. 66-71.

[Xu and Prince, 1998] Xu,C., and Prince,J.L., "Snakes, Shapes, and Gradient Vector Flow," *IEEE Transactions on Image Processing*, March 1998, pp. 359-369,

[Xu and Prince, 1999] Xu,C., and Prince,J.L., `http://iacl.ece.jhu.edu/projects/gvf/`

[Yue, et al., 1996] Yue,F.W., Cabestaing,F., Postaire,J.G., "An Analytical Comparison of the Performances of Two Edge Detectors", *1996 IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems*, 1996, vol.3, pp. 2254–8.

[Zhu and Yeh, 1986] Zhu,M., Yeh,P., "Automatic Road Network Detection on Aerial Photographs", *CVPR '86: Computer Vision and Pattern Recognition, Proceedings*, June 1986, pp. 34–40.