Semi-Automatic Medical Image Segmentation

by

Lauren O'Donnell

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

October 2001

Author .....
Department of Electrical Engineering and Computer Science
October 10, 2001

Certified by .....
W. Eric L. Grimson
Bernard Gordon Professor of Medical Engineering
Thesis Supervisor

Certified by .....
Carl-Fredrik Westin
Instructor in Radiology, Harvard Medical School
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Semi-Automatic Medical Image Segmentation

by

Lauren O'Donnell

Submitted to the Department of Electrical Engineering and
Computer Science
on October 10, 2001, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

We present a system for semi-automatic medical image segmentation based on the livewire paradigm. Livewire is an image-feature driven method that finds the optimal path between user-selected image locations, thus reducing the need to manually define the complete boundary. Standard features used by the wire to find boundaries include gray values and gradients.

We introduce an image feature based on local phase, which describes local edge symmetry independent of absolute gray value. Because phase is amplitude invariant, the measurements are robust with respect to smooth variations, such as bias field inhomogeneities present in all MR images. We have implemented both a traditional livewire system and one which utilizes the local phase feature. We have investigated the properties of local phase for segmenting medical images and evaluated the quality of segmentations of medical imagery performed manually and with both systems.

Thesis Supervisor: W. Eric L. Grimson
Title: Bernard Gordon Professor of Medical Engineering

Thesis Supervisor: Carl-Fredrik Westin
Title: Instructor in Radiology, Harvard Medical School

Acknowledgements

Thanks to everyone who used the segmentation system for their time, help, and feedback.

A sincere thank you to Dr. Martha Shenton's Schizophrenia Group at the Surgical Planning Laboratory of Brigham and Women's Hospital, for their trials of the system. Thanks to Ola Cizsewski for the caudate segmentations and her helpful collaboration in development of the system. Thank you also to Dr. Jim Levitt, Kiyoto Kasai, Toshiaki Onitsuka, and Natasha Gosek of the Schizophrenia Group.

Other members of the SPL were very helpful in development and evaluation of the system. Thanks to Dr. Ion-Florin Talos for the tumor segmentation, Arjan Welmers for feedback on the earliest versions of the system, Dr. Ying Wu for her ventricle segmentations, and Dr. Bonglin Chung for the pulmonary vein segmentations.

Thank you to Dr. Juan Ruiz-Alzola of the University of Las Palmas de Gran Canaria, Spain, for all of his help during preparation of the MICCAI paper based on this thesis. Thanks to the Radiation Oncology, Medical Physics, Neurosurgery, Urology, and Gastroenterology Departments of Hospital Doctor Negrin in Las Palmas de Gran Canaria, Spain, for their enthusiastic participation in evaluation of the system.

Special thanks to Mark Anderson for his experimentation with the software, feedback on the system, and help keeping the slicer running at the SPL.

Thanks to Dr. Mike Halle of the SPL for help getting started with VTK and a tremendous amount of disk space for slicer logs and segmentations.

Thanks to Dr. Ron Kikinis for feedback and many suggestions (some actually implemented) for improvement of the system.

Thank you to Dr. Eric Grimson for wonderful proofreading and input on the thesis,

as well as encouragement during my first two years here at MIT.

Finally, thanks to Dr. Carl-Fredrik Westin for explanations of local phase, great ideas, excitement about the project, and a tremendous amount of help.

During work on this thesis, the author was supported by a Rosenblith Fellowship and a National Science Foundation Graduate Fellowship.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Medical Image Segmentation

Medical image segmentation is the process of labeling each voxel in a medical image dataset to indicate its tissue type or anatomical structure. The labels that result from this process have a wide variety of applications in medical research and visualization. Segmentation is so prevalent that it is difficult to list the most oft-segmented areas, but a general list would include at least the following: the brain, the heart, the knee, the jaw, the spine, the pelvis, the liver, the prostate, and the blood vessels [20, 39, 35, 18, 25].

The input to a segmentation procedure is grayscale digital medical imagery, for example the result of a CT or MRI scan. The desired output, or "segmentation," contains the labels that classify the input grayscale voxels. Figure 1-1 is an example of a very detailed segmentation of the brain, along with the original grayscale imagery used to create the segmentation.

The purpose of segmentation is to provide richer information than that which exists in the original medical images alone. The collection of labels that is produced through segmentation is also called a "labelmap," which succinctly describes its function as a voxel-by-voxel guide to the original imagery. Frequently used to improve visualization of medical imagery and allow quantitative measurements of image structures, segmentations are also valuable in building anatomical atlases, researching

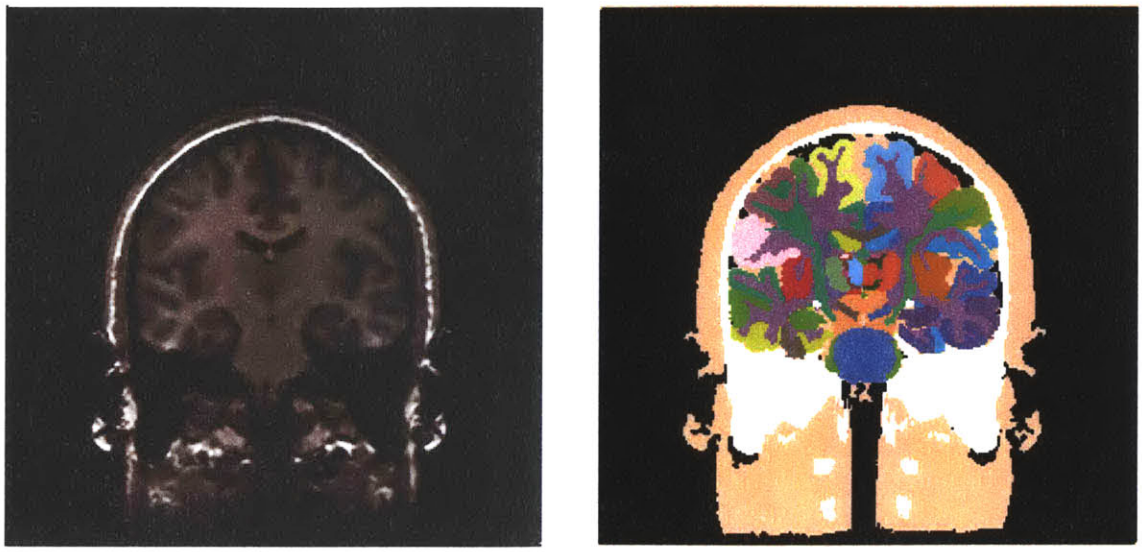


Figure 1-1: Segmentation example. A grayscale MR image of the brain (left) and a detailed matching segmentation, also known as a labelmap (right). The procedure followed to create the segmentation was partially automated, but a large amount of human effort was also required. The segmentation was initialized using an automatic gray matter/white matter/cerebrospinal fluid segmenter, and then individual neural structures were manually identified. This grayscale data set and segmentation were provided by Dr. Martha Shenton's Schizophrenia Research Group at the Surgical Planning Lab at Brigham and Women's Hospital.

shapes of anatomical structures, and tracking anatomical changes over time.

1.2 Applications of Segmentation

The classic method of medical image analysis, the inspection of two-dimensional grayscale images on a lightbox, is not sufficient for many applications. When detailed or quantitative information about the appearance, size, or shape of patient anatomy is desired, image segmentation is often the crucial first step. Applications of interest that depend on image segmentation include three-dimensional visualization, volumetric measurement, research into shape representation of anatomy, image-guided surgery, and detection of anatomical changes over time.

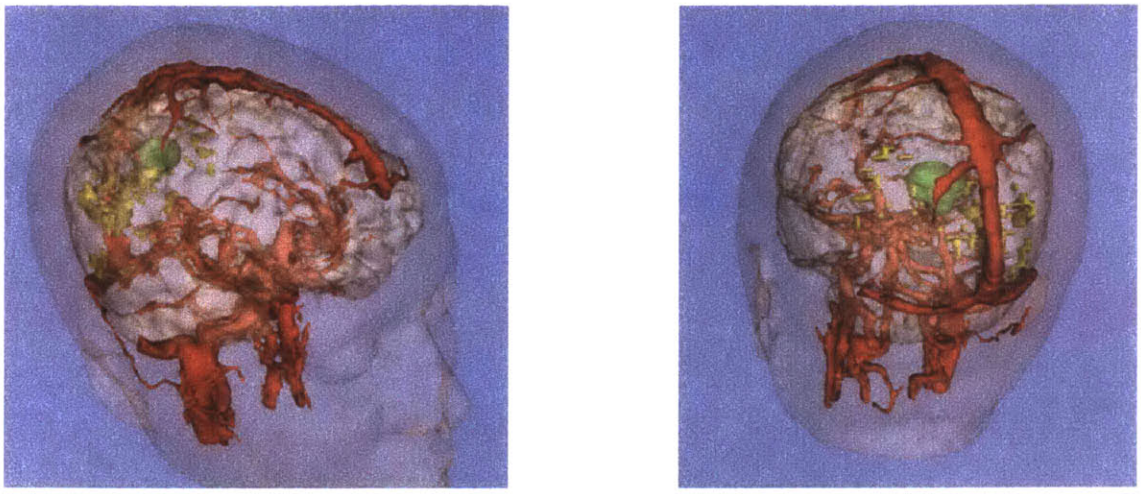


Figure 1-2: Example of three-dimensional surface models, created from segmented data using the Marching Cubes algorithm. These models were used in surgical planning and guidance. Each image is composed of five models: skin (light pink), neural cortex (light white), vessels (dark pink), tumor (green), and fMRI of the visual cortex (yellow). The fMRI, or functional MRI, shows areas of the brain that were activated during visual activities (areas which should be avoided during surgery).

1.2.1 Visualization

Segmentation of medical imagery allows the creation of three dimensional surface models, such as those in Figure 1-2, for visualization of patient anatomy. The advantage of a surface model representation of anatomy is that it gives a three-dimensional view from any angle, which is an improvement over two-dimensional cross sections through the original grayscale data [10]. Surface models can be created from segmented data using an algorithm such as Marching Cubes [26]. (Though three-dimensional models could be created directly from grayscale data using Marching Cubes, the segmentation step is used to provide the desired user-defined isosurfaces to the algorithm.)

1.2.2 Volumetric Measurement

Measurement of the volumes of anatomical structures is necessary in medical studies, both of normal anatomy and of various pathological conditions or disorders. This is an obvious application of segmentation, since it is not possible to accurately measure

anatomical volumes visually.

For example, in studies of schizophrenia, volume measurement is used to quantify the variation in neural anatomy between schizophrenic and control patients. Areas of interest in such studies include the lateral ventricles, structures in the temporal lobe such as the hippocampus, amygdala, and parahippocampal gyrus, the planum temporale, and the corpus callosum [27]. It is a time-intensive process to obtain accurate measurements of such regions, as the current method employs manual segmentation.

Volume measurement is also used to diagnose patients; one example is in measurement of the ejection fraction. This is the fraction of blood that is pumped out of the left ventricle of the heart at each beat, which is an indicator of the health of the heart and its pumping strength. To measure the ejection fraction, the blood in the left ventricle is segmented at different times in the cardiac cycle.

1.2.3 Shape Representation and Analysis

Various quantitative representations of shape are studied in order to mathematically describe salient anatomical characteristics. The first step in creating a representation of anatomical shape is segmentation: intuitively, one needs to know the structure's position and the location of its boundaries before its shape can be studied.

One example of a shape representation is a skeleton, a construct which is similar to the centerline of a segmented structure. One way to imagine a skeleton is the "brush fire" approach: one thinks of simultaneously lighting fires at all points on the boundary of the structure. The fires burn inward, traveling perpendicular to the boundary where they started, and then extinguish when they hit another fire. The connected "ash" lines left where the fires extinguish is the skeleton of the structure.

A richer shape representation is the distance transform, a function that measures the distance from each point in a structure to the nearest point on that structure's boundary. The distance transform can also be imagined with the pyrotechnic approach: it is the time that the fire first reaches each point in the structure. Consequently it is considered richer than the skeleton, since it contains more information.

Presumably, shape representations will become increasingly useful in making quan-

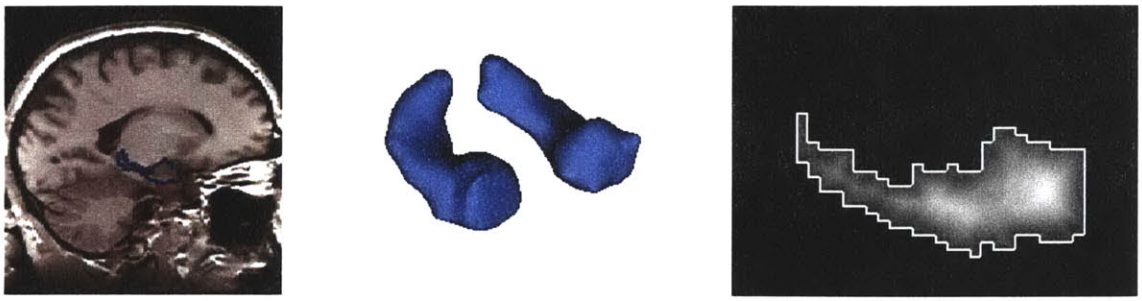


Figure 1-3: Shape representation example. A segmentation of the hippocampus-amygdala complex (left), a 3D surface model of the hippocampus-amygdala complex (center), and a distance map used to represent the shape of the hippocampus-amygdala complex (right). [12]

titative anatomical comparisons. Distance transform shape representations have already been applied to the classification of anatomical structures in a study that aims to differentiate between the hippocampus-amygdala complexes of schizophrenics and normals [12]. An example of grayscale MR image data and the shape representation derived from it for this study can be seen in Figure 1-3.

Shape representations can also be used to aid the segmentation process itself by providing anatomical knowledge [24]. A generative shape model, once trained from a population of shape representations, can then be used to visualize new shapes according to the learned modes of variance in the shape population (allowing visualization of "average" anatomy and of the main anatomical variations that may occur). Then, at each step of the segmentation of new data, fitting the model to the current most likely segmentation can provide anatomical information to the algorithm [24].

1.2.4 Image-Guided Surgery

Image-guided surgery is another medical application where segmentation is beneficial. In order to remove brain tumors or to perform difficult biopsies, surgeons must follow complex trajectories to avoid anatomical hazards such as blood vessels or functional brain areas. Before surgery, path planning and visualization is done using preoperative MR and/or CT scans along with three-dimensional surface models of the patient's

anatomy such as those in Figure 1-2.

During the procedure, the results of the preoperative segmentation may still be used: the surgeon has access to the pre-operative planning information, as three-dimensional models and grayscale data are displayed in the operating room. In addition, "on-the-fly" segmentation of realtime imagery generated during surgery has been used for quantitative monitoring of the progression of surgery in tumor resection and cryotherapy [33]. Figure 1-4 shows the use of preoperative surface models during a surgery.

1.2.5 Change Detection

When studying medical imagery acquired over time, segmenting regions of interest is crucial for quantitative comparisons. The Multiple Sclerosis Project at Brigham and Women's Hospital measures white matter abnormalities, or lesions, in the brains of patients suffering from MS. Because MS is a disorder that progresses over time, accurate temporal measurements of neural changes may lead to a better understanding of the disease. The stated goals of the MS project are analysis of lesion morphology and distribution in MS, quantitative evaluation of clinical drug trials, and monitoring of disease progression in individuals [16]. To this end, automatic segmentation is used to identify MS lesions, which appear as bright regions in T1- and T2-weighted MR scans of the brain, as shown in Figure 1-5. The volume of such lesions, as measured from segmented data, has been shown to correlate with clinical changes in ability and cognition [15, 14].

1.3 Difficulty of the Segmentation Problem

Two fundamental aspects of medical imagery make segmentation a difficult problem. The first aspect is the imaging process itself, and the second is the anatomy that is being imaged.

The imaging process, for example MR, CT, PET, or ultrasound, is chosen so that its interactions with the tissues of interest will provide clinically relevant information

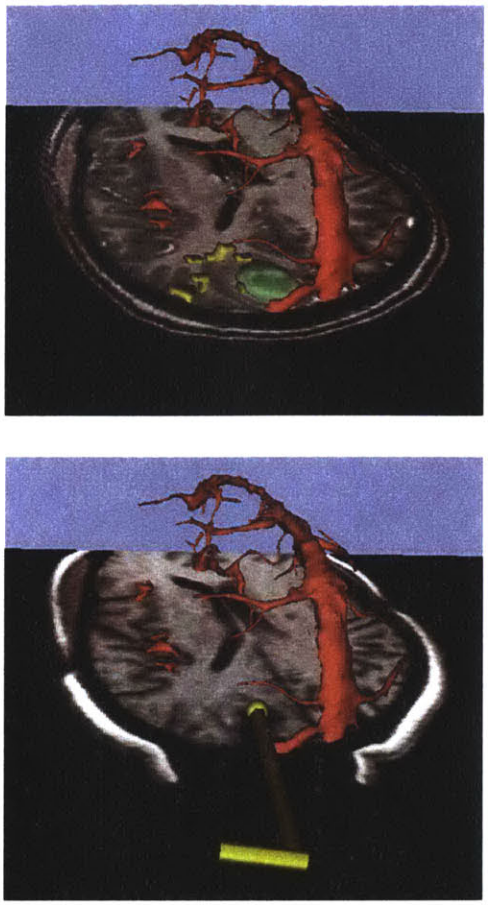
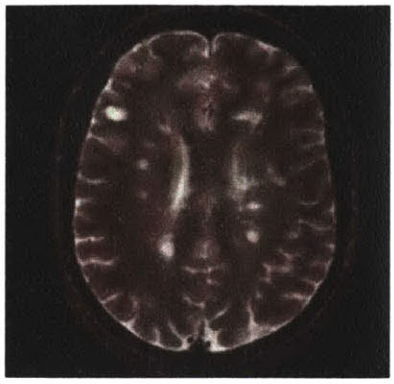
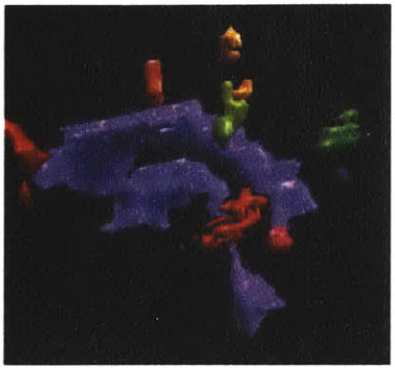


Figure 1-4: Surgical planning and navigation using surface models. The "before" picture (top) shows several 3D surface models used in surgical planning, along with one grayscale slice from the MR scan that was segmented to create the models. The green model is the tumor that was removed during the surgery.
The "after" picture (bottom) shows an image that was scanned during surgery, at the same location in the brain as the top image. The yellow probe is a graphical representation of the tracked probe held by the surgeon.



MS Lesions



Automatic Segmentation

Figure 1-5: Multiple Sclerosis Lesions (seen as bright spots). Automatic segmentation is used to track disease progression over time. Images were provided by Mark Anderson of the Multiple Sclerosis Group at the Surgical Planning Lab at Brigham and Women's Hospital.

about the tissue in the resulting output image. But this does not mean that the anatomical feature of interest will be particularly separable from its surroundings: it will not be a constant grayscale value, and strong edges may not be present around its borders. In fact, the interaction of the imaging process with the tissue of interest will often produce a "grainy" region that is more detectable by the human eye than by even sophisticated computer algorithms. (This is due to noise in the imaging process as well as to inhomogeneity of the tissue itself.) So simple image processing, such as thresholding or edge detection, is not generally successful when applied to medical image segmentation.

The second fundamental aspect that makes segmentation a difficult problem is the complexity and variability of the anatomy that is being imaged. It may not be possible to locate or delineate certain structures without detailed anatomical knowledge. (A computer does not approach the expert knowledge of a radiologist.) This makes general segmentation a difficult problem, as the knowledge must either be built into the system or provided by a human operator.

1.4 Our Method

In order to combine operator knowledge with computer aid, we chose to implement a semi-automatic segmentation method called livewire [1, 7, 8]. This type of segmentation tool was not previously available to our user community, the researchers who regularly perform manual segmentations at the Surgical Planning Lab at Brigham and Women's Hospital in Boston. To give an idea of the workload, it suffices to say that at any time during an average day at the Surgical Planning Lab, one can expect to find approximately ten people performing manual segmentations. In many cases, manual segmentation is used to complete a segmentation that was started with an automatic algorithm, and the manual segmentation can become the time bottleneck in the image processing pipeline.

Livewire is an image-feature driven method that finds an optimal path between user-selected image locations, thus reducing the need to manually define the complete boundary. To use livewire, one clicks on the boundary of interest in the image, and then, as the mouse is moved, a "live wire" curve snaps to the boundary, aiding manual segmentation. This interaction is formulated as a dynamic programming problem: the displayed live wire contour is the shortest path found between the user selected points, where the distance metric is based on image information.

The aim of the livewire approach is to reduce the time needed to segment while increasing the repeatability of the segmentation. We have implemented both a standard version of livewire, and a new version which employs a unique image feature. In our implementation, which we call "phasewire," we have investigated local phase as a feature for guiding the livewire [31]. Figure 1-6 shows several steps in performing a segmentation with our phase-based livewire. Models made from segmentations done with phasewire are shown in Figure 1-7.

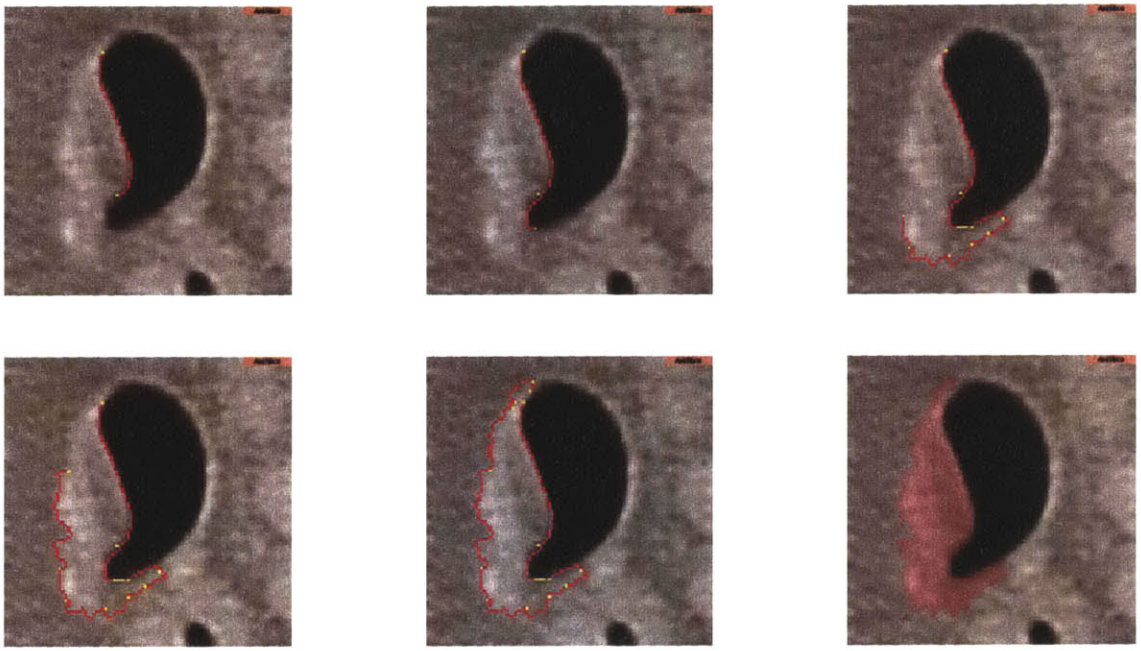


Figure 1-6: Example tumor segmentation. Steps taken during segmentation of a tumor (located next to the trachea, which is the dark region in the CT image). The segmentation begins with the upper left image, and the yellow pixels represent mouse clicks.

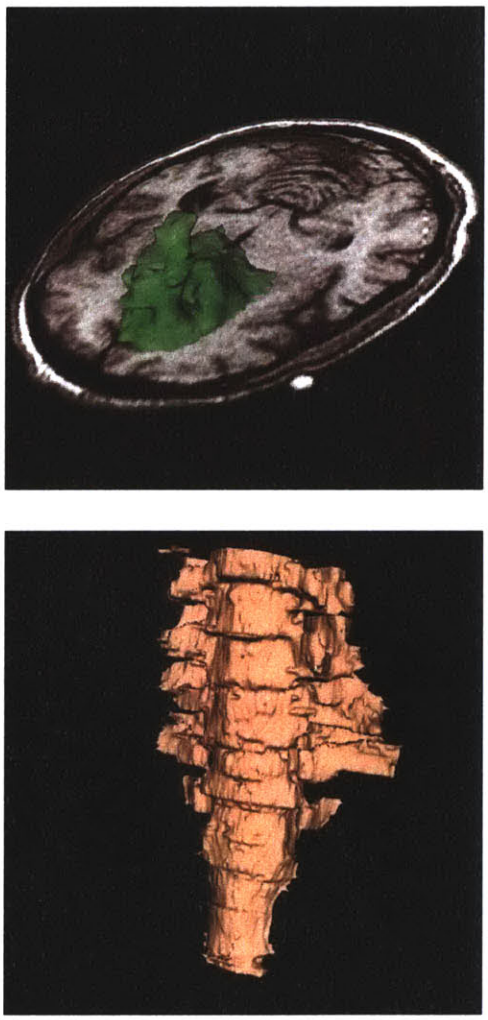


Figure 1-7: Surface models created from segmentations done with Phasewire. The top model is a tumor, while the bottom image shows a spine.

1.5 Roadmap

Chapter 2 begins with an overview of existing segmentation methods. Next, Chapter 3 gives an overview of the livewire algorithm, while Chapter 4 explains the concept of local phase. Then Chapter 5 describes our system in detail, and situates it in the framework introduced in the preceding chapters. Chapter 6 describes segmentation results from the method, with both user interaction measures and volumetric validation measures. Finally, Chapter 7 presents a discussion of the work.

Chapter 2

Background: Current Approaches to Medical Image Segmentation

In this chapter, we set the stage by describing some of the main methods of image segmentation in use today. We first describe the approaches in order according to the amount of knowledge they employ. (We will situate our algorithm in this order in Chapter 5.) Then we discuss factors that are important when rating a segmentation algorithm.

2.1 "Anatomical Knowledge Continuum" of Segmentation Algorithms

This section gives an algorithmic overview of the methods, situating each in a group with other algorithms that use similar knowledge to perform segmentation. A graphical summary of this section is in Figure 2-1, which attempts a visual presentation of the algorithms.

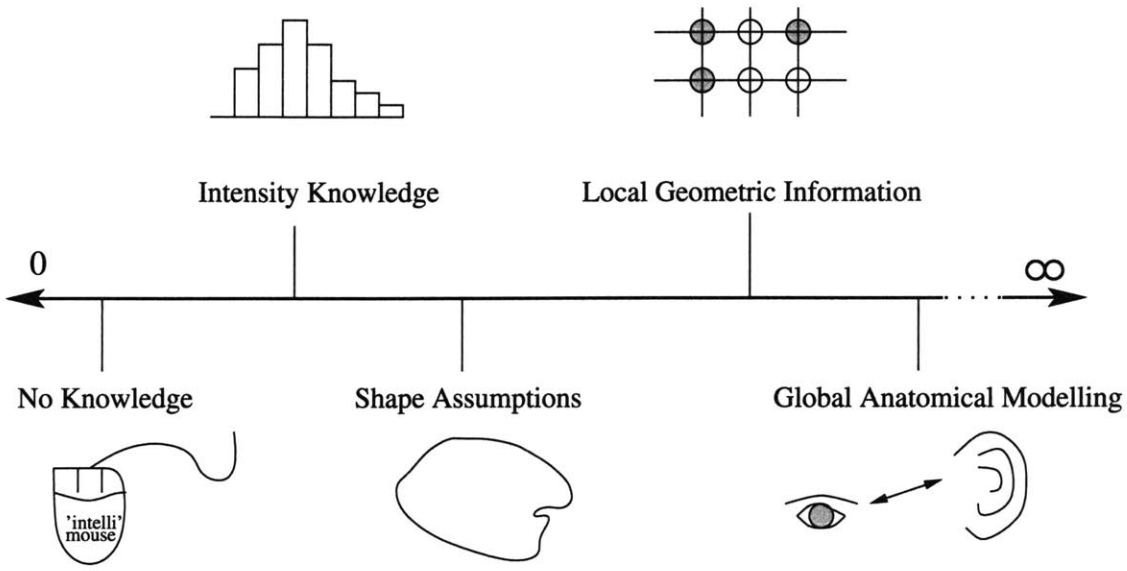


Figure 2-1: Segmentation algorithms as they lie on a continuum of anatomical knowledge, a graphical depiction. This figure places categories of segmentation algorithms on a scale according to the amount of knowledge they use. The left extreme side represents zero knowledge, while the far right right represents the ideal goal of infinite knowledge available to the algorithm.

2.1.1 No Knowledge Encoded in Algorithm

Manual Segmentation

The most basic segmentation algorithm, manual segmentation, consists of tracing around the region of interest by hand. This is done in each two-dimensional slice for the entire "stack" of slices that comprises a three-dimensional medical image volume.

The manual segmentation method is time-consuming and subject to variability across operators, for example up to 14 to 22 percent by volume in segmentation of brain tumors [19]. Consequently, manual segmentation is generally avoided if a comparable automatic method exists for the anatomical region of interest. However, despite its drawbacks, manual segmentation is frequently used since it provides complete user control and all necessary anatomical knowledge can be provided by the operator. The manual segmentation method may be the one chosen in regions where maximal anatomical knowledge is needed, such as when labeling cortical sulci and gyri, or segmenting hard-to-see regions such as the globus pallidus.

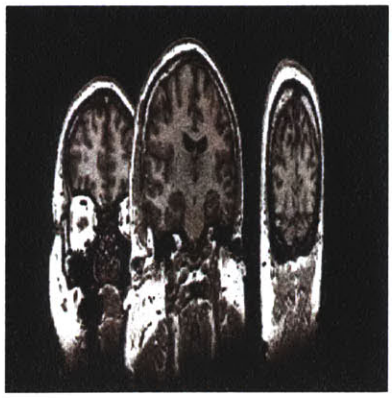


Figure 2-2: Slices from a medical image dataset. Only three slices were selected for display, but the entire image volume contains 124 slices, each 1.5 mm thick. Manual segmentation involves editing on each 2D slice that contains the structure of interest.

Simple Filtering

Morphological operations involve filtering a labelmap such that the boundary of a labeled region either grows (dilation) or shrinks (erosion). Sequences of morphological operations can augment manual segmentation by filling in small holes or breaking connections between regions.

Thresholding is another filtering method that is used to label voxels whose grayscale values are in a desired range. One of the simplest segmentation methods, thresholding can only be used when the grayscale values in the region of interest overlap very little with the grayscale values seen in the surrounding image.

2.1.2 Local Intensity Knowledge Used by Algorithm

In this section we describe algorithms that use local grayscale values in some way when performing segmentation. The algorithms placed in this category are quite dissimilar, but have been grouped together since their main source of information is grayscale values.

Livewire

Livewire is an image-feature driven method that finds the optimal path between user-selected image locations, thus reducing the need to manually define the complete boundary. This interaction is formulated as a dynamic programming problem: the displayed live wire contour is the shortest path found between the selected points, where the distance metric is based on image information. The image information used by implementations of livewire has included image gradients, Laplacian zero-crossings, and intensity values [8, 29].

Marching Cubes

In Marching Cubes segmentation, isosurfaces are found along a chosen grayscale value, essentially separating voxels of a higher value from voxels of a lower value [26]. This is accomplished by an algorithm that places cubes connecting the voxel centers, and if the isosurface lies in the cube, it decides what type of local surface polygon should pass through the cube. There are a limited number of possible surface topologies, which allows this local approach to rapidly construct a three-dimensional polygonal model. This type of segmentation is directly applicable to medical image data when the desired structure has very clear and constant boundaries, such as bone in a CT image.

2.1.3 Global Statistical Intensity Knowledge Used by Algorithm

In this section we describe algorithms that use global grayscale information when performing segmentation.

Expectation-Maximization (EM)

The EM algorithm is a method frequently used in machine learning to fit a mixture of Gaussians model to data. It is a two-step method that is applicable when only part of the data is observable. The first step, expectation or E-step, assumes that the

current Gaussian mixture model is correct and finds the probability that each data point belongs to each Gaussian. The second step, maximization or M-step, "moves" the Gaussians to maximize their likelihood (i.e. each Gaussian grabs the points that the E-step said probably belong to it).

In EM segmentation of the brain [36], the knowledge, or model, can be expressed as "there are three tissue classes, gray matter, white matter, and CSF," "tissue classes have a Gaussian intensity distribution" and "MR inhomogeneities are a multiplicative bias field."

As applied to image segmentation, the EM algorithm iteratively estimates the tissue class assignments of the voxels (in the E-step) and a multiplicative bias field (in the M-step). The output of the algorithm is a classification of voxels by tissue type and an estimate of the bias field. As the EM method uses prior knowledge of the number of tissue classes to segment, and can be initialized with a prior distribution on pixel class, it makes quite sophisticated use of grayscale information.

2.1.4 Global Shape Assumptions Used by Algorithm

In this section, "anatomical knowledge" refers to a global assumption made by the algorithm that influences the type of shapes that will be segmented. This is in addition to knowledge of grayscale values or gradients, etc., that may be used by the algorithm. The most-used shape assumption, which is useful for segmenting many types of medical data, is that anatomical structures will have smooth boundaries without high curvature.

In some cases this assumption may fail, such as in vessel segmentation where high curvature is expected around the vessel and low curvature along the vessel. The original assumption has been modified to allow higher curvature to remain around vessels, a modification which introduces more knowledge into the algorithm [25].

Segmentation algorithms using this type of shape knowledge are known as deformable models, and include snakes and level sets [5, 28, 23]. In these methods, the segmentation problem is formulated as an energy-minimization problem, where a curve evolves in the image until it reaches the lowest energy state.

These types of models subject the curve to external and internal forces which control the evolution. The forces represent the two types of knowledge used by the algorithm: grayscale and shape. The external forces are derived from image information, and use knowledge such as "high gradients are found along the border of the region to be segmented." These external forces are what cause the curve to stop evolving (for example when it reaches a high-gradient border in the image). The internal forces control the shape of the curve using the knowledge that "low curvature and slow changes in curvature are expected" [28]. For a curve that evolves inward, the basic internal force would cause it to have a speed proportional to its curvature, which would quickly smooth sharp curves while keeping low curvature regions the same.

Snakes

Snakes are two-dimensional deformable models, generally represented as parametrized curves, that evolve in the plane of a grayscale image. Manual curve initialization is followed by curve evolution until the snake settles into a low-energy state, at which point it is expected to enclose the structure of interest in the image.

Level Sets

In level set methods, the curve to be evolved is embedded into a higher dimensional space. Then the higher-dimensional representation is evolved instead of the original curve. The advantage of this approach is that topological changes of the original curve can be handled, since these changes do not complicate the higher-dimensional representation [5]. In addition, this algorithm extends to arbitrary dimensions.

For example, when segmenting a 2D image, the 2D curve is embedded in a 3D surface. The surface is generally initialized as the signed distance function from the curve, which means that points on the curve have value zero, points inside the curve are negative, and points outside are positive. These values can be thought of as the height of the surface above or below the image plane. Consequently, the surface intersects the image at the location of the curve. As the curve is at height 0, it is

called the zero level set of the surface.

The zero level set remains identified with the curve during evolution of the surface. The final result of segmentation is found by cutting the surface at height 0 to produce the output curve in the image.

2.1.5 Local Geometric Models Used by Algorithm

A geometric model describes relationships between structures in an image volume. This type of information is used locally to better classify voxels in an extension of EM segmentation, as follows.

Models of Local Relationships between Structures

One geometric model of relationships between anatomical structures is known as a distance model. This type of model encodes information such as "structure A lies about 10 mm above structure B." Such a model is created by choosing primary structures in a segmentation (structures easily identifiable by automatic, low-level image processing) and then mathematically describing their relationships to secondary structures, which are harder to detect automatically [18]. The fully-trained model may be used to create a statistical prior distribution on structure, expressing the likelihood that the voxel in a certain location will contain a certain secondary anatomical structure. This prior distribution can be used to initialize a statistical classifier, such as the EM-MF (Expectation Maximization-Markov Random Field) segmenter, a noise-insensitive variant of the EM segmentation algorithm [18].

2.1.6 Global Anatomical Models Used by Algorithm

Algorithms in this section use global anatomical information iteratively during execution to improve the classification of voxels.

Templates

A template is a global anatomical model, such as an anatomical atlas, which can be fitted to data to guide segmentation. The adaptive, template moderated, spatially varying statistical classification (ATM SVC) method warps an anatomical atlas to the data during segmentation [34]. Then it creates distance maps from each structure in the warped atlas: these maps basically give the likelihood that any voxel is part of a the structure. This information is used to spatially modify the statistical classification that is performed.

Shape Models

Another use of distance is in the creation of shape models. Shape models can be created from a training set of signed distance maps [24]. The models are derived from the data using Principal Component Analysis, or PCA, which finds the mean shape and the primary modes of variation of the shape. (In PCA, the training set is modeled as a high-dimensional Gaussian, where eigenvectors of the covariance matrix are the principal axes of variation, and eigenvalues indicate how much variation occurs along the axes [24]. These axes are also known as modes of variation.)

The signed distance map shape model is suited to integration into a level set segmentation method since the representation of the shape is the same as the surface representation in the algorithm. Both are signed distance maps, where the 0 level set (the set of all voxels with value 0) is a smooth surface that represents the current shape.

By adding another term to the energy-based evolution equation, the level set can be pulled toward toward the current best-fit shape that can be derived from the model. To influence the level set evolution, the location (pose) of the shape and the most likely shape parameters must be estimated at each iteration of the segmentation algorithm [24]. Then the surface evolution is encouraged to evolve toward this currently most likely shape.

2.2 Important Factors in Comparing Segmentation Methods

When evaluating segmentation methods for a particular application, the following factors are important.

- Running time and/or amount of user interaction
- Accuracy
- Reproducibility
- Generality/Applicability to the problem at hand

We will discuss these factors, with the aim of providing the necessary background for evaluation and comparison of our algorithm later in the thesis.

2.2.1 Segmentation Time

The overall time to complete a segmentation includes the running time of the algorithm and/or the time spent performing interactive segmentation. The algorithms described previously fall into four categories: automatic, automatic after initialization, semi-automatic, and manual as shown in Table 2.1. The semi-automatic algorithms are expected to take less time than a manual segmentation, which may take from minutes to days depending on the number of structures and slices. The running time of the automatic algorithms listed is on the order of seconds to hours, depending on the algorithm and the machine on which it is run.

2.2.2 Accuracy

The accuracy of a segmentation algorithm is generally evaluated by comparison with a manual segmentation, as there is no gold standard. Ideally, all methods should be evaluated for performance on data from a phantom or cadaver, but this is not practical. So the expert manual segmentation is compared with the output of the

Automatic	Requires Initialization	Semi-Automatic	Manual
Marching Cubes	Level Sets	Snakes	Manual Editing
ATM SVC		Livewire	
EM		Phasewire	
EM-MF			
Level Set and Shape Model			

Table 2.1: Division of segmentation algorithms by automaticity.

segmentation method, often with volumetric measures that do not address the main question of surface differences on the boundary of the segmentation.

The following measures could be used for evaluation of accuracy:

- Volume
- Histogramming by intensity of pixels on edge, just inside, and just outside of the boundary
- Overlap measures: fraction of pixels in structure A that are not also in structure B and vice versa, as well as fraction of the two surfaces that is common to both
- Histogramming of overlapping and non-overlapping pixels
- Bounds on distance between the segmented surfaces

2.2.3 Reproducibility

Reproducibility refers to the ability of an algorithm or operator to produce the same results more than once, or for different operators to produce the same result. This can be evaluated using the same measures as for accuracy, except that instead of comparing with a manual segmentation, comparison is done between segmentations that have been redone, or reproduced.

2.2.4 Generality and Applicability

Generality is considered useful in a segmentation method, and the introduction of more knowledge may limit generality by reducing the applicability of a method to various types of data. On the other hand, if an algorithm exists with beneficial knowledge of the specific problem at hand, it will be preferred for the application.

Chapter 3

Background: The Livewire Method

In this chapter, the Livewire method is explained in detail. We describe the steps of the algorithm and discuss existing implementations of livewire. Then in Chapter 5 we will situate our system in the framework introduced here.

3.1 The Steps of the Livewire Algorithm

The motivation behind the livewire algorithm is to provide the user with full control over a segmentation while having the computer do much of the detail work [1]. In this manner, the user's anatomical knowledge complements the ability of the computer to find boundaries of structures in an image.

Initially when using livewire, the user clicks to indicate a starting point, and then as the mouse is moved it trails a "live wire" behind it. When the user clicks again, this live wire freezes, and a new live wire starts from the clicked point. Images demonstrating user interaction during a segmentation were shown in Chapter 1 in Figure 1-6.

Here we first use an analogy to motivate introduction of the details of the algorithm, then we give an overview of its two steps. These steps are then discussed further in the remainder of the chapter.

In Livewire, the user draws with a "sticky contour" that snaps to boundaries in the image. This analogy with "stickiness" provides a perfect intuitive understanding

of the method. The "stickiness" of each pixel in the image is determined by measuring its local image characteristics. The stickiest pixels are those near a boundary between structures in the image. These pixels attract the livewire. Consequently, when the user selects points of interest for segmentation, the contour is drawn along the "stickiest" pixel path between them.

Continuing with the analogy, the livewire method has two steps which can be thought of as follows:

- "Stickiness" step: Image information is used to assign stickiness values to all points in the image.
- "Stickiest" step: A contour is drawn, along the stickiest possible path that connects the two points the user has chosen.

The livewire algorithm works by solving a dynamic programming problem, the search for shortest paths in a weighted graph. The two main parts of the livewire algorithm are the creation of a weighted graph using image information, and then the calculation and display of the shortest paths in the graph [7, 29]. Returning to the "stickiness" analogy, but adding the information that stickiness is really a cost value in the weighted graph, we have the following as the two steps in the livewire method:

- "Stickiness" step: Image information is used to assign costs to all points in the weighted graph.
- "Stickiest" step: The contour follows the shortest, or lowest-cost, path in the graph that connects the two points the user has chosen.

The first step employs image filtering to extract features of interest in the image, which are then passed through a cost function to produce graph weights. The livewire method is traditionally driven only by image information, with no explicit geometric constraint, so extracting knowledge from the image is of primary importance.

The second step uses Dijkstra's algorithm, a well-known shortest-path algorithm, to find shortest paths in the graph [6]. So each "sticky contour" drawn on the image by

the user is actually a shortest path, where the distance metric that defines "shortest" is based on image information.

Now we proceed with more details from each step.

3.2 Step One: Creation of the Weighted Graph

3.2.1 Graph Representation

The first step in the livewire process is the conversion of information from the image into a weighted graph. This allows (in step two) the application of Dijkstra's graph algorithm for finding shortest paths.

A weighted graph is a network of nodes connected by graph edges, where each edge has a weight associated with it. Figure 3-1 shows an example weighted graph. The weight, or cost, is the penalty for traveling from one node to another along the edge. The shortest path between any two nodes is defined as that path which has the lowest total cost (calculated by summing costs along all edges in the path). So, to go from the green node at the bottom to the red node in the upper right, there are two paths: one that goes directly, and another that travels through the blue node. The shortest is the direct one here (total cost of 4 as opposed to 15).

The graph is essentially overlaid on the grayscale image, and the user draws segmentation paths along the edges of the graph. So the graph edges define the possible segmentation boundaries in the image.

The purpose of the weights on the graph is to define the cost to segment (travel) from one image location (node) to another. Said another way, the weights are the image forces that attract the livewire. Desirable segmentation boundaries in the image should correspond to low weights in the graph: from the user's perspective, these desirable boundaries are "sticky."

The function of the weighted graph is now clear, but its physical location is not. The weighted graph can be thought of as a lattice that lies on top of the image, but how should it be aligned with the image? Should the nodes be pixels? Or should the

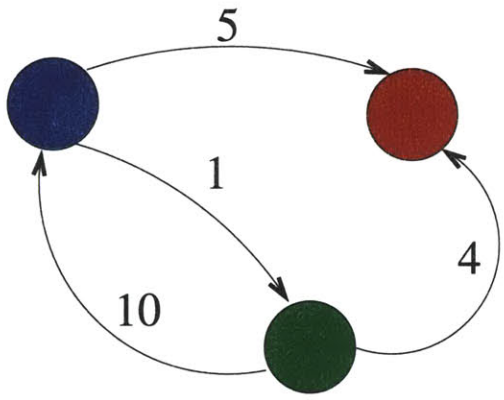


Figure 3-1: A weighted graph example. The nodes are the colored circles, while the edges are the arcs with arrows that connect the nodes. Each edge has a weight, or cost, number beside it. This is the cost for traveling along the edge in the direction indicated by the arrow.

edges be pixel edges? Either approach seems to make sense.

The problem of aligning the weighted graph with the image data has been approached in two ways in the literature. Both methods are shown in Figure 3-2. The first way is to place a node at each corner of each pixel, and have the graph edges be aligned with the "cracks" between pixels [7]. Our first standard livewire implementation used this method, and we encountered pros and cons as described in Table 3.1.

Pros	
Speed	Fast short paths since there are only 4 neighbors for each node.
Smoothness	Several (3) edges are required to enclose a protruding pixel.
Subpixel	Infinitesimally thin boundary.
Cons	
Interaction	User node selection (of pixel corners) is ambiguous.
Complexity	Shortest path encloses pixels in the structure.
Complexity	Must segment clockwise to know which pixels are in the structure.

Table 3.1: Pros and cons of situating graph nodes on pixel corners.

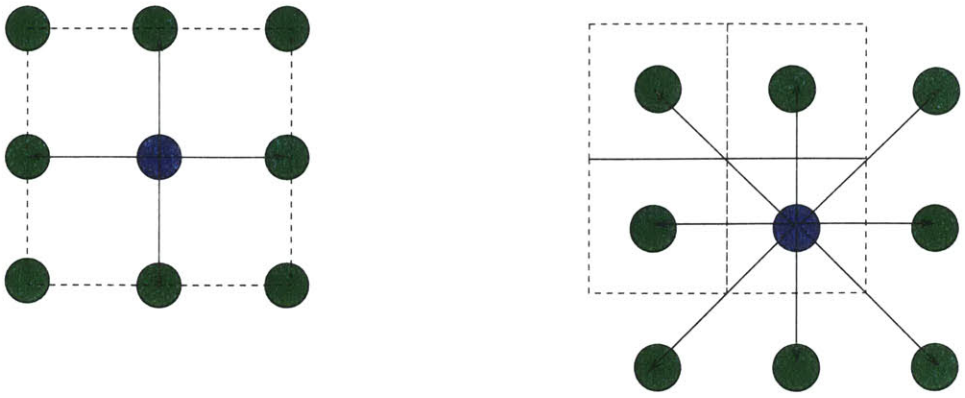


Figure 3-2: Two methods of aligning the weighted graph and the image. The dashed squares represent pixels in the image. The circles are graph nodes, and all edges leaving the blue center nodes have been drawn. On the left, the graph is aligned with the image such that the pixel corners are the graph nodes, and the edges separate pixels. On the right, the graph is aligned so that nodes are in the center of each pixel, and edges connect pixels.

The second manner of aligning the weighted graph with the images is to treat each pixel as a node in the graph, and consequently place the graph edges as connectors between pixels [29]. Table 3.2 lists the pros and cons of this method, as experienced with our implementation of phase-based livewire using this graph alignment.

In our application, the "path along pixels" method was found to be preferable to the "path along pixel edges" method, primarily due to the easier user interaction. In other words, choosing a pixel corner as a start point does not make much sense to users, and that became apparent in their comments regarding this implementation, such as "it was more difficult to handle the "tail" of the livewire."

However, it is important to note that when choosing a path along pixels, it may be harder to define the local boundary characteristics. This is because a "pixel crack" may better divide two dissimilar regions than a line of pixels, since the pixels are contained within one region or another. So in fact pixels that should belong to both regions may be chosen as part of the "boundary line" of pixels. Then when the boundary line is included in the final segmented shape, it may contain more pixels than desired.

To get a visual idea of the path that is followed in the image and graph, graph

Pros	
Interaction	It is clear which pixel is chosen by the user.
Simplicity	Labeling pixels for display of the path is straightforward.
Simplicity	The user may segment in any direction.
Speed	Eight neighbors per node doesn't significantly slow shortest paths.
Cons	
Smoothness	Only two edges are needed to connect an errant protruding pixel.
Speed	Filtering can be slower if it is done in eight orientations.
Accuracy	The boundary line may include pixels from both regions.

Table 3.2: Pros and cons of situating graph nodes on pixel centers.

edges have been drawn over a segmentation in progress in Figure 3-3. This figure shows the eight-connected version where the path travels along the pixels in the image. At the highlighted yellow endpoint, the possible future directions that the path may take (all outward-traveling graph edges from the yellow node) are shown with blue arrows.

3.2.2 Local Image Features

The weights on each edge in the weighted graph are derived from some combination of image features, where an image "feature" is any useful information that can be obtained from the image. In other implementations, these features have included image gradients, Laplacian binary zero-crossings, and intensity values [8, 29]. Generally, the features are computed in a neighborhood around each graph edge, transformed to give low edge costs to more desireable feature values, and then locally combined in some user-adjustable fashion.

The purpose of the combined features is edge localization, but individual features are generally chosen for their contributions in three main areas, which we will call "edge detection," "directionality," and "training." We give here an overview of these

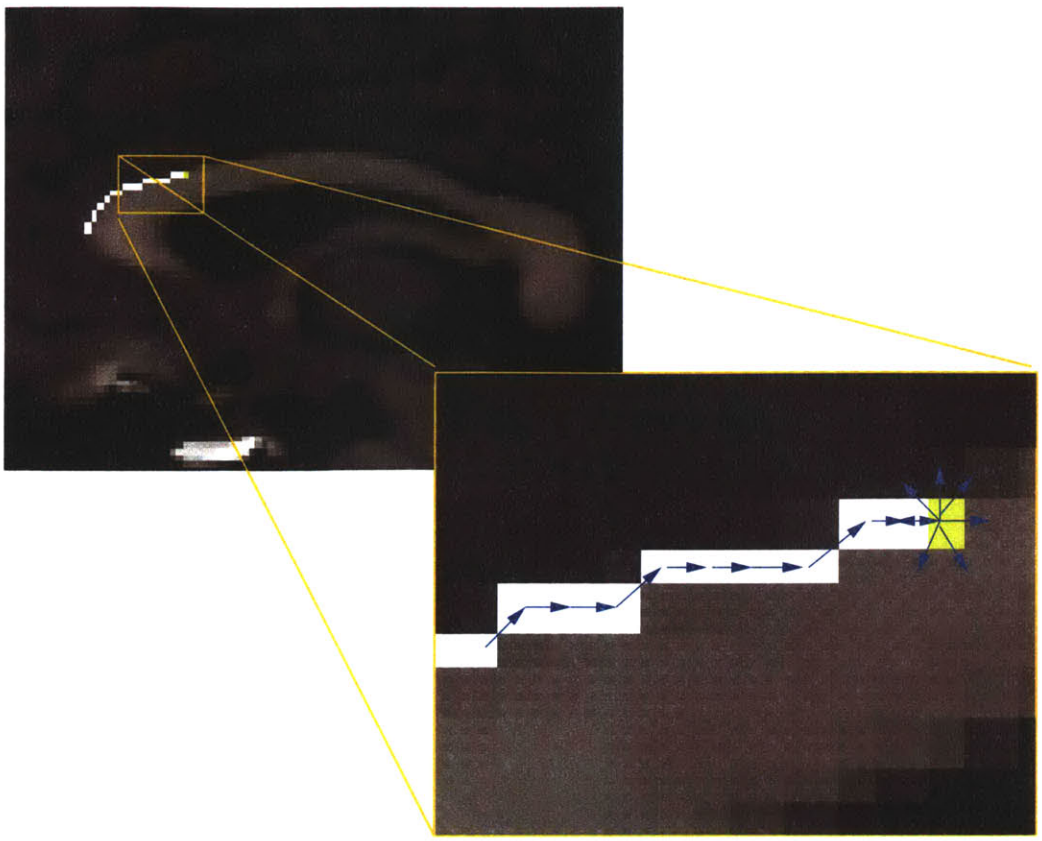


Figure 3-3: Livewire shortest path. The arrows represent the graph edges followed when calculating this path.

areas, and then later we discuss the specifics of two implementations of livewire-style segmentation tools.

Edge Detection

Edge detection is fundamental in the livewire segmentation process. The gradient [8] and the Laplacian zero-crossing [30], as well our new feature, phase, are primarily useful in edge detection. (More details can be found in Section 3.2.3.) These features are the main attractors for the livewire.

Directionality

The overall summing of costs along the path will prevent most incorrect local direction choices, since they will lead to higher path cost in the end. However, this may not

be enough to prevent small incorrect steps, especially in the presence of noise or near other attractive structures.

Various features can influence which direction the path should take locally. Directional gradients computed with oriented kernels [8], or the "gradient direction" feature based on the angle between each potential edge direction and the local gradients [30], provide local direction information. Details of the computation of these features can be found in Section 3.2.3.

An interesting ability of the livewire, if the direction of segmentation is known (i.e. clockwise), is to distinguish between inside and outside pixels [8]. Consequently, instead of being attracted by all areas with gradient of X, the livewire can pay attention only to those regions with such a gradient where the inside is, for example, brighter than the outside. This ability of the livewire to orient itself is useful, especially in avoiding similar boundaries with the reversed sense, though the directional restriction on segmentation can be confusing to users.

Training

This is the process by which the user indicates a preference for a certain type of boundary, and the features are transformed accordingly to give low graph edge costs to the regions preferred by the user. Image gradients and intensity values [8], as well as gradient magnitudes [30], are useful in training.

3.2.3 Local Image Features: Implementation Details for Two Methods

Descriptions of computing image features from two published methods follow.

Oriented Boundary Method

The implementation that situates nodes on pixel corners (finds paths along "pixel cracks") defines the neighborhood shown in Figure 3-4 [8]. This neighborhood is used to compute features relevant to the dark graph edge in the center of the two

colored pixels. Note that this neighborhood is rotated for all possible orientations and directions of the graph edge, and the features are computed with each orientation.

The following features are defined in the local neighborhood, where the letters refer to the intensities of the pixels as labeled in Figure 3-4:

- Intensity on the "positive" side of the edge, where positive is defined relative to the local gradient direction. This is p or q.
- Intensity on the "negative" side of the edge, where negative is defined relative to the local gradient direction. This is p or q.
- Various gradient magnitude features, including:
  - $|p - q|$
  - $\frac{1}{3}|p + t + v - q - u - w|$
  - $\frac{1}{4}(|p - u| + |t - q| + |p - w| + |v - q|)$
- Orientation-sensitive gradient magnitude, which basically multiplies the gradient magnitude defined above by -1 if the local gradient orientation does not match the neighborhood orientation.
- Distance from boundary traced on the previous slice.

The advantage of the preceding definitions is that they are selective for boundary orientation. This means that the algorithm can tell the difference between a boundary with darker pixels inside than outside, and the reverse: a similar boundary with light pixels inside and dark outside. This, however, assumes that the user is consistently tracing either clockwise or counterclockwise.

Paths Along Pixels Method

This method is called "Intelligent Scissors," and it aligns the graph such that the pixels are nodes in the graph. The image features used in this implementation follow [29].

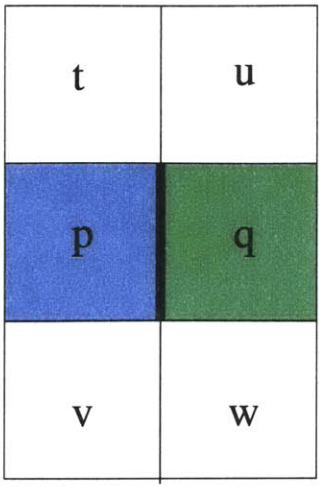


Figure 3-4: One local neighborhood for computing livewire features [8]. Each square represents a pixel, and in this system the graph edges are the "cracks" between pixels. So the dark line separating the blue and green pixels is an edge in the weighted graph.

- Laplacian Zero-Crossing: This binary feature is created by convolution with a Laplacian kernel, and then for all neighboring pixels with opposite signs, the pixel closest to 0 becomes the zero-crossing.
- Gradient Magnitude: $G = \sqrt{I_x^2 + I_y^2}$
- Gradient Direction: Smoothness constraint that says that the link between two pixels should run perpendicular to the gradient directions at both pixels. This involves a dot product at each of the two connected pixels, between the unit vector perpendicular to the local gradient and the unit vector representing the direction of the graph edge.

The advantage of this formulation is that it includes an explicit directional smoothness constraint. But it does not differentiate between boundary orientations.

3.2.4 Combination of Features to Produce Edge Weights

The two goals in feature combination are to emphasize desired features and to control the input to Dijkstra's algorithm. The version of Dijkstra's algorithm that we implemented needs bounded input values, and in general Dijkstra's algorithm cannot

handle negative edge weights, so the conversion from features to edge weights needs to handle these criteria.

Several methods in the literature have been used to transform features into edge weights. One is to transform each feature using a function, for example a Gaussian, whose parameters are set such that desired feature values are converted into small values [8]. These output values are then combined in a weighted sum, where the weights control the influence of each feature on the graph weights.

Another method is simple scaling, such that the feature image is converted to an edge cost image whose values are bounded. Other variations are used, for example an inverse linear ramp function will emphasize low feature values, and a lookup table can be used to apply arbitrary conversions between feature values and edge costs [30].

We investigated Gaussian models, simple scaling, inversion of feature values, and a lookup table for combining feature values in our implementations of livewire.

3.3 Step Two: Shortest Paths

In the second part of the livewire algorithm, shortest paths are found in the weighted graph. The livewire is defined as the shortest path that connects the two user-selected points (the last clicked point and the current mouse location). This second step is done interactively so the user may view and judge potential paths, and control the segmentation as finely as desired.

3.3.1 Dijkstra's Algorithm

Dijkstra's algorithm is used to find all shortest paths extending outward from the starting point [6]. In the standard Dijkstra's algorithm, shortest paths to all nodes from an initial node are found. The algorithm works by finding paths in order of increasing path length, until all shortest paths have been found. Execution of the algorithm spreads out like a wavefront from the start point, looking at neighbor nodes of those nodes whose path has already been found.

Each node goes through two steps in the algorithm: first it is examined, as the

neighbor of a node whose shortest path has been found. It is then saved in a list of nodes that are potential candidates for the "next shortest path." Also, its current path length and the neighbor it was found from are saved.

Then in the second step, the "next shortest path" (in increasing order of path lengths) is found. The node waiting in the list (call this node n) that has the shortest path length to the original point is selected, and this path becomes its shortest path. Node n is now done and is removed from the list. Then its neighbors are examined, and the path to reach them is updated if a shorter path exists through node n.

Because of the manner of execution of the algorithm, and the fact that all shortest paths are composed of subpaths that are also shortest paths, the path found for node n in the second step is guaranteed to be optimal (shortest).

A downside of this algorithm in the context of image segmentation is that shortest paths are found from the original point to all points in the image. This does not make much sense, as the user is highly unlikely to segment, for example, from the center of the image to a corner in one step. So using the above algorithm as is would be computationally wasteful.

3.3.2 Live Wire on the Fly

To reduce the amount of computation, first the exact amount of computation that is necessary for each interaction with the user must be defined. The interaction with the user during drawing of one path segment consists of one mouse click to indicate the start point, followed by mouse movement indicating a set of endpoints, and finally another click to indicate the final choice of endpoint. So ideally, computation should be performed to find the shortest path from the original point to the first endpoint, and longer shortest paths should be computed only as needed to reach additional endpoints. (Note that since paths are found in order of increasing path length, in order to find a path of length L to the current endpoint, all paths of length $< L$ must be found first.) This also should be interactive: each path from start to current endpoint must be displayed until the mouse moves again, requesting another endpoint.

An existing algorithm called "Live Wire on the Fly," does this: it computes the

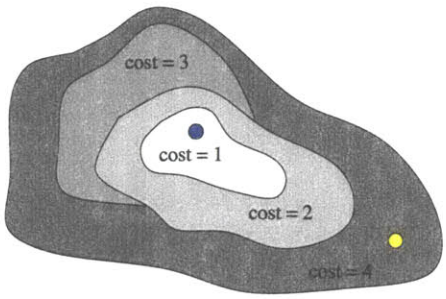


Figure 3-5: Shortest paths computed until reaching the endpoint. Imagine that this diagram overlays a medical image. The diagram represents the action of the algorithm as shortest paths are computed in a region that "spreads outward" from a start point (blue) toward an endpoint (yellow). Before reaching the endpoint, all paths of cost less than 4 must be computed, and multiple paths of cost 4 may be computed before the endpoint is found. The information about the paths in the gray area is saved for the next interaction with the user, in Figure 3-6.

shortest paths for only as much of the image as necessary [7]. It uses a modified version of Dijkstra's algorithm that stores nodes currently under investigation in a sorted circular queue, and is able to stop executing when it reaches the user-defined endpoint [7]. Computed shortest path information is retained and re-used when the user moves the mouse again to choose another endpoint. This is possible because of the classic characteristic of Dijkstra's algorithm, that each shortest path is comprised of prior shortest paths. Figures 3-5 and 3-6 are stylized depictions of the algorithm in progress, displaying two steps in shortest path calculation and demonstrating the caching of already-computed information.

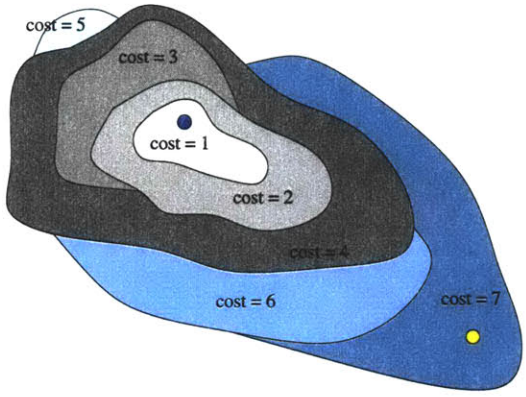


Figure 3-6: Shortest paths computed using prior stored information. When the mouse is moved to the new yellow endpoint, the algorithm starts computing remaining paths of cost 4, based on the cached information from the gray region, and it will compute paths in the blue area until reaching the endpoint. When the endpoint is reached, all paths of costs 5 and 6, and some of cost 7, will have been computed.

Chapter 4

Background: Local Phase

In this chapter, we give a theoretical overview of the concept of local phase. This provides the background for our application of local phase to user-guided image segmentation, which is described in Chapter 5.

4.1 Fourier Phase

The Fourier transform of a signal can be a complex signal. Thinking of the Fourier transform as a decomposition of the signal into sinusoids, the magnitude of the complex signal gives the amplitude of the sinusoid of each frequency, while the argument of the signal, the phase, describes the spatial (or time) shift the sinusoid undergoes.

Signals that are evenly symmetric about the origin will have real Fourier transforms, while signals that have an odd symmetry will have imaginary Fourier transforms. So the cosine has a real Fourier transform, which means that the argument, or phase spectrum, must be 0 (or π) for all frequencies. Similarly, the sine has a purely imaginary Fourier transform and a phase of $\pi/2$ (note it is a cosine phase-shifted by 90 degrees). Sinusoids that are neither perfectly odd nor even will have Fourier transforms that have both real and imaginary parts, and their phase will describe their symmetry about the origin (for example, more odd than even, or vice versa).

Figure 4-1 motivates the introduction of local phase with plots of two simple sinusoids that exhibit different symmetries about the origin, the sine and the cosine.

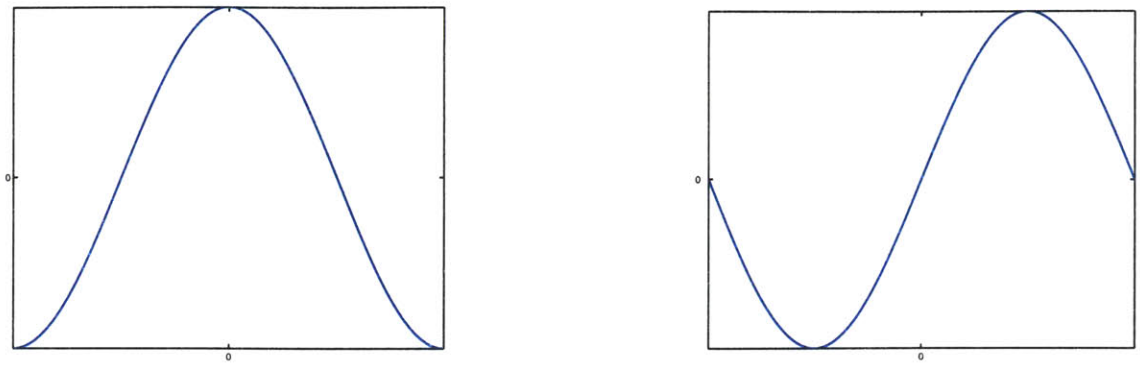


Figure 4-1: Basic sinusoids: the sine and cosine. The sine is a 90-degree phase-shifted version of the cosine. About the origin, the cosine is evenly symmetric while the sine has an odd symmetry.

4.2 Introduction to Local Phase

The local phase aims to mimic the behavior of the Fourier phase of a signal, localized by a spatial windowing function. The similarity between local and global phase should not be taken too far since they are very different concepts. The Fourier phase describes spatial relations globally, and the the local phase gives a symmetry statement about the signal in a certain position. However, for a simple signal such as a sinusoid of a certain frequency, the local phase coincides with the Fourier phase.

The local phase of a sinusoid is shown in Figure 4-2. At each point, the local phase value is equivalent to the Fourier phase that would be calculated if that point were the origin.

We will continue describing the local phase as follows. First we will define the local phase for one-dimensional signals, then we describe an extension to two-dimensional signals. We then motivate this extension by giving an example of the local phase of a synthetic image. Finally, we summarize the advantages of local phase for image segmentation.

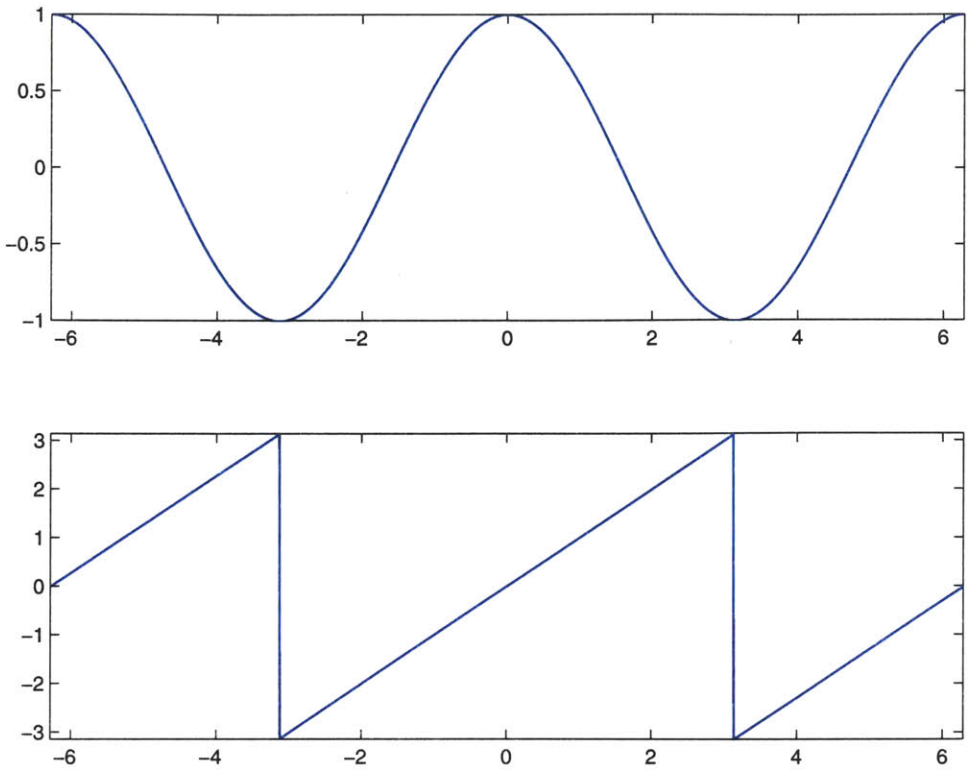


Figure 4-2: Local phase of a sinusoid.

4.3 Local Phase of a One-Dimensional Signal

4.3.1 Definition of Local Phase

The local phase as used in this project is a multidimensional generalization of the concept of instantaneous phase, which is formally defined in one dimension as the argument of the analytic function

$$f_A(x) = f(x) - if_{Hi}(x) \tag{4.1}$$

where f_{Hi} denotes the Hilbert transform of f [2, 3].

The Hilbert transform in one dimension is a convolution, defined as

$$f_{Hi}(x) = f(x) * -\frac{1}{\pi x} \tag{4.2}$$

[13].

So in the Fourier domain the Hilbert transform becomes a multiplication operation, where the sign function means a multiplication by the sign of the frequency, which is either positive or negative one.

$$F_{Hi}(u) = F(u) \times i \times sign(u) \tag{4.3}$$

[13]

An analytic function, by definition, has no negative frequencies in the Fourier domain. The formula for the analytic function in the Fourier domain is clearly just a removal of negative frequencies:

$$F_A(u) = F(u) - iF_{Hi}(u) = F(u)[1 + sign(u)] \tag{4.4}$$

This has the effect of doubling the positive frequencies and canceling out the negative ones. This operation is not as drastic as it appears, since for a real input signal, "chopping off" the negative frequencies does not destroy any information. This is because for signals that are purely real in the spatial domain each half of the frequency space contains all of the signal information [13].

4.3.2 Phase Measurement in One Dimension

The analytic function, as the name implies, is useful in the analysis of the original signal. From the analytic function one can robustly measure the instantaneous amplitude, phase, and frequency of the original signal [13].

- The instantaneous amplitude is measured as the amplitude of the analytic signal.
- The instantaneous phase is measured as the argument of the analytic signal.
- The instantaneous frequency is the rate of change of the instantaneous phase angle.

For a pure sinusoid, the instantaneous amplitude and frequency are constant, and are equal to the amplitude and frequency of the sinusoid. This is interesting because measuring the amplitude A of the original sinusoid will give any number between $-A$ and A, while measuring the amplitude of the analytic signal will give the true amplitude of the original signal. The instantaneous phase of the pure sinusoid, on the other hand, is not constant: it varies as the local phase angle changes over each period of the sinusoid. This gives a sawtooth signal as shown in Figure 4-2.

The instantaneous phase of a simple one-dimensional signal is shown in Figure 4-3. This simple signal can be thought of as the intensity along a row of pixels in an image. Note that in this "image" there are two edges, or regions where the signal shape is locally odd, and the phase curve reacts to each, giving phase values of $\pi/2$ and $-\pi/2$. Our system uses these phase values to indicate the presence of anatomical boundaries, or desireable contours for the livewire to follow.

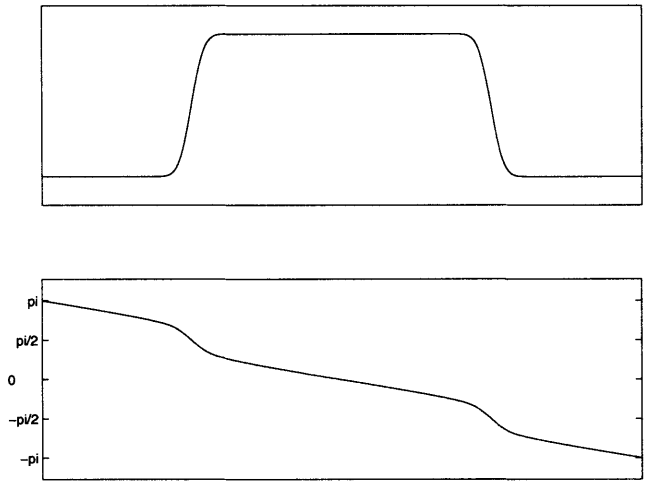


Figure 4-3: A simple signal (top), and its instantaneous phase (bottom).

This example also shows the global aspects of the local phase: since it is measured within a window, it is affected by nearby variations in the signal. Consequently the effect of the edges in the signal in Figure 4-3 is seen also in the neighborhood of the edge, giving the phase a smooth slope throughout.

4.4 Local Phase of a Two-Dimensional Signal

The extension of the above definition of local phase to two or higher dimensions is not straightforward, because there is no clear definition of negative frequency in more than one dimension. We describe the two-dimensional case here.

In two dimensions, a reference direction may be chosen in the Fourier plane, and the local phase can be computed along that direction exactly as described for one dimensional signals. This approach, however, will not suffice because we are interested in a statement about the overall symmetry that is not limited to only one direction.

To provide rotational invariance, the oriented filter kernels used in estimation of local phase have a cos^2 angular function assuring even coverage to the Fourier plane when summed. This allows computation of the phase in any direction and in addition, estimation of the local orientation of image features, with a fixed number of filter kernels [22, 17, 13].

Alternative methods to estimate local phase in one dimension include for example using Gabor filters or first and second derivative Gaussian filters [37]. Extending these methods to higher dimensions, although conceivable, is not straightforward. Challenges include optimal cancellation of negative frequencies (since the odd and even parts of these filters are not related through the Hilbert transform), and obtaining rotational invariance (which can be problematic since the angular filter shape is dictated by the filter creation process).

4.4.1 Interpreting the Local Phase of a Two-Dimensional Image

The local phase of an image describes how even or odd the signal is in a window around each pixel. Consequently, structures of interest such as lines and edges in the image can be detected and differentiated. The local phase of lines and edges can be visualized on a unit circle where the prototypical structures of phase 0, $\frac{1}{2}\pi$, π, and $\frac{3}{2}\pi$ are drawn alongside the circle [13]. This representation of phase is shown in

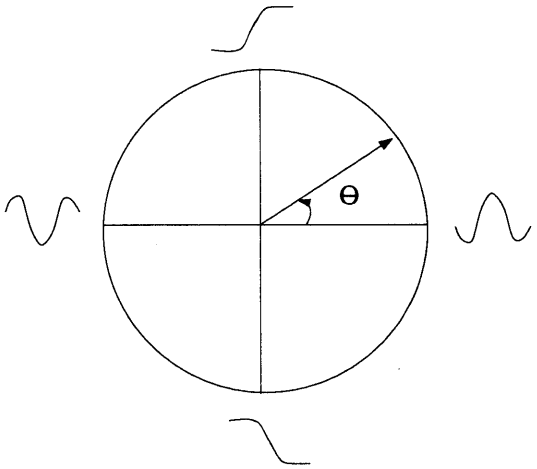


Figure 4-4: Local phase on the unit circle [13]. Note that any combination of line and edge, light or dark, can be represented as an angle.

Figure 4-4, where $\frac{1}{2}\pi$ and $\frac{3}{2}\pi$ represent edges of different orientations, while 0 and π represent dark or light lines. Intermediate phase values give the local balance between the closest two prototype structures.

Edges, or transitions between light and dark in the image, are odd functions, so they are expected to have phase that is a multiple of $\frac{1}{2}\pi$. Conversely, lines, being even functions, should have phase values near 0 or π. Figure 4-5 shows the local phase of a synthetic example image. It is clear from the figure that local phase is a very reasonable edge detector, which is a desirable property in guidance of interactive medical image segmentation.

4.5 Advantages of Local Phase

The local phase has several advantages in the context of image segmentation:

- The local phase is invariant to signal intensity. (This is because by definition the phase and magnitude of a complex number are separate, so the magnitude does not affect the phase.) Consequently, lines and edges can be detected from small or large signal variations.

- The local phase generally varies smoothly with the signal [13].

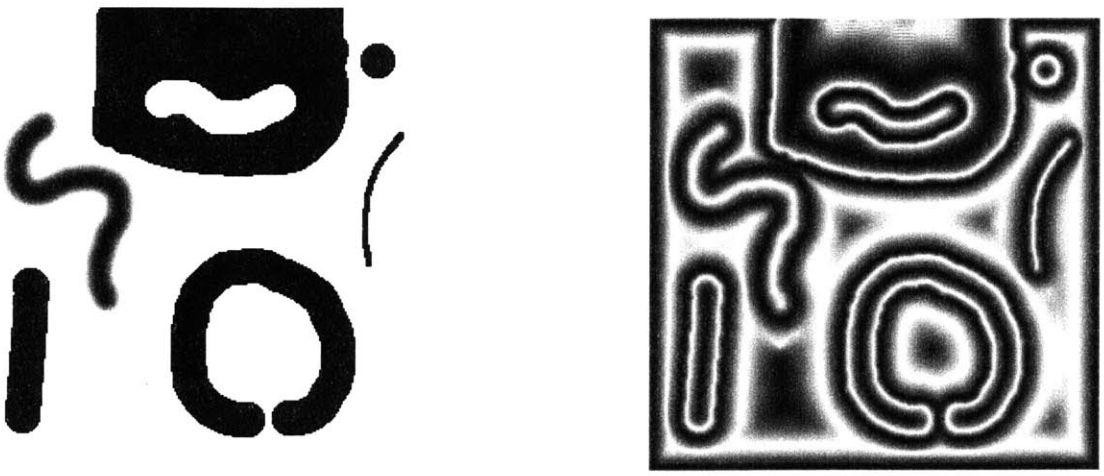


Figure 4-5: Local phase of an example image. The image on the left was filtered to extract local phase information, which is shown in the image on the right. (We give a description of the filters used in Chapter 5.) In the phase image, values from 0 to $\frac{\pi}{2}$ are dark, while values of $\frac{\pi}{2}$ to π are bright. Here, the phase as shown in Figure 4-4 has been "folded upward" such that all edges ($-\frac{\pi}{2}$ and $\frac{\pi}{2}$) have phase of $\frac{\pi}{2}$.

- Local phase can provide subpixel information about the location of edges in the image. (As Figure 4-5 shows, the phase values surrounding an edge reflect the nearness of the edge.)

- Phase is generally stable across scales [9].

Chapter 5

Phase-Based User-Steered Image Segmentation

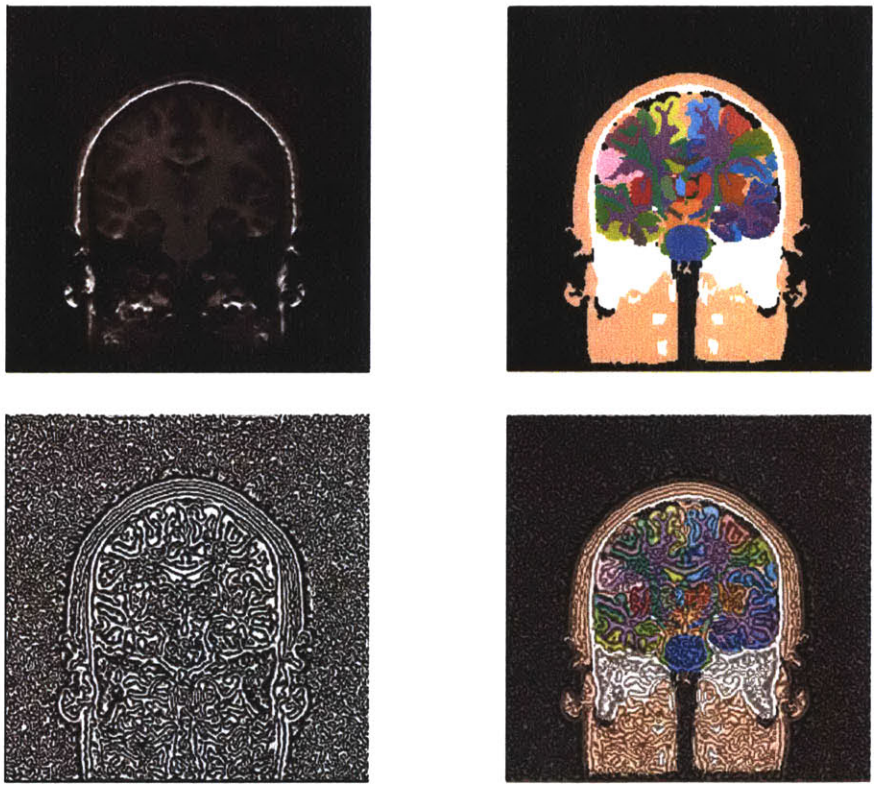


Figure 5-1: Visual comparison of phase image and expert segmentation. From the left, a grayscale image, the matching expert segmentation, the phase image, and an overlay of the phase image on the segmentation.

In this chapter we describe our system using the framework introduced in the previous chapters. The first part of the chapter explains our algorithm in the context of the livewire overview presented in Chapter 3. Here we also describe our use of local phase as an image feature, extending concepts introduced in Chapter 4 [31].

The second part of the chapter situates our algorithm in the "Knowledge Continuum" presented in Chapter 2. Finally, the third part of the chapter gives details of the system features and user interface.

5.1 Phase-Based Livewire Step One: Creation of the Weighted Graph

5.1.1 Graph Representation

The phase-based livewire situates graph nodes on pixel centers, and the shortest paths are found along pixels. This method of graph alignment was preferred by users since it provided intuitive user interaction. The other method, situation of graph nodes on pixel corners, was more difficult to use since each mouse click needed to define a pixel corner, which introduced ambiguity (since each pixel has four corners). More information about graphs used in livewire was presented in Chapter 3.

5.1.2 Local Image Features: Edge Detection Using Local Phase

Computation of local phase in two dimensions is an extension of the one-dimensional instantaneous phase defined in Chapter 4 to two dimensions [13]. We use a set of oriented filters, known as quadrature filters, in computing local phase. Each filter defines a window of interest in both the spatial and frequency domain (which implies that the kernel should be designed to capture the features of interest in the data). The amplitude of the filter output is a measure of the signal energy in the window of the filter, while the argument of the filter's output, or the phase, gives a statement

about local signal symmetry within the window.

We compute two features for input to the livewiring process. The first is the local phase, and the second we call certainty, since it expresses how certain we are that an edge exists in a particular image location. The computation of both image features is described in detail in the following sections.

Filter Kernels in the Fourier Domain

The quadrature filters are designed in the Fourier domain, where each filter is limited to half of the plane in order to filter out the negative frequencies. (Negative frequencies in two dimensions are defined relative to a reference direction [13], which in this case is the orientation direction of the filter. The important part is that eliminating any half of the frequency domain allows one to produce the analytic signal. So then any further action of the filter is actually filtering the analytic signal.)

We currently employ four oriented filters in computation of local phase. Though it is not necessary to use that many filters to compute local phase, the four filters have nice properties for computing information about local orientation. Exploiting this would be interesting for future work.

The filters are oriented to provide even directional coverage in the plane, as shown in Figure 5-2. For each filter kernel, the negative half-plane is on the far side of a line drawn perpendicular to the reference direction. (This sounds confusing, but is more obvious that the left half of the plane is dark when we show a filter kernel in Figure 5-4.)

The quadrature filters are designed with a radial frequency function that is Gaussian on a *logarithmic* scale:

$$\nu(\rho) = \mathrm{e}^{-\frac{4}{\log 2} B^{-2} \log^2(\frac{\rho}{\rho_i})} \tag{5.1}$$

where ρ_i is the center frequency and B is the width at half maximum, in octaves. This function is plotted in Figure 5-3 for $B = 2$ and $\rho_i = \frac{\pi}{3}$. The purpose of this kernel shape is to filter the analytic signal, keeping the low-frequency signal of interest while

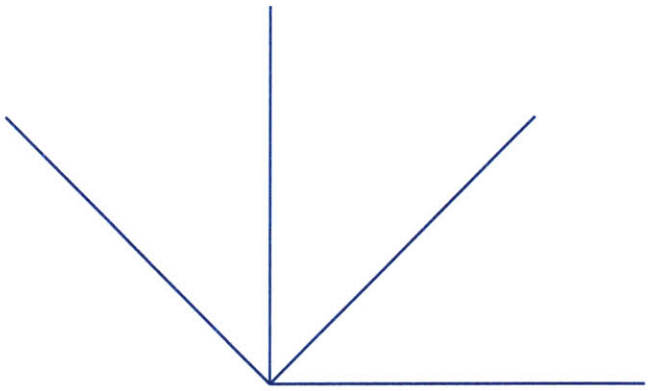


Figure 5-2: Quadrature filter orientations. These orientations are the same in both the spatial and Fourier domains. One of the oriented kernels is shown in Figure 5-4.

attenuating the higher frequency components which are potentially noise. The center frequency basically defines the size of the image events that can be observed, while the bandwidth controls the specificity to that size of event.

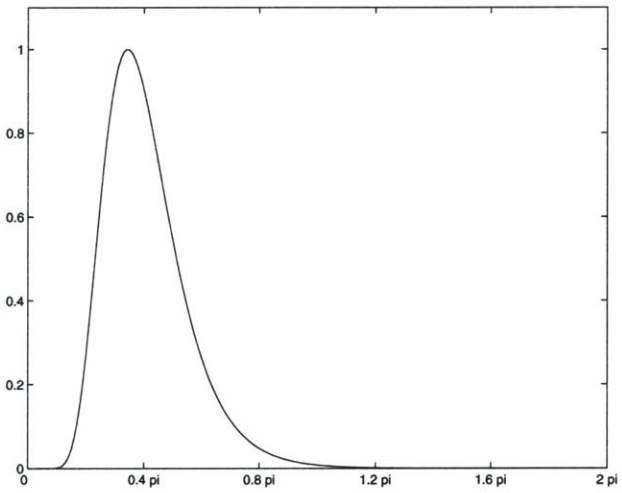


Figure 5-3: Profile of lognormal filter kernel shape, as described in Equation 5.1. Values of $B = 2$ and $\rho_i = \frac{\pi}{3}$ were used to generate this curve.

One filter kernel is shown from above in Figure 5-4. In this colormap, warm colors (the red and maroon) indicate larger values than cool colors (the blue background). An example set of four kernels in the Fourier domain can be seen in Figure 5-5.

We have found that, in addition, multiplying the lognormal function by a cos^2

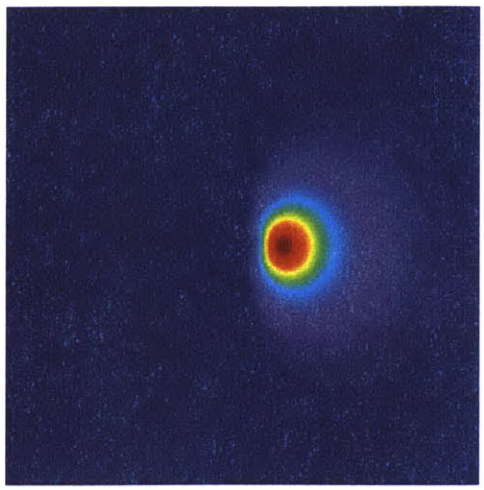


Figure 5-4: An example quadrature filter kernel in the Fourier domain. The others are the same but oriented along the other directions that were shown in Figure 5-2. This image is like a bird's-eye-view when compared to Figure 5-5.

radial function window in the frequency domain forces the "positive-frequency" tail smoothly to 0 to avoid an abrupt cutoff at the edge of the kernel. This reduces ringing artifacts arising from the discontinuity at π when using filters with high center frequencies and/or large bandwidth.

Filter Kernels in the Spatial Domain

The filter kernels are purely real in the Fourier domain, which means that in the spatial domain they must have an even real component and an odd imaginary component [13]. Consequently, each kernel produces a pair of oriented kernels in the spatial domain, where the even-shaped kernel will be sensitive to even events (for example, lines) and the odd-shaped kernel will be sensitive to antisymmetric events such as edges. See Figure 5-6 for a filter pair.

In contrast to Gabor filters, these filters have zero response for negative frequencies because the lognormal function does not have a tail that crosses into the "negative-frequency half plane." This ensures that the odd and even parts constitute a Hilbert transform pair (using the definition of the analytic signal) which makes the filters ideal for estimation of local phase [22].

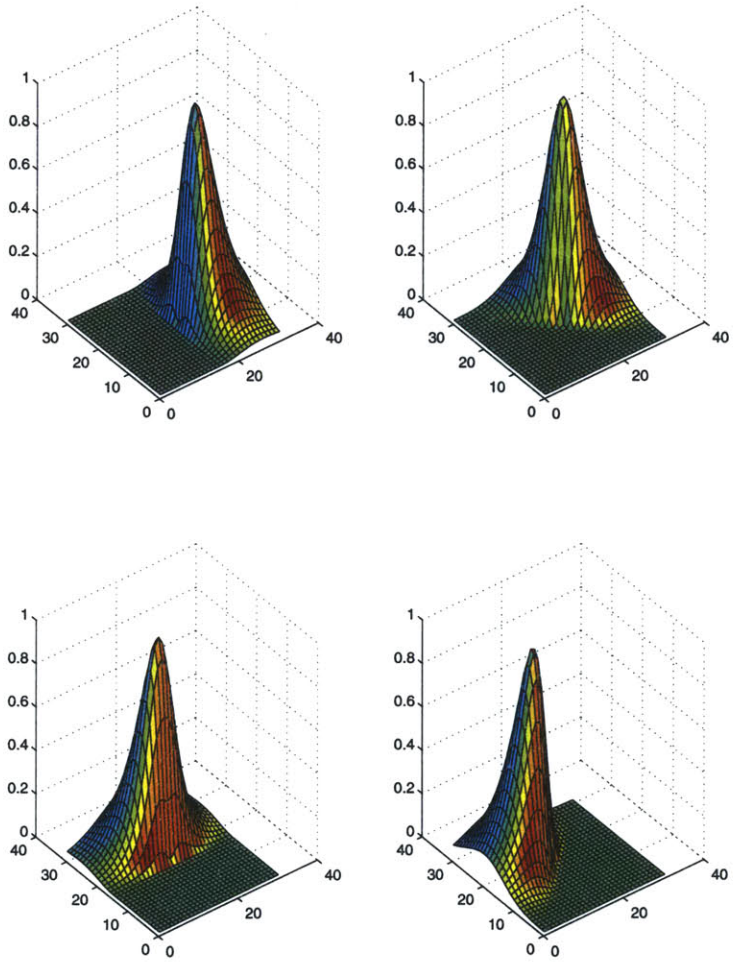


Figure 5-5: Four example quadrature filter kernels in the Fourier domain.

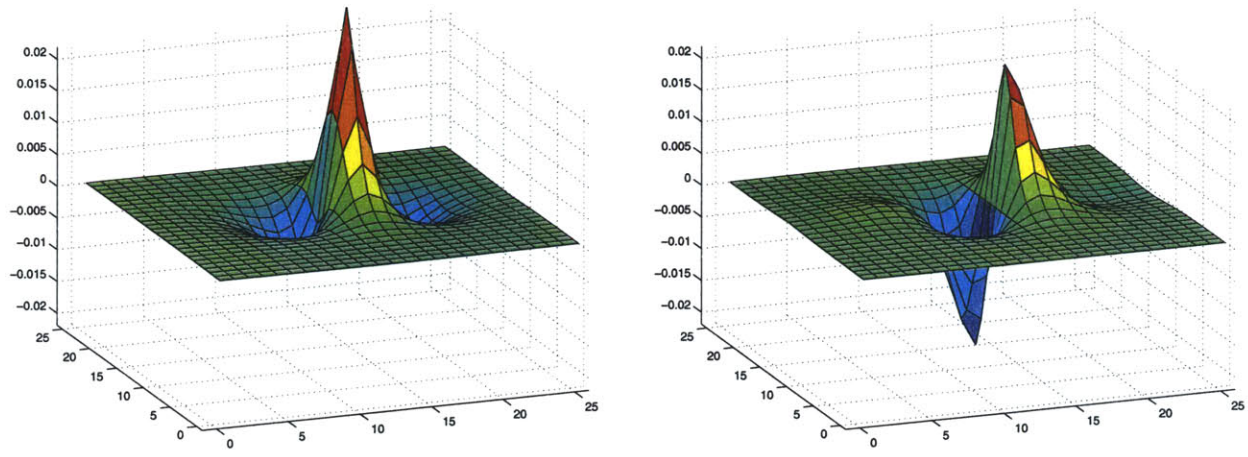


Figure 5-6: Example quadrature filter pair in the spatial domain. This filter kernel pair is one of four oriented pairs used on our data. The filter kernel on the left has an even symmetry, and the filter kernel on the right has an odd symmetry. The two kernels together form a complex filter kernel pair, with the even kernel being the real part, and the odd kernel being the complex part.

Computation of Local Phase Feature from Filter Output

Our goal is to detect boundaries in a medical image in order to aid segmentation between anatomical structures. This "boundary location" information will then become input to the livewire semiautomatic segmentation tool. The local phase is our primary feature, serving to localize edges in the image: the livewire essentially follows along low-cost curves in the phase image. The phase feature is scaled to provide bounded input to the shortest-paths search.

The local phase of an image describes how even or odd the signal is in a window around each pixel. One way to imagine this is by thinking of a complex kernel pair in the spatial domain, and realizing that the phase is the argument of the output. Consequently the phase angle measures how much the odd (imaginary) kernel responded versus the even (real) kernel in the neighborhood. See Figure 5-7.

For input to the shortest-paths search, at each voxel we must produce one number that represents its "goodness" as a boundary element with respect to the segmentation task at hand. Consequently, it is necessary to combine the information from the four

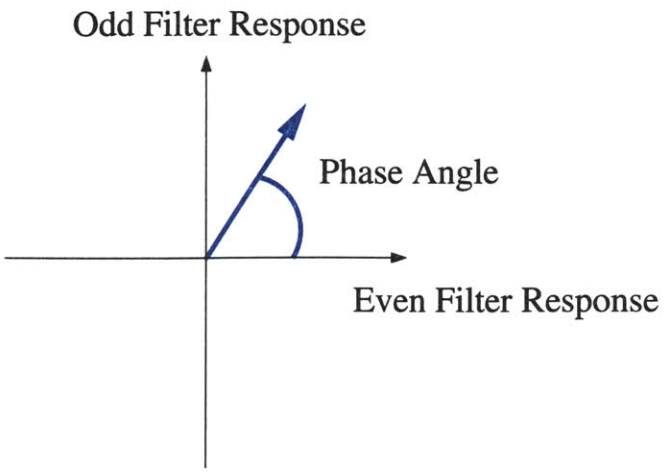


Figure 5-7: Phase angle as argument of complex filter kernel output. The phase measures the local relationship between oddness and evenness.

oriented filter kernels in order to produce one phase value per pixel. We do this by summing the outputs of the even (real) kernels and summing the absolute values of the outputs of the odd (imaginary) kernels. Then the phase is computed as the argument of the summed complex output value at each pixel. (This is the arctangent of the imaginary part divided by the real part, as shown in Figure 5-7.)

The absolute value of the result from the odd kernels is used to avoid pathological cancellation of edges, which is important as we are primarily interested in the edge information from the output. The "pathological" (but actually common) case would be when an edge is oriented such that some kernels measure a phase of approximately $\pi/2$ and other kernels measure $-\pi/2$. Since either value represents an edge, we first take the absolute value of the odd filter outputs to avoid cancellation in the sum.

Figure 5-8 shows a phase image and the original image from which it was derived.

Multiscale Phase Stability

In this set of experiments, we show the stability of phase information across different scales [9]. The input to the quadrature filters was blurred using a Gaussian kernel of variance 4 pixels. Figure 5-9 demonstrates the robustness of phase information to changes in scale, as the two segmentations are quite similar. The leftmost image is

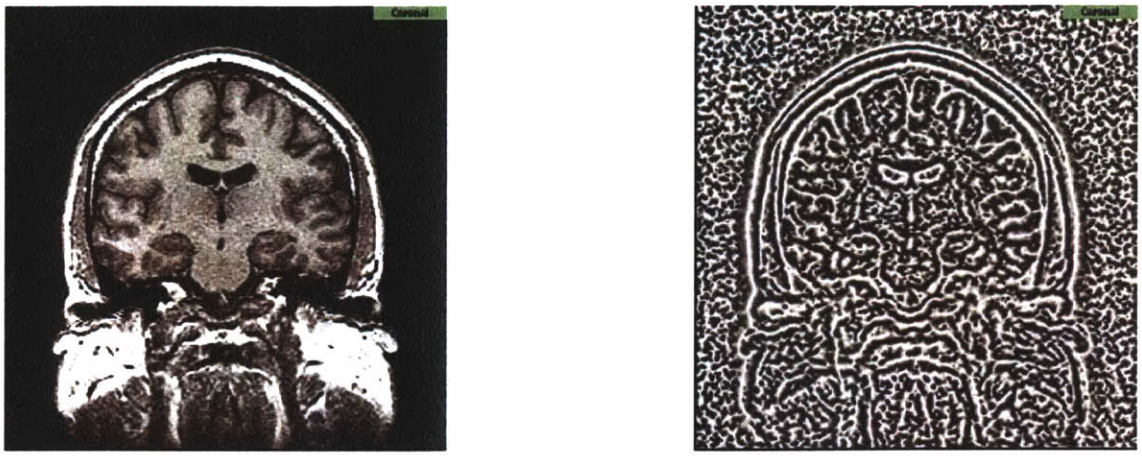


Figure 5-8: Phase image derived from an MR scan of the brain.

the original grayscale and its segmentation, while the center image shows the result of Gaussian blurring. The rightmost image displays the segmentation curve from the blurred image over the original image for comparison.

Original Image Gaussian Blurred Input Overlay of Segmentation

Figure 5-9: Illustration of insensitivity to changes in image scale: initial image, result of Gaussian blurring, and overlay of second segmentation (done on the blurred image) on initial image. Note the similarity between the segmentation contours, despite blurring the image with a Gaussian kernel of four pixel variance.

Computation of Certainty Feature from Filter Output

The phase image (Figure 5-8) is noticeably noisy outside the brain: since phase is not sensitive to signal magnitude, there is phase "everywhere" in an image. To reduce this effect, we can also use signal energy information derived from the magnitude of

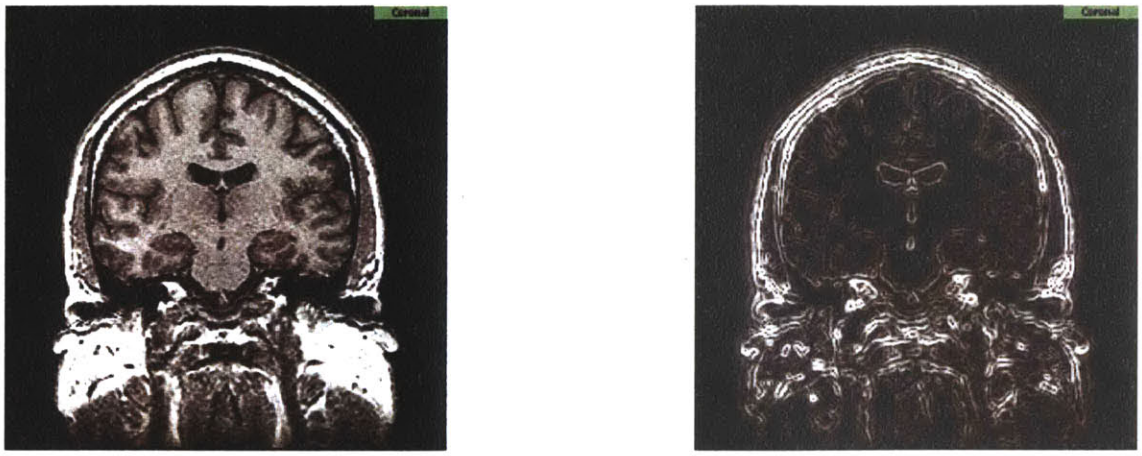


Figure 5-10: Certainty image derived from an MR scan of the brain.

the odd (imaginary) filter outputs. (Note that the odd-shaped filters are the ones that should react to edges in the image since their shape matches the shape of an edge.) In image regions where the odd filter output magnitude is large, the energy of the signal is high, which implies that the phase estimate in that region has high reliability, or certainty.

We define the certainty using the magnitude from the odd quadrature filters, mapped through a gating function. The gating function clamps the upper values the certainty can take on, to avoid overly biasing the segmenter toward strong edges. By inverting this certainty measure, since high certainty implies low cost, it can be used as a second feature in the livewire cost function. One "certainty" image, which shows how confident we are that there is an edge at a particular location and at a particular scale (center frequency), is shown in Figure 5-10.

5.1.3 Local Image Features: Directionality

Currently, we do not employ a specific feature that influences the local direction that the livewire path should take. We plan in the future to investigate use of local directionality information that can be obtained from the quadrature filters.

5.1.4 Local Image Features: Training

Several features of our system allow the user to input more information about the structure of interest, in order to instruct the wire to perform as desired.

Feature Size Selection

The first system feature is the ability to select the image feature size of interest. This is important since edges exist at many scales in the image, and the center frequency of the filters controls the scale at which edges will be detected. To this end, a menu for small, medium, and large feature sizes, as described in Section 5.4.2, was added to the Phasewire module.

Window Leveling Before Phase Computation

The second feature allows the user to intuitively select the grayscale range of interest in the data by window-leveling the images.

Originally we computed phase directly from the grayscale data. However, depending on the local shape of the image signal, the phase edge did not always align with the perceived boundary in the image. In order to concentrate only on the grayscale range of interest, we began computing phase from window-leveled images. Window-leveling removes uninformative regions of the grayscale histogram, and thus both the phase and certainty features are computed from better information than if the original grayscales were used for input.

In Figure 5-11 the effect of window-leveling on the combined phase/certainty image is shown. The gray-white matter boundaries of interest are more marked in the image that was computed from window-leveled data.

Phase Selection

The third feature allows the user to follow along phase values that are higher or lower than $\frac{\pi}{2}$. The effect of this is to segment more toward light regions, or more toward dark regions. It is similar to an erosion or dilation of the boundary.

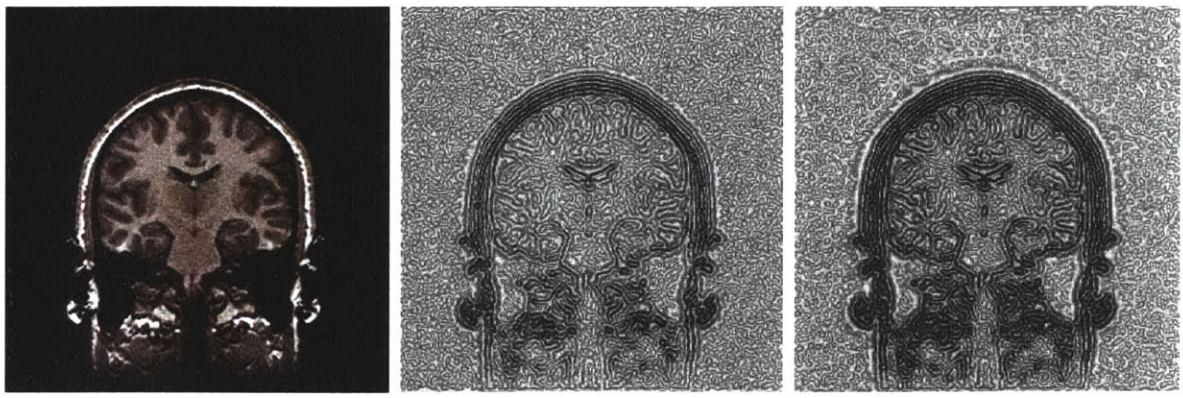


Figure 5-11: Window-leveling the data before phase computation better defines boundaries of interest. The leftmost image is the original grayscale (displayed after the window-leveling operation). The image on the far right was computed from window-leveled data, while the center image was computed from raw data. (Note that in these images dark regions are "attractive" to the livewire.)

5.1.5 Combination of Features to Produce Edge Weights

We investigated Gaussian models, simple scaling, inversion of feature values, and a lookup table when combining feature values in our two implementations of livewire.

The final Phasewire implementation uses a weighted combination of the phase feature and the clamped, inverted certainty feature. Before summing, the features are scaled and converted to integer values to provide bounded integer input to the shortest path search. In the Advanced part of the user interface, it is possible to change the relative weighting of the features in the sum.

In Figure 5-12 we display an example of the combined phase and certainty input to the shortest-paths search. The dark regions in the image represent phase of $\pi/2$ or $-\pi/2$, and are attractive to the livewire. Note that in the upper left of the image, in the anterior region of the brain, there is some intensity inhomogeneity. As the local phase feature is insensitive to this, in the right-hand image, there is decent elucidation of local image structure in this area.

A demonstration of the utility of the two features in segmentation is given in Figure 5-13.

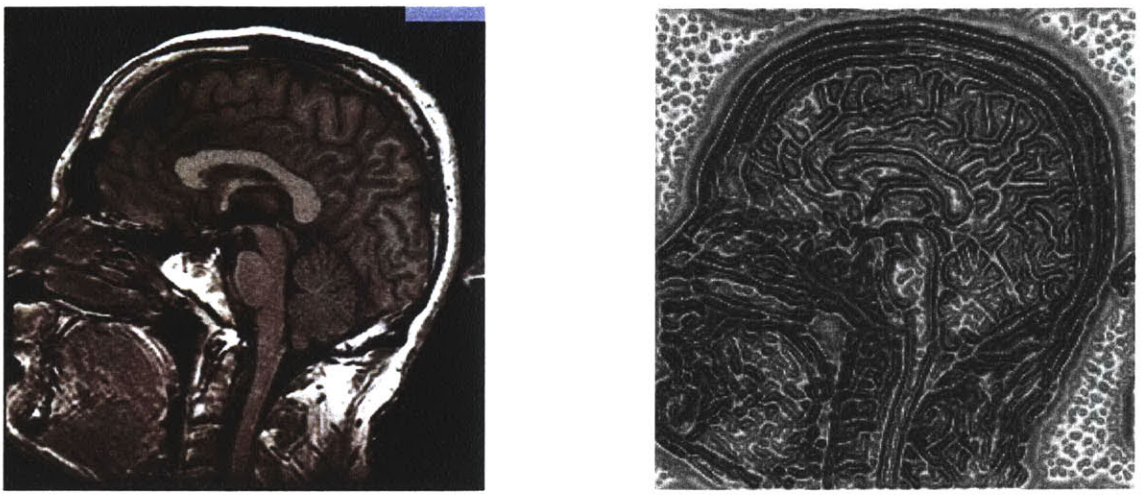


Figure 5-12: Weighted graph computed from a reformatted sagittal neural MR image. The image on the left is a reformatted grayscale slice through the original dataset, while the image on the right was computed in the Slicer as a combination of phase and certainty information. In this case, the darkest regions in the phase image correspond to $+/-\pi/2$ of phase, which is an edge in the original image.

5.2 Phase-Based Livewire Step Two: Shortest Paths

We use the algorithm "Live Wire on the Fly" as described in Section 3.3.2 in Chapter 3 [7]. This method is faster than the standard application of Dijkstra's algorithm which entails computing shortest paths from the initial point to all points in the image. Instead this method computes shortest paths for as much of the image as necessary to reach the endpoint, and stores this information for use when the mouse moves to select another endpoint.

5.3 Situation of Our Method in the "Knowledge Continuum"

Here we place our method in the "Knowledge Continuum" introduced in Chapter 2.

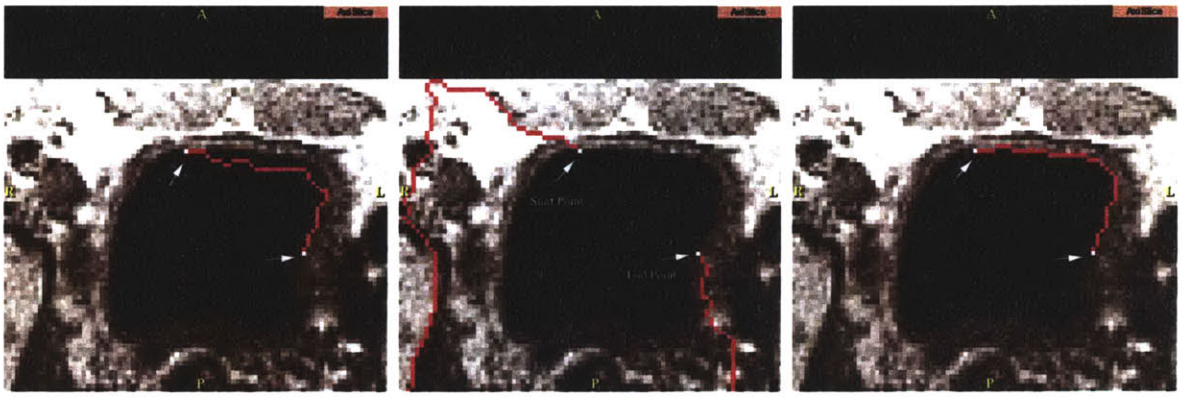


Figure 5-13: Utility of both image features: segmentation with individual features and the combination. The leftmost image is an attempt to segment using the phase feature only. The contour is distracted by phase values in the interior of the bladder (remember the magnitude of the edge does not influence phase). The center image is an attempt to segment using the certainty feature only, and the contour is distracted by the highest-value boundaries in the image. The rightmost image displays a contour drawn using a combination of these features, which functions better than either feature alone. The center frequency of the filters used was the "Large" setting in the Slicer, $\frac{\pi}{2}$.

5.3.1 Knowledge Employed By Livewire and Phasewire

The phase-based livewire method uses sophisticated filtering to extract local phase information from the image. This knowledge places it into the category of "Local Intensity Knowledge Used by Algorithm" as defined in Chapter 2. Livewire does not employ anatomical or shape knowledge, but its generality and interactivity allow it to be applied to various segmentation problems where an automated, knowledgeable solution does not exist.

5.3.2 Use of Additional Knowledge

Like many other segmentation algorithms, the information used as input to livewire can be thought of as separate from the algorithm itself. In other words, the features used to guide the livewire can come from any source of information, so it is possible to inject additional knowledge about a specific segmentation problem if the knowledge is available. For example, information that could be incorporated into the weighted graph includes: distances from other structures in the image, the distance from the same structure in the previous slice, or a prior probability that a certain graph region should contain the structure.

As the livewire method is interactive, it provides an interesting testbed for features that might be useful for segmentation. The pros and cons of the feature when using livewire will be similar to those encountered if it is used to drive an automatic method, such as level set evolution. But the livewire is simpler and our user population at the Surgical Planning Lab has access to it, so there exists the potential for a great amount of feedback from clinicians regarding its performance when guided by a new feature.

5.4 System Features

In this section, we introduce specific details about the interface and demonstrate the features that are available to users of our system.

5.4.1 User Interface for Controlling Filter Parameters

We have integrated the phase-based livewire segmentation method into the 3D Slicer, the Surgical Planning Lab's platform for image visualization, image segmentation, and surgical guidance [10]. Our system has been added into the Slicer as a new module, PhaseWire, in the Image Editor.

Figures 5-14 and 5-15 show the interface of the PhaseWire module and its integration into the 3D Slicer visualization environment. Most of the parameters presented to the user are similar to those in the Draw function of the Image Editor, with additional parameters for filter control.

5.4.2 Filter Parameters: Center Frequency

The quadrature filters are not selective to grayscale value, nor orientation, but their shape must pick up the appropriate edges in the image. So it is important in practice to choose a center frequency that corresponds to the size of the structures of interest in the images.

As shown in the user interface in Figure 5-14, the user may select the relative size of the structure of interest. Experimentation with various center frequencies, as defined in Equation 5.1, led to the following settings which we employ in the PhaseWire module:

- Small Image Feature Size: $\rho_i = \pi$
- Medium Image Feature Size: $\rho_i = \frac{\pi}{\sqrt{2}}$
- Large Image Feature Size: $\rho_i = \frac{\pi}{2}$

Figure 5-16 shows phase images computed using various center frequencies. Figures 5-17 and 5-18 demonstrate the effect on segmentation of varying the center frequency.

5.4.3 Filter Parameters: Bandwidth

We found that setting B (the width at half maximum) to 2 for all kernels was effective. B controls the specificity of the filter to the center frequency. Choosing B is a tradeoff because one desires to restrict the filter's response to the frequency of interest, in order to ignore unwanted events at other frequencies, but one also desires to include events that occur at frequencies close to the center frequency. Setting B to 2 provided reasonable behavior in our system, though further investigation of this parameter, perhaps for fine-tuning segmentation of specific structures, would be interesting.

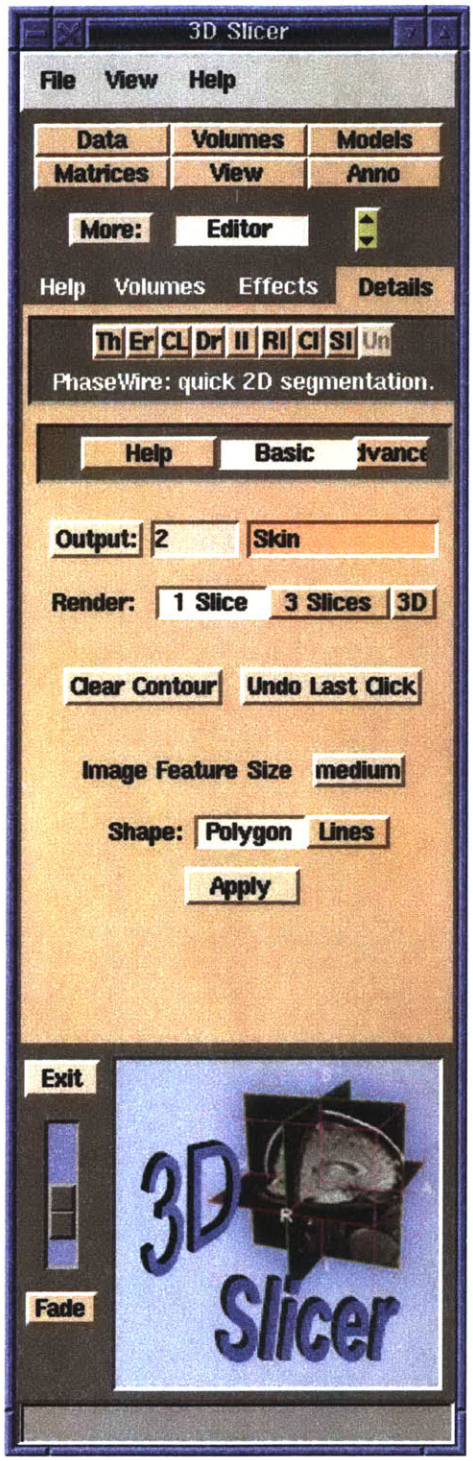


Figure 5-14: User Interface for PhaseWire. Options presented to the user are similar to other Slicer editing modules, with the exception of the "Clear Contour," "Undo Last Click," and "Image Feature Size" buttons, which are specific to PhaseWire.

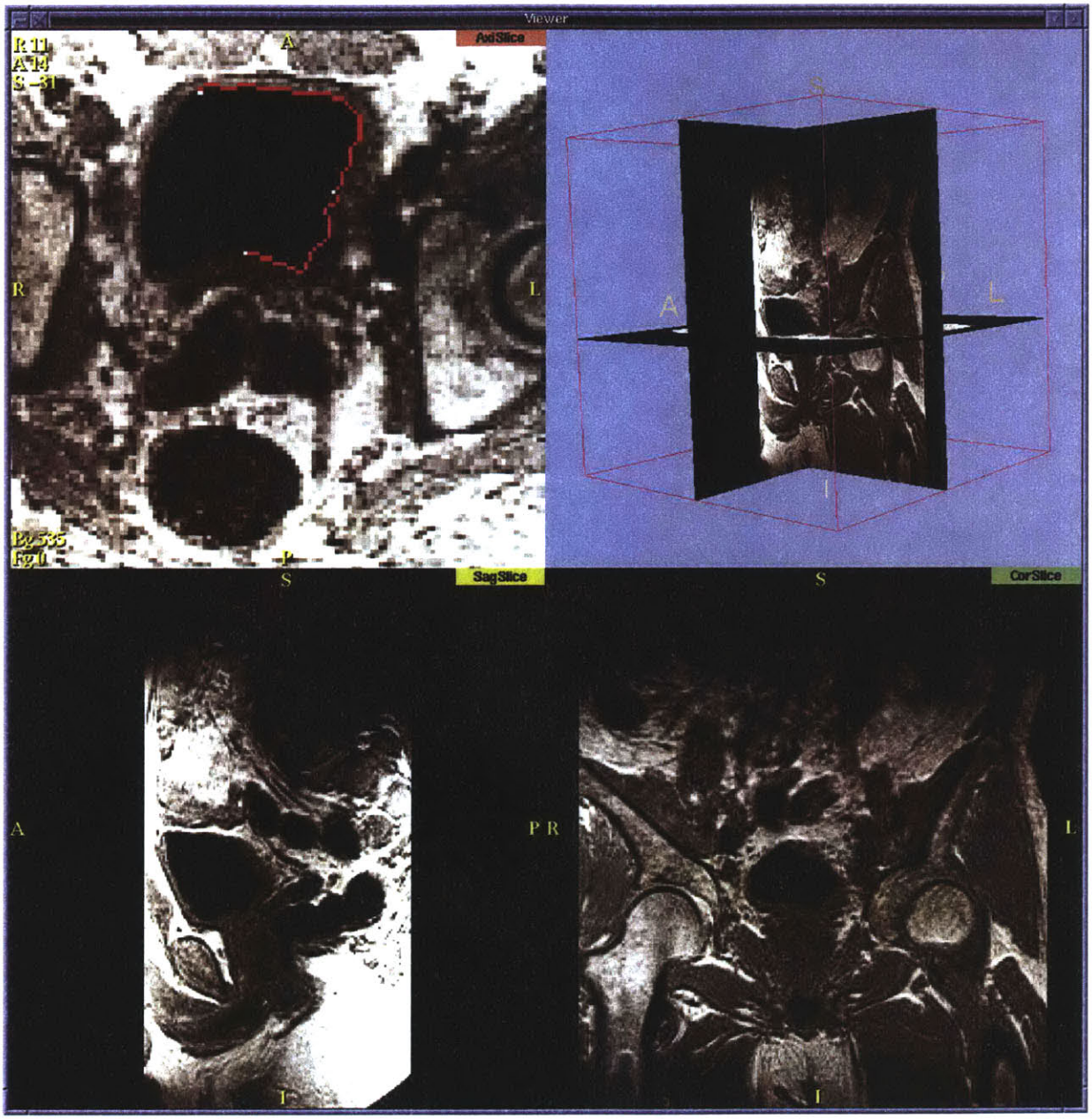


Figure 5-15: 3D Slicer image display interface, during a PhaseWire editing session.

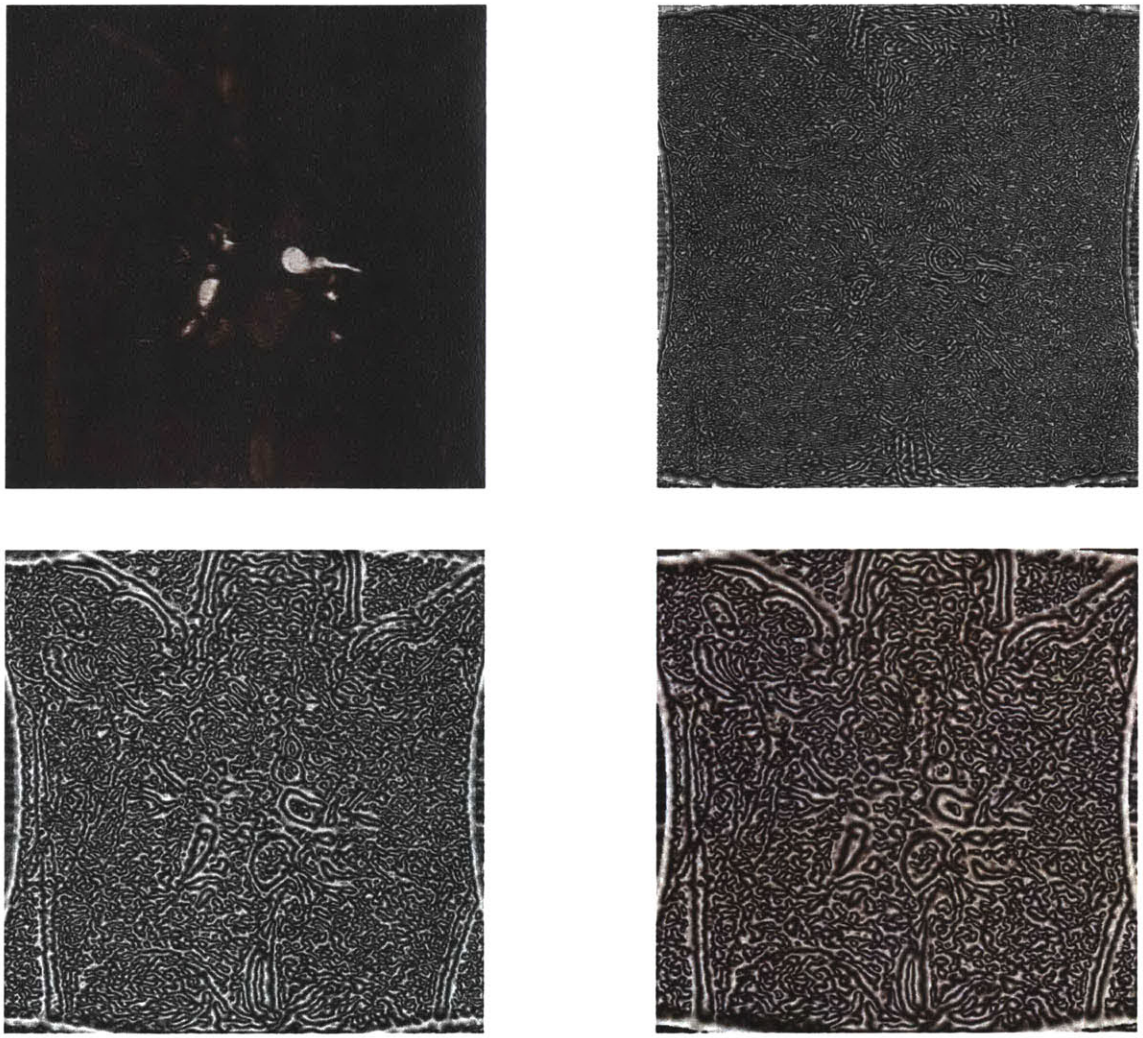


Figure 5-16: Effect of varying the center frequency. The phase was computed from the grayscale image using center frequencies of $\frac{\pi}{3}$ (upper right image), $\frac{\pi}{6}$ (lower left image), and $\frac{\pi}{8}$ (lower right image). Close examination shows that the higher frequency, $\frac{\pi}{3}$, captures best the detail, though it is harder to interpret visually.

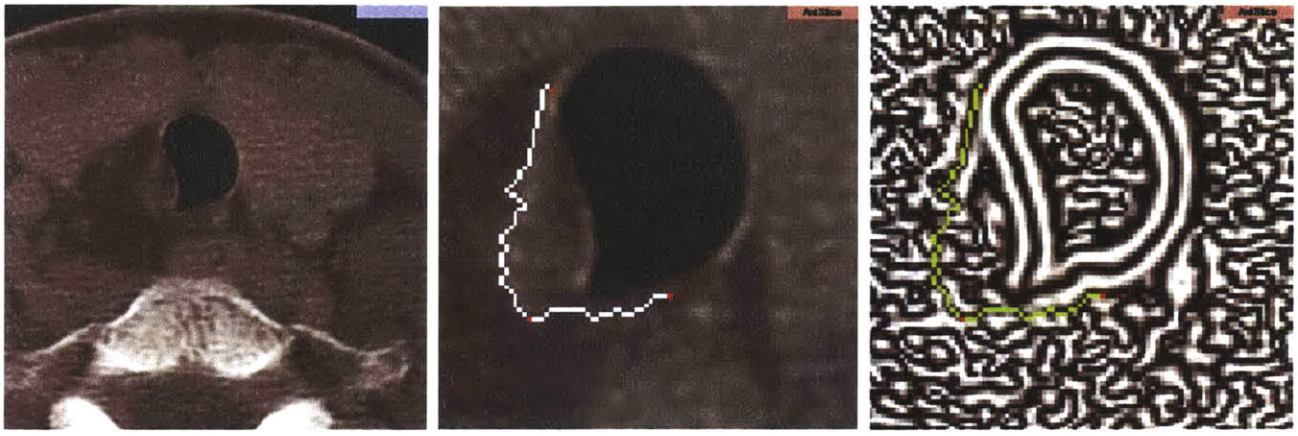


Figure 5-17: Center frequency example. The leftmost image is a cervical CT scan. The middle image is a zoom on the area of interest, a tumor bordering the trachea. The white livewire contour contains three red dots, which represent the points where the user clicked in the image. The rightmost image is the phase image computed from the grayscale, with the segmentation contour in yellow overlay. The filter kernels used had a center frequency of $\pi/4$. This frequency is high enough to capture both the posterior border of the trachea (the black shape) and the border immediately below it, which is possibly tumor. (Since these features are relatively close, they are only separable if the center frequency is chosen high enough.)

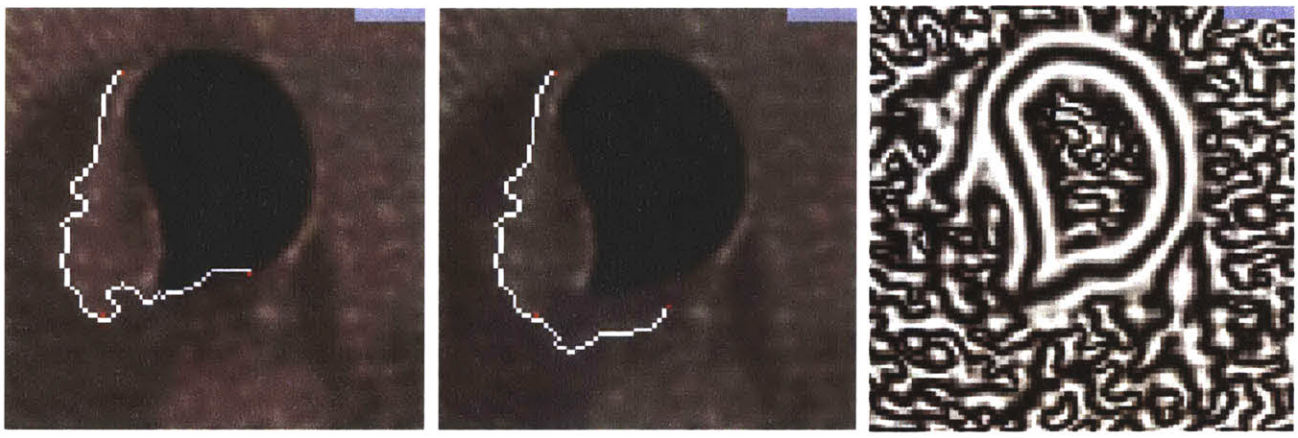


Figure 5-18: Center frequency example, effect of a lower center frequency. The filter kernels used had a center frequency of $\pi/5$. These images are from the same cervical CT as shown in Figure 5-17. But the lower center frequency is tuned to capture larger features in the image. The leftmost two images show the livewire snapping to the upper and lower regions that are attractive (dark) in the accompanying (rightmost) phase image. Note that the livewire still finds another boundary inferior to the trachea, but it now finds a more distant one due to the lower center frequency (which corresponds to a wider filter in spatial domain).

Chapter 6

Analysis of Expert Segmentations Done With Phasewire and Livewire

6.1 Introduction

In this chapter we present segmentation results from both the Phasewire and the original Livewire systems.

The Phasewire system has been in use at the Surgical Planning Lab at Brigham and Women's Hospital for four months. At the Surgical Planning Lab, we have conducted validation trials of the system and obtained expert opinions regarding its functionality. In addition, a Phasewire validation study was conducted at Hospital Dr. Negrin in Las Palmas de Gran Canaria, Spain.

6.2 Statistical Measures

Before presenting segmentation results, we give an overview of methods used to quantify the segmentation quality.

6.2.1 Volume Statistics

First, the volume of a segmentation is calculated as the number of voxels multiplied by the volume of each voxel.

Now referring to Figure 6-1, we define the "Percent Overlap" of one segmentation on the other as the fraction of segmentation A that is contained within segmentation B. This is the volume of the intersection divided by the volume of the segmentation (A or B). Finally, what we will refer to as the "Percent of Total Volume in Common" is the volume common to both segmentations (the intersection) relative to the total volume (the union).

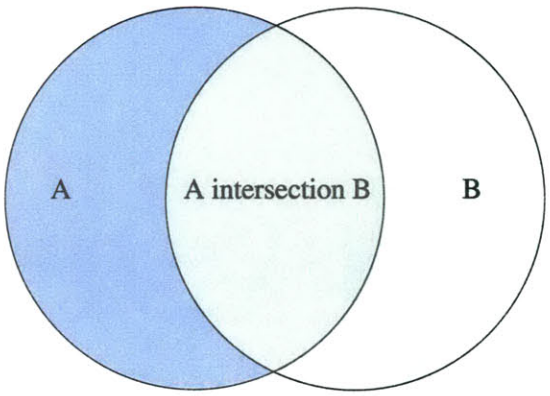


Figure 6-1: A Venn diagram. The union of A and B is the entire shaded area (both circles).

6.2.2 Hausdorff Distance

The Hausdorff distance measures the degree of mismatch between two sets. The standard version identifies the point in one set that is the farthest from the other set, and outputs the distance to the nearest point in the other set.

A generalization of this, which is less sensitive to outliers, sorts the distances between each point in one set and the corresponding closest point in the other. Then statistics are available to quantify the nearness of the points in one set to the other: for example, one can state that 80% of the voxels of one dataset are within a certain distance from the other dataset.

In addition, since the measure is not symmetric, it makes sense to compute it for one dataset relative to the other, and then vice versa, and take the maximum resulting distance. This can give an upper bound on the distance of voxels in each segmentation from the other. In this section we use the fourth quintile, or 80%, generalized Hausdorff distance metric, which gives an upper bound on the distance of 80 percent of the voxels in each segmentation from the other [24].

6.3 Logging

In order to enable validation of the segmentation method, we have created a system that continuously records user interaction and can generate a database containing the number of user interactions, such as mouse events, and time stamps from various editing modules. A selection of the events that can be recorded is listed in Table 6.1.

Item Logged	Details
mouse clicks	recorded per slice, per label value, per volume, and per editor module
elapsed time	recorded per slice, per label value, per volume, and per editor module
description	includes volumes viewed during the session
username	may allow investigation of learning curve

Table 6.1: Selected items logged by the 3D Slicer for validation and comparison purposes.

This logged data is included in the segmentation information that follows when it was reasonably consistent, or collected during a controlled study. This is because several factors reduced the reliability of the logged data. The main issue was that the program usage time was inaccurate if the user left the 3D Slicer window open during the day. This is very common practice at the Surgical Planning Lab, where an open Slicer window will prevent the loss of one's computer during breaks or lunch. Another factor is that users segmented many scans in one session, complicating the log files. Finally, some users had the habit of exiting the program abruptly with a Ctrl-C key sequence, which prevented logging.

6.4 Expert Segmentations

6.4.1 Controlled Segmentation Experiments

In this section we present the results of the segmentation trial performed by doctors from Hospital Doctor Negrin in Las Palmas de Gran Canaria, Spain. Abdominal and neural CT scans were segmented in a controlled experiment to ensure validity of the logged information. Table 6.2 demonstrates the speed of the segmentation method using information from the logging system, while Table 6.3 gives doctors' opinions regarding the ease of use of the system and the quality of the output segmentations. The doctors' opinions about the system were positive, and they considered that the segmentations produced by the system were superior to the manual segmentations, since the phasewire aided them in producing smooth boundaries.

Study	Method	Total Clicks	Clicks/Slice	Time/Slice (sec)	Volume (mL)
CT brain tumor	manual	234	26.0	39.3	68.5
	phasewire	97	10.8	28.7	67.3
	phasewire	109	12.1	25.5	69.2
CT bladder	manual	1488	28.1	31.7	715.6
	phasewire	359	6.8	21.5	710.8

Table 6.2: Example segmentations performed with Phasewire. Clicks and time per slice are averages over the dataset.

For example, Figure 6-2 shows a liver segmentation done with phasewire, in which the doctor remarked that the three-dimensional model was more anatomically accurate than the one produced with the manual method, where both segmentations were performed in approximately the same amount of time. This is likely due to the smoothness of the contour obtained with phasewire, in comparison with the contour obtained with the manual method.

Study	Method	Ease of Use	Segmentation Quality
CT brain tumor	manual	3	3
	phasewire	4	4
CT bladder	manual	2.7	3
	phasewire	4	4

Table 6.3: Doctors' comparison of Phasewire and Manual segmentation methods on the datasets from Table 6.2. The scale is from 1 to 5, with 5 being the highest.

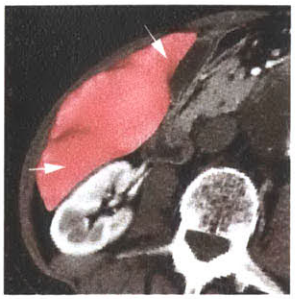
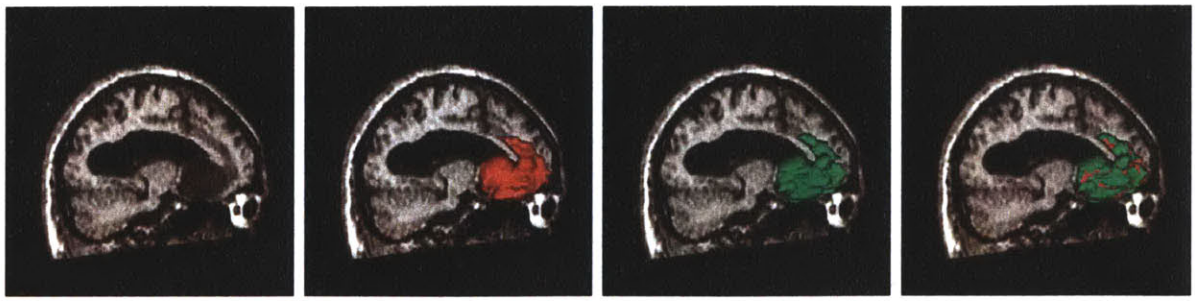


Figure 6-2: A surface model of the liver, created from a phasewire segmentation, shows well-defined surface indentations between the liver and the kidney and gallbladder. These surfaces are marked with arrows.

6.4.2 Comparison of Manual and Phasewire Tumor Segmentations

In this case, a neural MR scan containing a tumor was rapidly re-segmented by a neurosurgeon in his first time using the Phasewire system. We present a comparison between this segmentation and the very careful manual segmentation that was done previously by the same neurosurgeon. These results are shown to demonstrate that a novice user of the system can rapidly produce a segmentation that is similar to a manual segmentation. The neurosurgeon in this case stated that some of the differences in the segmentations were due to the fact that he had not studied the case recently, and might have chosen slightly different borders for the tumor each time.

Figure 6-3 displays models generated from both segmentations. Then Table 6.4 makes a quantitative comparison between the two.



Original Image Manual Phasewire Overlay of Models

Figure 6-3: Manual and Phasewire tumor segmentation. The manual segmentation was carefully done while the Phasewire segmentation was rapidly traced.

Labelmap	Volume in mL	Percent Overlap
Manual	48.866	90.87%
Phasewire	47.969	92.57%

Percent of Union in Common	Hausdorff Dist.
84.69%	2.5mm

Table 6.4: A tumor segmentation. Between 7% and 9% of each segmentation is not contained within the other, but 80% of the pixels of each segmentation are within the generalized Hausdorff distance of 2.5 mm from the other segmentation. As 2.5 mm was the slice thickness of the scan, and the in-plane resolution was 0.859375 mm, this means that most boundary voxels were within 1-3 voxels of the boundary of the other segmentation.

6.4.3 Pulmonary Vein Segmentations

Pulmonary vein segmentations were performed in three MR angiography cases using the original Phasewire implementation, and then repeated using an updated version. The newer version contained several improvements to the method. First, the local phase was computed after the data had been window-leveled. This enabled the user to indicate the grayscale range fo the data of interest in an intuitive manner. Second, the filters were improved by multiplication with a $cosine^2$ radial function which forced the kernels to smoothly go to 0 on their boundaries.

A visual comparison of three-dimensional models derived from the data shows that the segmentations done using the newer version were more complete, in that the

veins could be followed deeper into the lungs. Figure 6-4 shows two models, one from the original Phasewire implementation, and one where window-leveled data was used for local phase computation. Both segmentations were done solely with Phasewire for the purposes of this study. However, since semi-automatic segmentation of such data was time-consuming, it was thought that the phasewire would be better used as a fine-tuner to complete the segmentation after initial thresholding.

Original Phasewire

Improved System

Figure 6-4: Pulmonary veins segmented with Phasewire. The model on the left was segmented using the original system. The green model overlayed in the right-hand image was segmented using local phase and certainty information computed after window-leveling the data. It captures more of the pulmonary circulation.

6.4.4 Repeatability of Three Methods on the Temporal Pole

The temporal pole, located in the anterior tip of the temporal lobe, has an important role in retrieval of semantic and episodic memory especially in a emotional context. This region is segmented by schizophrenia researchers at the Surgical Planning Lab.

The temporal pole region in a neural MR scan was segmented six times, twice manually, twice with the original livewire implementation, and twice with Phasewire. The repeatability of the Livewire and Phasewire systems was comparable, while the two Hausdorff distance implied that the manual segmentations were actually quite different (despite their similar volume measures). Table 6.5 gives Hausdorff distances for the 4th quartile, as measured for the repeated segmentation pairs.

	Manual	Livewire	Phasewire
80% Hausdorff Distance	4.47 mm	0.94 mm	0.94 mm

Table 6.5: Repeatability of three methods on the temporal pole. Generalized Hausdorff distances were measured between original and repeated segmentations done with each method.

Grayscale Image | Manual Segmentation | Phasewire Segmentation

Figure 6-5: Phasewire vs. Manual on Partial-Volume Border Pixels.

The Phasewire system in this case chose to include more pixels on the border, as demonstrated in the images in Figure 6-5. This increased the overall volume of the structure. However, this repeatability study was done before the introduction of the prefilter window-leveling or the phase selection slider, both of which are aimed at allowing greater user control in this type of situation. It is also true that there in the absence of a gold standard it is difficult to state absolutely which pixels should be chosen on the boundary.

Volumetric comparisons of the various segmentations follow in Table 6.6.

6.4.5 Fusiform Gyrus

The fusiform gyrus, located in the ventromedial surface of the temporal lobe, is important for face perception and recognition. In two cases, this structure was segmented using manual and Phasewire segmentation. Volumetric and time results are shown in Table 6.7. In the first case, the overall volume was larger with Phasewire, and so the

Labelmap	Volume in mL	Percent Overlap	Percent of Total Vol. in Common
Manual	8.724	0.9124	0.8362
	8.755	0.9092	0.8362

Labelmap	Volume in mL	Percent Overlap	Percent of Total Vol. in Common
Livewire	8.840	0.8957	0.8107
	8.846	0.8951	0.8107

Labelmap	Volume in mL	Percent Overlap	Percent of Total Vol. in Common
Phasewire	9.345	0.9308	0.8758
	9.285	0.9368	0.8758

Table 6.6: Volumetric measures from repeated segmentations.

second case was measured with more frequent clicks. However, a large time savings was found in segmentation of the first case. (This study was also done with the first version of Phasewire, and consequently investigating further with the present system and future improvements is of interest.)

	Manual	Phasewire
case1 volume	5.264 mL	6.699 mL
case2 volume	5.984 mL	5.996 mL
case1 time	87 min	48 min

Table 6.7: Segmentation of the fusiform gyrus.

6.4.6 Thalamus

Phasewire was found to be useful in the lateral border of the thalamus (marked with an arrow in Figure 6-6), which is a weak boundary along which segmentation is difficult. Figure 6-6, which shows a segmentation of the thalamus, highlights a strength of the phase feature used by the system: its insensitivity to grayscale magnitude. Phasewire's combination of local signal shape information and user input produces a reasonable segmentation of this difficult structure.

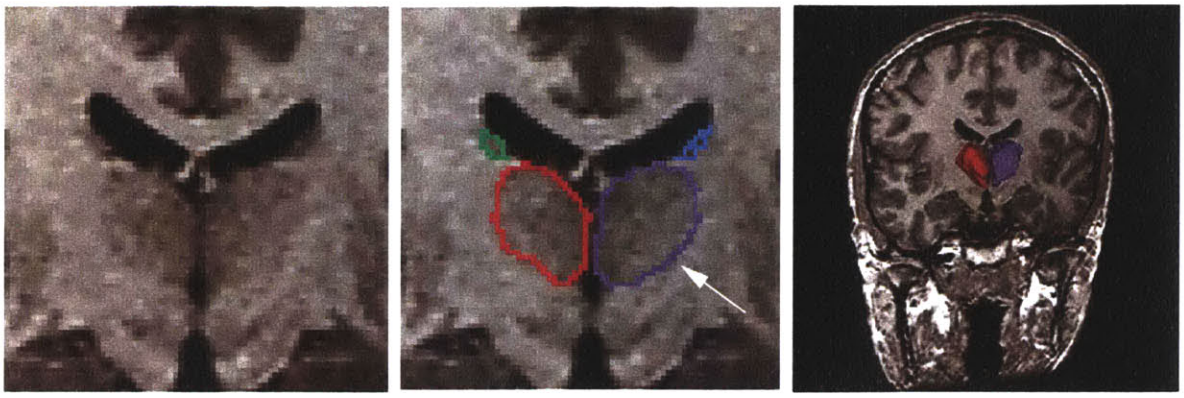


Figure 6-6: Phasewire segmentation of the thalamus, right and left sides. The weak lateral border (marked with an arrow in the middle image) is difficult to segment manually, but Phasewire can be used effectively in this region.

6.5 Discussion of Segmentation Results

Generally, where speed and reproducibility are important factors, and there exists no automatic method, Phasewire would be preferred over manual segmentation. However, in extremely critical segmentations, such as those of the schizophrenia group at the Surgical Planning Lab, years of research have been based on manual segmentation and volume measurement of certain structures. Consequently, the adoption of a new system is difficult unless it can perform for all intents and purposes exactly like a manual segmenter. This is not an easy problem, and more tweaking of the system would be necessary to approach this functionality.

Chapter 7

Discussion

In this work, two systems were implemented based on the livewire paradigm. The first used the image features described in [8] while the second employed a novel feature, local phase. We investigated the behavior of both tools in segmentation of medical imagery. In this chapter, we will describe problems encountered and system improvements, address the pros and cons of both approaches, discuss future work, and finally summarize the outcome of the project.

7.1 Issues and Improvements to System Functionality

The main issue with the Phasewire system was matching the contour produced by the system to the contour that was desired by the user. This, of course, is not a simple problem as the operator segments based on grayscale information plus a wealth of anatomical knowledge, while the algorithm only sees the grayscale imagery. We added three improvements to the system to address this issue.

The first system feature is the ability to select the image feature size of interest. This is important since edges exist at many scales in the image, and the center frequency of the filters controls the scale at which edges will be detected. To this end, a menu for small, medium, and large feature sizes ($\rho_i = \pi$, $\rho_i = \frac{\pi}{\sqrt{2}}$, and $\rho_i = \frac{\pi}{2}$ are

the respective center frequencies) was added to the Phasewire module.

The second feature allows the user to intuitively select the grayscale range of interest in the data. We began computing local phase on already window-leveled data in order to focus in on only the values of interest, and better localize edges.

The third feature, the phase slider in the Advanced tab of the Phasewire module, allows the user to follow along phase values that are higher or lower than $\frac{\pi}{2}$. The effect of this is to segment more toward light regions, or more toward dark regions. It is similar to an erosion or dilation of the boundary. In structures where the boundary is consistent around the structure, this feature is useful. If the boundary changes, the desired phase value may need to be changed as well. Experiments were performed with picking a new phase value at every mouse click, but this was found to be a bit noisy. This is an area of interest for future investigation.

Finally, another issue was encountered with the filter kernels themselves. The kernels did not smoothly go to 0 at the boundaries, which caused an abrupt cutoff of frequencies of interest instead of the desired smooth lognormal tail. To ensure that all kernels had the desired property, even those sensitive to high frequencies, we multiplied by a cos^2 radial function in the frequency domain. The resulting filtered images were qualitatively smoother.

7.2 Comparison of the Two Systems

The Phasewire system was generally preferred by users for several reasons. The first reason was the more intuitive user interaction due to the fact that the paths were computed along pixels.

The second reason was that the local phase generally gives smooth contours for the livewire to follow, which produces smoother movement of the wire. This made the phase-based system seem more intelligent. Generally, the phasewire is not distracted locally by extraneous attractive pixels, but instead by another path. If so, movement of the mouse can generally set it back on the border of interest.

The third reason was that the training required for the original system was confus-

ing, and not always successful since varied boundary characteristics were not captured well. However, it was true that when training was successful, the original system was very specific to the desired grayscale levels, unlike the Phasewire system.

7.3 Future Work

Several potential improvements to the system are of interest for future investigation.

First, though it has been found with Phasewire that a specific local direction feature is not necessary, utilizing information about the local orientation of structures in the image could prove beneficial (both for smoothness and for rejecting unsuitable paths). A vector describing the most prevalent local orientation can be obtained from the oriented quadrature filters, so it is logical to add this as another image feature in the system.

Currently, the system treats both dark edges on a light background and light on dark as the same thing. This can cause ambiguity and "jumping" from one to the other, since both are equally preferred. This problem has been addressed in the training process of the original Livewire system [7]. Using a more complete representation of local phase, which includes the edge orientation (as determined using the orientation of the quadrature filters), the two cases of edge can be disambiguated. This representation of phase is three-dimensional: it adds another dimension to the phase diagram shown in Figure 4-4 in order to include the angle of dominant local orientation [13, 17].

Another problem of interest is training, or learning from the user what is desired. This has been addressed in other work by computing averages of features and using a Gaussian model, and also by building a distribution of gradient magnitudes. This information is then used to assign lower costs to preferred image regions. In our first system, the Gaussian model type training was implemented, and the main difficulty was that structures with varying boundary characteristics could not be segmented well due to averaging of the characteristics during training. To address this, some experimentation was done with a smoothed histogram, or Parzen density estimator,

but the resulting distribution was noisy and did not aid segmentation. It would be interesting to revisit this problem in the context of phase-based livewire, perhaps applying additional machine learning techniques.

7.4 Conclusion

We have presented a user-steered segmentation algorithm which is based on the livewire paradigm. Our results using local phase as the main driving force are promising. The application of the tool to a variety of medical data has been successful. The method is intuitive to use, and requires no training despite having fewer input image features than other livewire implementations.

Bibliography

[1] W. A. Barrett and E. N. Mortensen. Interactive live-wire boundary extraction. *Medical Image Analysis*, 1(4):331–341, 1997.

[2] Boualem Boashash. Estimating and interpreting the instantaneous frequency of a signal - part 1: Fundamentals. *Proceedings of the IEEE*, 80(4), 1992.

[3] Boualem Boashash. Estimating and interpreting the instantaneous frequency of a signal - part 2: Algorithms and applications. *Proceedings of the IEEE*, 80(4), 1992

[4] V. Caselles, R. Kimmel, G. Sapiro, C. Sbert. Minimal Surfaces: A Three Dimensional Segmentation Approach. Technion EE Pub 973, 1995

[5] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic Active Contours. *Proceedings of Fifth International Conference on Computer Vision* 694–699, 1995.

[6] E. W. Dijkstra. A Note On Two Problems in Connexion With Graphs *Numerische Mathematik*, 269–271, 1959

[7] A. X. Falcao, J. K. Udupa, and F. K. Miyazawa. An Ultra-Fast User-Steered Image Segmentation Paradigm: Live Wire on the Fly. *IEEE Transactions on Medical Imaging*, 19(1):55–61, 2000.

[8] A. X. Falcao, J. K. Udupa, S. Samarasekera, and S. Sharma. User-Steered Image Segmentation Paradigms: Live Wire and Live Lane. *Graphical Models and Image Processing*, 60:233–260, 1998.

[9] D. J. Fleet and A. D. Jepson. Stability of Phase Information. *IEEE Trans. PAMI*, 15(12):1253–1268, 1993

[10] D. Gering, A. Nabavi, R. Kikinis, N. Hata, L. O'Donnell, W. Eric L. Grimson, F. Jolesz, P. Black, W. Wells III. An Integrated Visualization System for Surgical Planning and Guidance Using Image Fusion and an Open MR. *Journal of Magnetic Resonance Imaging*, 13:967–975, 2001

[11] D. T. Gering, A. Nabavi, R. Kikinis, W. E. L. Grimson, N. Hata, P. Everett, and F. Jolesz and W. Wells. An Integrated Visualization System for Surgical Planning and Guidance Using Image Fusion and Interventional Imaging. *Medical Image Computing and Computer-Assisted Intervention - MICCAI'99*, 809–819, 1999.

[12] P. Golland, W. E. L. Grimson, Martha E. Shenton, and R. Kikinis. Small Sample Size Learning for Shape Analysis of Anatomical Structures *Medical Image Computing and Computer-Assisted Intervention - MICCAI'2000* 72–82, 2000

[13] G. A. Granlund and H. Knutsson. *Signal Processing for Computer Vision*, Kluwer Academic Publishers, 1995.

[14] C. Guttmann et al. Computerized Image Processing for Quantitative Follow-up of Patients. *Journal of Magnetic Resonance Imaging* 9(4):509-518, 1999.

[15] C. R. G. Guttmann, H. L. Weiner, L. Hsu, S. J. Khoury, E. J. Orav, M. J. Hohol, S. S. Ahn, R. Kikinis, F. A. Jolesz. The natural course of relapsing-remitting and chronic progressive multiple sclerosis. *International Society for Magnetic Resonance in Medicine*, 2:1327, 1998.

[16] C. R. G. Guttmann et al. Multiple Sclerosis Project Website, BWH Surgical Planning Lab. *http://splweb.bwh.harvard.edu:8000/pages/projects/ms/ms.html*

[17] L. Haglund, H. Knutsson, G. H. Granlund On Phase Representation of Image Information. *The 6th Scandinavian Conference on Image Analysis*, 1082–1089, 1989

[18] T. Kapur. Model based three dimensional Medical Image Segmentation. *Ph.D. Thesis, Artificial Intelligence Laboratory, Massachusetts Institute of Technology*, 1999.

[19] M. Kaus, M. Warfield, F. Jolesz, and R. Kikinis Adaptive template moderated brain tumor segmentation in MRI. *Bildverarbeitung fur die Medizin* 102–106, 1998.

[20] M. Kaus et al. Automated Segmentation of MR Images of Brain Tumors *Radiology*, 218(2):586-591, 2001.

[21] Kitware. *The Visualization Toolkit*, http://www.kitware.com/vtk.html

[22] H. Knutsson and C.-F. Westin and G. H. Granlund. Local Multiscale Frequency and Bandwidth Estimation *Proceedings of IEEE International Conference on Image Processing*, 36–40, 1994

[23] J. Liang, T. McInerney, D. Terzopoulos. Interactive Medical Image Segmentation with United Snakes. *Medical Image Computing and Computer-Assisted Intervention - MICCAI'99*, 116–127, 1999.

[24] M. Leventon. Statistical Models for Medical Image Analysis. *Ph.D. Thesis, Artificial Intelligence Laboratory, Massachusetts Institute of Technology*, 2000.

[25] L. M. Lorigo, O. Faugeras, W. E. L. Grimson, R. Keriven, R. Kikinis, C.-F. Westin. Co-dimension 2 Geodesic Active Contours for MRA Segmentation. *International Conference on Information Processing in Medical Imaging.* June/July 1999, Visegrad, Hungary.

[26] W. E. Lorensen and H. E. Cline. Marching Cube: A High Resolution 3-D Surface Construction Algorithm. *Computer Graphics* 21(3):163–169, 1987.

[27] R. W. McCarley, C. Wible, M. Frumin, Y. Hirayasu, J. J. Levitt, I. A. Fischer, M. E. Shenton. MRI Anatomy of Schizophrenia. *Biological Psychiatry 1999*, 45:1099-1119, 1999.

[28] T. McInerney, D. Terzopoulos. Deformable Models in Medical Image Analysis: A Survey. Medical Image Analysis, 1(2):91–108, 1996.

[29] E. N. Mortensen and W. A. Barrett. Interactive Segmentation with Intelligent Scissors. *Graphical Models and Image Processing*, 60(5):349–384, 1998.

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

[31] L. O'Donnell, Carl-Fredrik Westin, W. Eric L. Grimson, Juan Ruiz-Alzola, Martha E. Shenton, Ron Kikinis Phase-Based User-Steered Image Segmentation *Medical Image Computing and Computer-Assisted Intervention - MICCAI'01*, 1022–1030, 2001.

[32] E. Simoncelli and W. Freeman The Steerable Pyramid: A exible architecture for multi-scale derivative computation *Int'l Conference on Image Processing*, 3:444–447, October 1995.

[33] S. Warfield et al. Intraoperative Segmentation and Nonrigid Registration for Image Guided Therapy. *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2000*, 176–185, 2000.

[34] S. Warfield et al. Adaptive, Template Moderated, Spatially Varying Statistical Classification. *Medical Image Analysis*, 4(1):43–55, 2000.

[35] S. Warfield et al. Automatic Segmentation of MRI of the Knee. *ISMRM Sixth Scientific Meeting and Exhibition*, p.563, April 18-24, 1998.

[36] W. M. Wells, R. Kikinis, W.E.L. Grimson, F. Jolesz. Adaptive segmentation of MRI data. *IEEE Transactions on Medical Imaging*, 15:429–442, 1996.

[37] C-J Westelius Focus of Attention and Gaze Contol for Robot Vision. *Dissertation No 379, Linköping University, Sweden* ISBN 91–7871–530–X, 1995

[38] C.-F. Westin, J. Richolt, V. Moharir, R. Kikinis Affine Adaptive Filtering of CT Data *Medical Image Analysis* 4(2):161–172, 2000

[39] C. F. Westin et al Tensor Controlled Local Structure Enhancement of CT Images for Bone Segmentation. *Medical Image Computing and Computer-Assisted Intervention - MICCAI'98*, 1205–1212, 1998