1-21-2014

Brain tumor segmentation with minimal user assistance

Liqun Liu
The University of Western Ontario

Supervisor
Olga Veksler
The University of Western Ontario

Graduate Program in Computer Science
A thesis submitted in partial fulfillment of the requirements for the degree in Master of Science

BRAIN TUMOR SEGMENTATION WITH MINIMAL USER ASSISTANCE

(Thesis format: Monograph)

by

Liqun Liu

Graduate Program in Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Masters of Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

Abstract

In this thesis, we propose a brain tumor segmentation system that requires only 4 clicks from users to specify a tight bounding box that completely contains the tumor.

We convert the segmentation problem to an energy minimization problem. We utilize the basic energy function that combines intensity appearance and boundary smoothness. Global and local appearance models are experimented and compared in our work.

The basic energy function does not assume any shape prior and thus leads to unrealistic shapes. We take the advantage of the fact that most of the tumors are approximately convex in shape and incorporate the star shape prior to prohibit unlikely segmentations.

Another problem with the basic energy function is the undersegmentation problem. With the bounding box provided by the user, we are able to have a rough idea of the tumor size. Therefore, to encourage the segmentation to be a certain size, we add volumetric bias to our energy, which helps solve this problem.

We also try to model the tumor as multi-region object where regions have distinct appearance. Specifically, we incorporate interior+exterior model for the tumor into our energy function.

Our final result is promising in terms of f-measure. Our best performance for 88 volumes is 87% using volumetric ballooning.

Keywords: Interactive segmentation, graph cuts, star shape, multi-region, volumetric ballooning

Acknowledgement

I would like to express my sincere gratitude to my advisor Dr. Veksler, who walked me through each stage of the thesis. I am very grateful for her patience in reviewing the experimental results, always spotting the underlying problem in one glance. I have learned very much from her professional attitude in research and I enjoyed every meeting with her. I really appreciate her help in reviewing my thesis and giving detailed comments even during the holidays.

I am especially thankful to Dr. Lena Gorelick as an informal co-advisor. Despite my language barriers, she was always the first to understand me. I appreciate of her help in explaining my opinion to others at every meeting, as well as for the idea of incorporating the multi-region framework into our project: it is the most interesting idea I have tried.

In addition, I would like to thank Dr. Olga Veksler and Dr. Lena Gorelick for forgiving me my frustration when there was misunderstanding between us.

I would also like to thank Dr. Yuri Boykov, who is very passionate in computer vision. I am inspired by his expertise and the various and interesting ways he explained every detail in the class.

I would also like to thank other members in the vision group: Meng Tang, Xuefeng Chang, Igor Milevskiy and Yuchen Zhong. I really enjoyed our projects discussing with them, although they were different. Without these discussions, I would not be able to understand my project so thoroughly.

In the end, I want to thank my parents, my brother and my boyfriend Song Zeng for supporting me until now and trusting me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

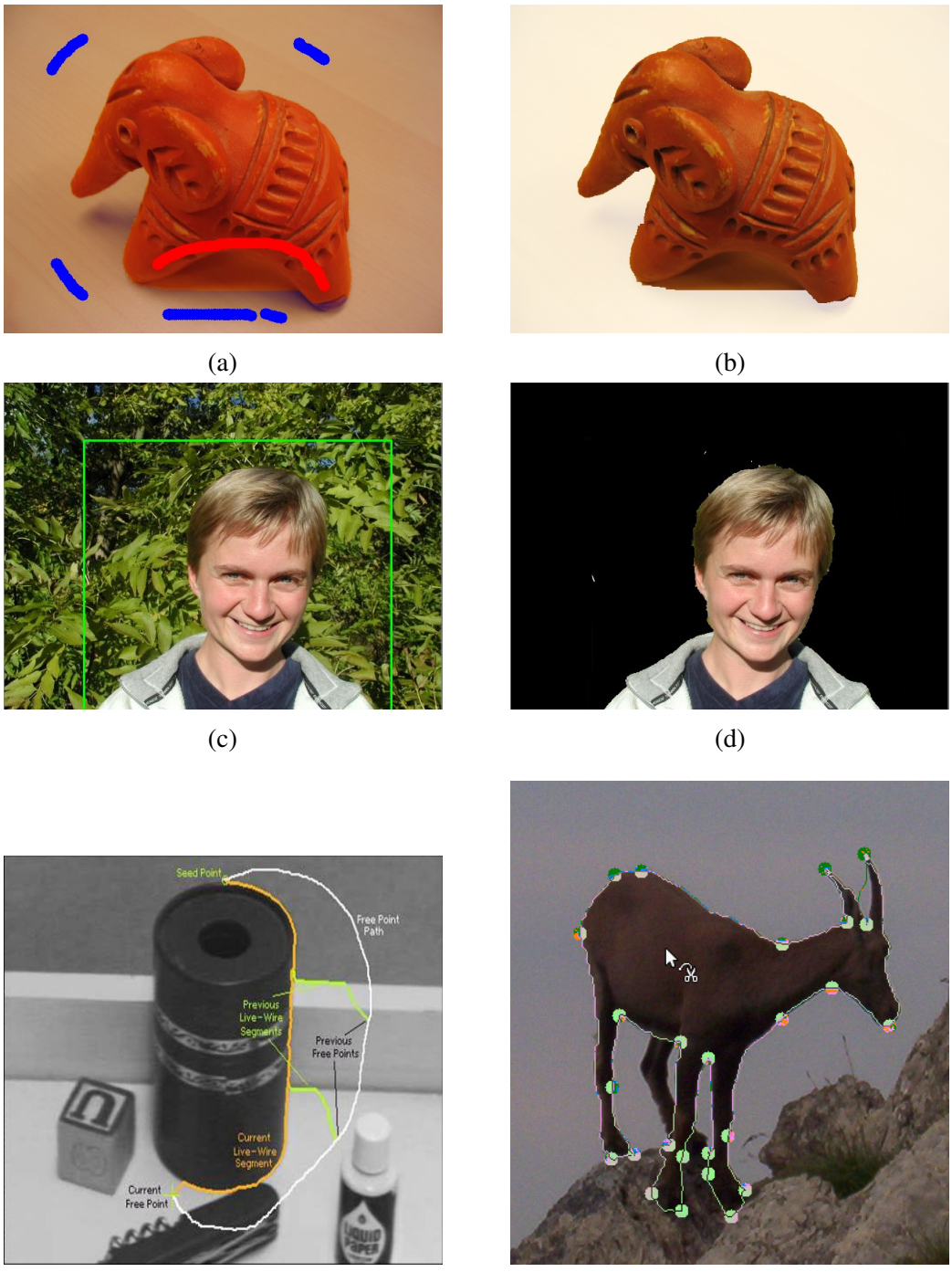
Segmentation is a process of partitioning an image into different meaningful segments, which usually correspond to different objects or object parts in the image. Image segmentation can be stated as labeling problem where each pixel is to be assigned some label (see Figure 1.1). Sometimes labels are integers representing the region number. Sometimes labels have semantic meaning like ”grass”, ”water”, etc. Pixels with similar visual characteristics such as intensity, color, texture, etc. are often assigned to the same region or the same label, if regions are represented with labels.

Segmentation methods usually fall into two classes: interactive segmentation and automatic segmentation. Automatic segmentation methods segment an image without any user assistance. Automatic object segmentation generally uses machine learning techniques to learn the appearance model for the object, based on which kind of segmentation is performed. Machine learning approach requires a large number of hand-labeled and segmented images as training data. In contrast, interactive segmentation methods usually do not rely on learning appearance models from a labeled dataset, but instead, they require additional knowledge about the object of interest, to be provided by the user. In interactive segmentation, the user can specify a rough location about the object [20], or some ”seeds” indicating some pixels as object or background [6]. There are also interactive segmentation methods where users have to point out the boundary of the object like live-wire [17]. Figure 1.2 gives examples on different interactive segmentation tools.

Applications for image segmentation include content-based image retrieval [4], object detection [22], etc. Another important application is in medical imaging. In this thesis, we develop an algorithm for brain tumor segmentation, which is important in brain tumor diagnosis and treatment decision. For example, one often needs to accurately localize the tumor and moreover, measure the tumor volume for the purpose of further treatment of cancer patient. For this purpose, accurate tumor segmentation is essential. Currently, segmenting tumors from normal

(a) (b)

Figure 1.1: (a): an image of beach. (b): the segmentation of the beach scene. Each color represents a label. (Figures from [1])



(a) (b) (c) (d) (e) (f)

Figure 1.2: Examples for different interactive segmentation tools. Top row: interactive graph cuts segmentation tool [6]. (a): User input, blue strokes are seeds for background and red strokes are seeds for object. (b) The segmentation result. Middle row: Grabcut [20]. (c):The user input with a rectangle containging the object. (d): The segmentation result. Bottom row: live-wire segmentation [17]. (e): User interaction by clicking on object edges (seed points in the image), yellow contour is the object boundary detected by the algorithm (figure from [17]). (f): Application in software Gimp (figure from [2]). Dots are user clicks.

tissues is highly time-consuming and tedious, and it takes up a significant amount of time for radiation oncologists, radiologists and other medical experts. Thus a segmentation tool, even if it requires user interaction, should be highly useful for medical practitioners.
Although there are many image segmentation techniques in computer vision, not all of them work well for medical image segmentation, where low contrast, noise and image ambiguity are often the serious challenges. In this work, our goal is to design an algorithm to segment tumors with minimum user assistance.

1.1 MRI and brain tumor

Our image data is acquired with MRI technology. Magnetic resonance imaging (MRI) is a powerful tool for visualizing internal structure inside a body in a safe way. It has the ability to record signals that can distinguish between different 'soft' tissues (such as gray matter and white matter) [8]. Moreover, the fact that MRI does not use ionizing radiation makes it even more popular among patients. In general, MRI of a brain is 3D scans of human brain, or a sampling of the brain structure in 3 different dimensions x, y, z. The sampling rates can be different along different dimensions which means that voxels of the brain scan are not equally spaced. This is a property that differs from natural images where the distance between pixels is assumed to be the same in all directions. In Figure 1.3, we show two slices of a brain scan from different patients. Each slice contains a tumor that we can easily locate by its appearance. MRI is used extensively in brain diseases examination, diagnosis and treatment. Brain tumor treatment radiation therapy depends on an accurate MRI segmentation, which requires correct labeling of the pixels in MRI images as tumor or healthy tissue. Among all medical image segmentation tasks, brain tumor segmentation is particularly challenging. In next section, we focus on the difficulties of segmenting a brain tumor.

1.2 Challenges for brain tumor segmentation

Brain tumor segmentation is particularly challenging due to the fact that tumors vary greatly in size and position, have a variety of shape and appearance properties, have intensities overlapping with normal brain tissue, and often an expanding tumor can deflect and deform nearby structures in the brain giving an abnormal geometry also for healthy tissue [9]. In this section, we give details on these challenges with examples from our data set.

High diversity in appearance

As tumors are created by an abnormal and uncontrolled cell division, they have no typical

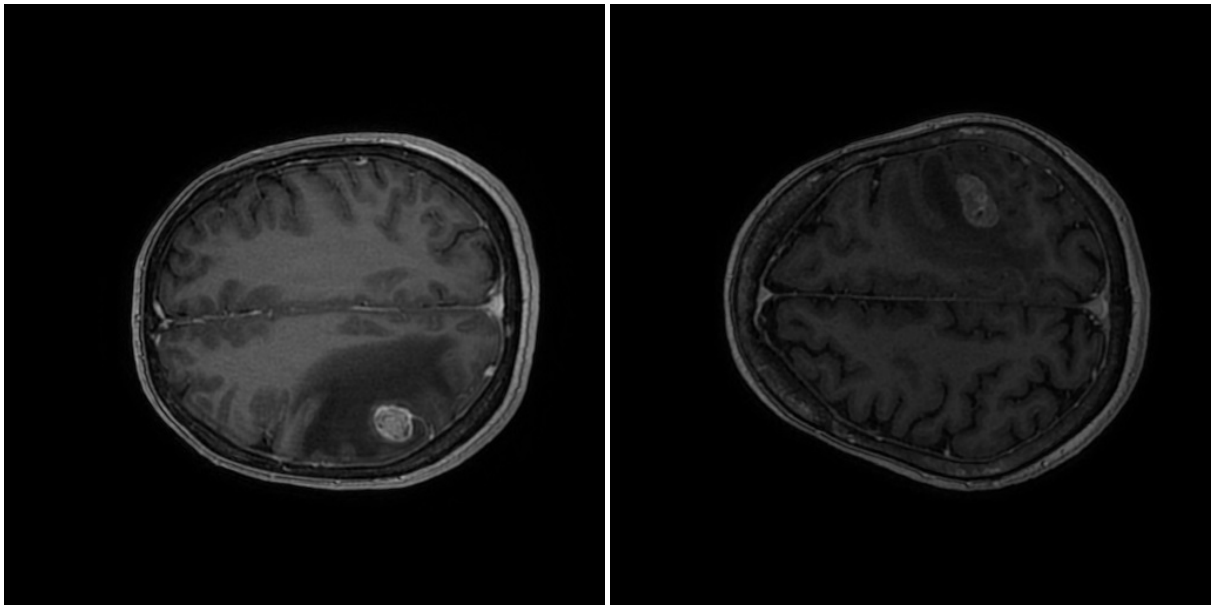
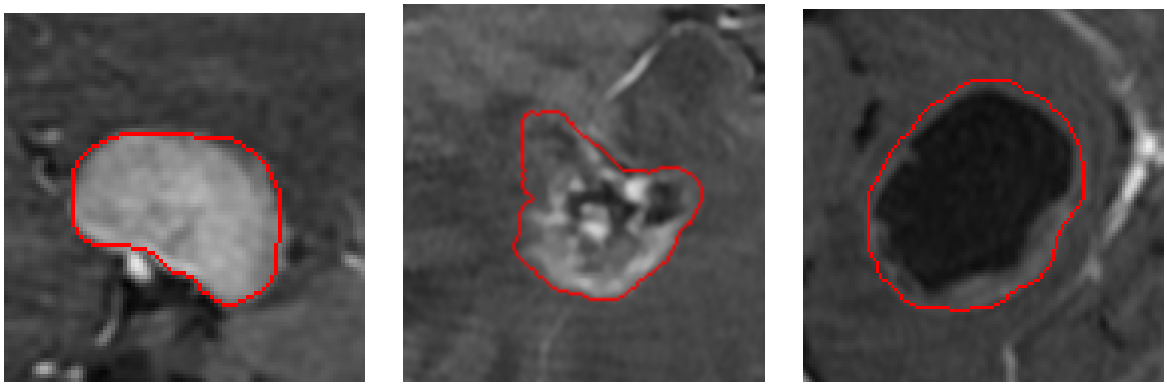


Figure 1.3: Examples for slices of a brain MRI data with a tumor.

reliable appearance. On MRI, a tumor can be darker or brighter than normal tissues, it can even appear as the same intensity as normal tissue. It holds a high diversity in appearance among different patients and even among different tumors in one patient. Figure 1.4 shows tumor from 3 different patients that share no obvious similarity in appearance. Learning appearance model from such data would probably fail due to the high diversity in appearance.



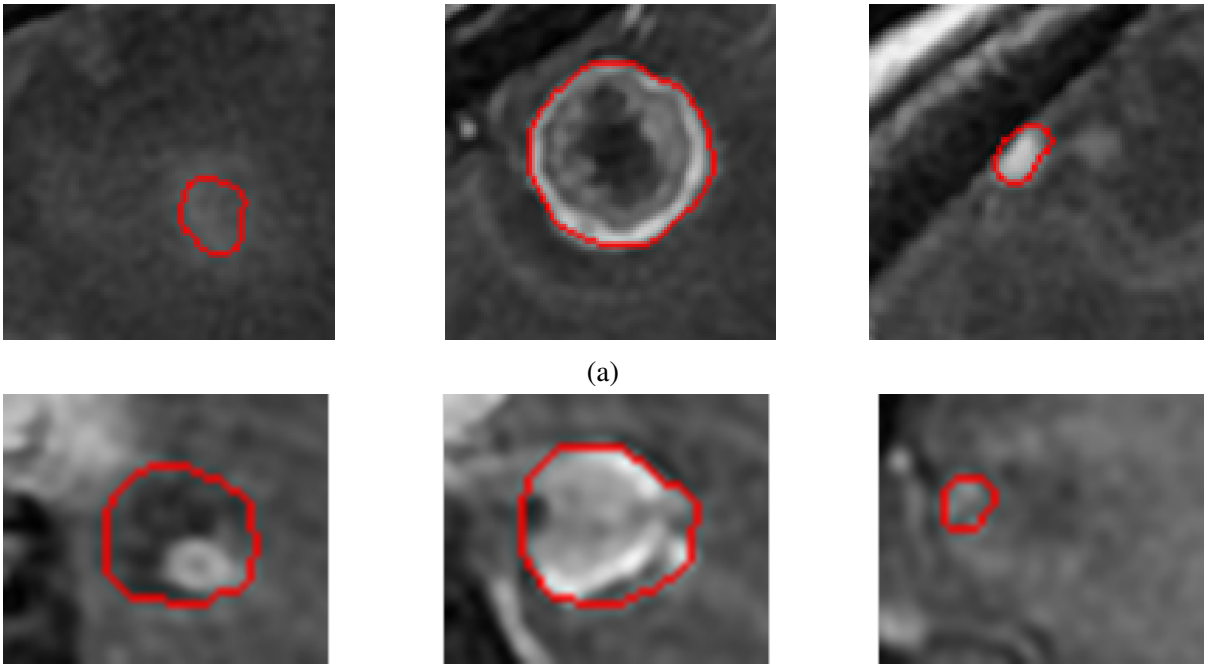
(a) (b) (c)

Figure 1.4: (a): Tumor with white pixels inside. (b): Tumors with different intensity patches. (c): Tumor with dark pixels inside and gray pixels closer to the boundary. Red contour is the tumor boundary.

Inconsistent appearance

Different regions of a tumor can look different even they are within one slice of a tumor.

Different slices often have significant different appearances. Figure 1.5 shows two examples with such situation. This adds to the difficulty of methods based on appearance models.



(a)

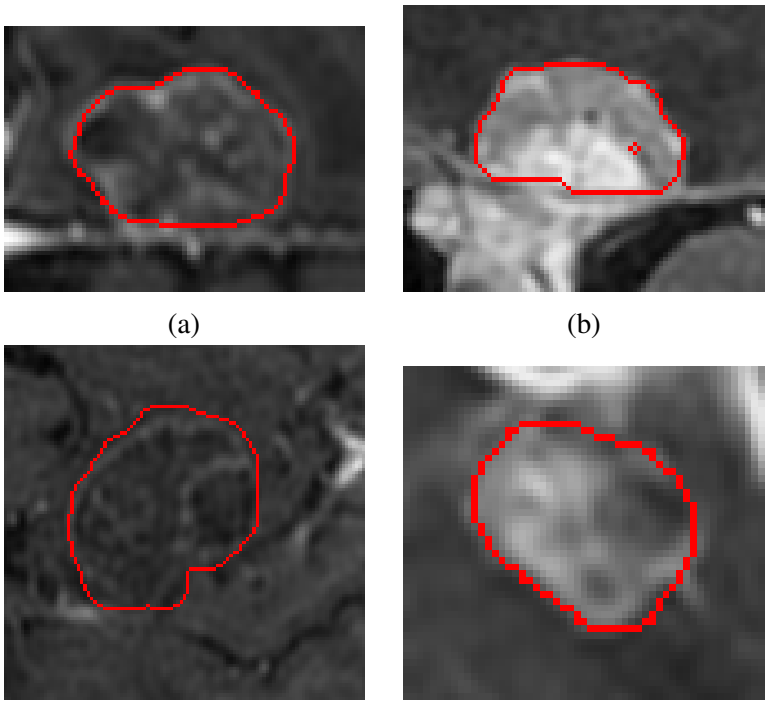
(b)

Figure 1.5: Examples for inconsistency in appearance. Each row is 3 slices from one tumor. Left and right columns are slices near the bordering of tumors. The first row shows a tumor with totally different appearance at border (dark gray and white respectively) and a mixture of dark, gray and white at the center. The bottom row is a tumor with a mixture of dark and bright near the left border and gray near the right border, while mostly bright at the center. Red contour is the tumor boundary.

Similarity between tumor and normal tissue

Sometimes, tumors are very similar to their surrounding normal tissues. In this case, there is a large intensity overlap between the tumor and the background. If strong intensity boundaries exist between the tumor and normal tissue, one can still extract the tumor with the cue of image intensity edges. Otherwise, the tumor is hard to locate even with human vision. In many cases, a tumor could be distinct from most of the normal tissue, but is very similar to some other brain part, as illustrated on the top right in Figure 1.6. This kind of tissues often obstructs the segmentation process due to their strong boundaries.

Besides these three main challenges explained above, brain tumors are hard to segment due to the fact that they vary greatly in size, position and shape.



(a) (b)

(c) (d)

Figure 1.6: (a) and (c) are tumors similar to healthy tissues while without distinct boundary. (b): tumor grows from the healthy tissue that is also different from other tissues. (d): tumor which has a large intensity overlapping with the healthy tissue, moreover, the boundary is hard to localize. Red contour is the tumor boundary.

1.3 Our approach

In this thesis, we use a semi-automatic approach that requires user interaction to provide us rough information about the position and size of the tumor. Specifically, we ask the user to locate the slice which has the largest tumor area and put a bounding box around it. This requires 2 clicks, one for the top left corner and the other for the bottom right corner of the box. In addition, we ask the user to locate the first and the last slice containing the tumor, and click roughly in the center of the tumor in these slices. This requires two more clicks. This interaction should be simple for medical experts because they just locate the tumor instead of labeling all the tumor pixels slice by slice. In addition, it is very minimal in terms of user clicks, just four clicks are required. At the same time, these four clicks give a lot of information for our algorithm. We know the rough size of the tumor, and a rough location. The semi-automatic approach has other advantages as follows

- User interaction gives us the location of the tumor and eliminate the process of locating the tumor from the large amount of voxels of the brain.
- The bounding box provided by the user simplify the problem of segmenting a tumor from the entire brain to segmenting a tumor from volumes within a bounding box whose size is comparable to the tumor. This can accelerate the segmentation process by several orders of magnitude.
- No training data is required and this reduce the chance that algorithm would over-fit to a certain data set.

After we acquire user assitance, we formulate the segmentation task as a binary labeling problem, that is the problem of assigning each pixel either a label ”OBJ” (object, or tumor) or label ”BKG” (background, or normal tissue). We address the binary labeling problem in the energy minimization framework. And energy formulation framework is advantageous because we can incorporate the desired constraints of the problem into the energy function in principled way. Afterwards, a globally optimal or approximately optimal minimum of the energy function is sought.

In energy minimization framework, an energy function is defined based on what kind of constraints we have. The most commonly used energy function for binary image segmentation with graph cuts combines regional information (appearance) with boundary information (length of the object segment, often weighted by image gradient). Typically, the regional information is encoded in the regional term, or the data term, which is another name for the regional term and the one we most often use in this thesis. The data term is used to insure the pixel being

assigned to some label actually is a good fit to the appearance model associated with this label. Usually this means that the color of the pixel (or some other feature at that pixel) is a likely color to be observed according to the appearance model. The boundary constraint encourages most nearby pixels to be assigned the same label, or, in other words, it encourages the boundary of the object segment to be small. Usually, the boundary length is weighted by the inverse intensity gradient, so that the segment boundaries are encouraged to align with the strong intensity edges.

The energy function with the regional and boundary terms can be usually optimized efficiently and exactly with the graph cut algorithm [6]. However, in our case, the appearance models are not known exactly. Therefore, we follow the approach of grabcut [20] to include appearance estimation as part of the energy function. The energy with appearance model included is no longer easy to optimize, in fact, it' becomes NP-hard [20]. We follow the same approach of block coordinate descent [20] which iteratively estimates the segmentation, keeping the appearance models fixed, and then re-estimates the appearance models, keeping the segmentation fixed. This approach is guaranteed to decrease the energy, and so convergence is guaranteed as well, albeit to a local minimum. In practice, this approach works quite well and is very popular in computer vision segmentation problems.

We found that the standard regional and boundary terms do not provide enough constraints to solve the brain tumor segmentation reliably. Therefore, we incorporate additional constraints to help improve the performance of our algorithm. In particular, we incorporate a certain shape prior, called the "star shape prior". This shape prior ensures that the object (tumor) region is connected, without wholes, and has shape not too far from convex. In addition, an energy function with the star shape prior can still be globally optimized with graph cuts.

While the shape of our objects is improved with the star shape prior, we find that the object region tends to be undersegmented, that is the object region has area consistently smaller than the ground truth, especially in the slices close to the ends of the volume. To help obtain segmentations with a larger object region, we incorporate volumetric balooning into the energy function. Volumentric ballooning just adds unary (linear) terms into the energy function, and, again, does not change the computational complexity of the algorithm.

Lastly, we noticed that many tumors consist of an "inner" and "outer" parts with distinct appearance, with inner part being strictly inside the outer part. We incorporate this knowledge into segmentation energy. This is no longer a binary problem, but can still be optimized globally and efficiently. A rough flow chart of our algorithm is shown in Figure 1.7.

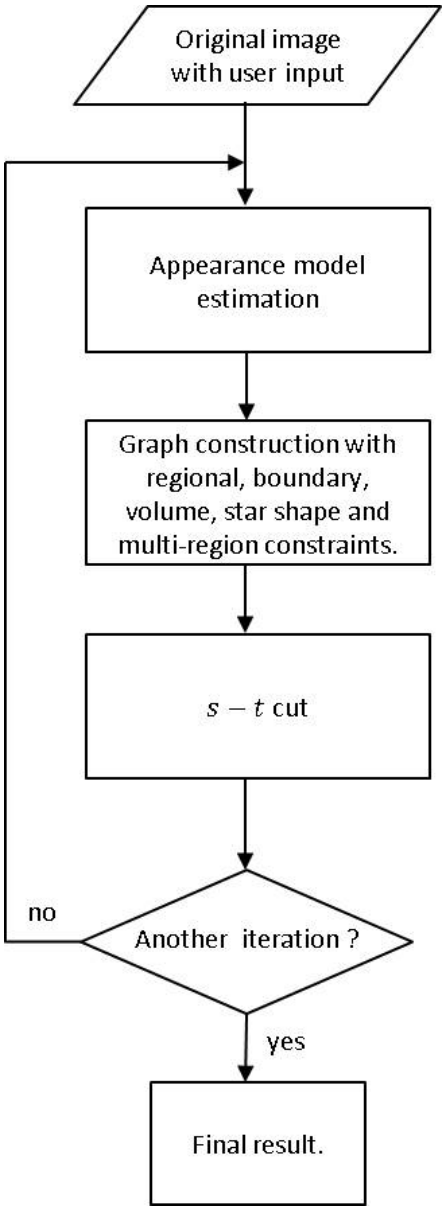


Figure 1.7: The flow chart of our algorithm

1.4 Outline of this thesis

The thesis is organized as follows: **Chapter 2** is an overview of the energy minimization framework with graph cuts and brief introduction to binary image segmentation using graph cuts. In **Chapter 3** work of *interactive graph cuts segmentation, Grabcut, star shape prior* and *multi-region framework* is analyzed. **Chapter 4** gives a detailed explaination of our graph cuts based segmentation algorithm and all the constraints we add to the energy function. We also show how we make full use of the user input in our algorithm. **Chapter 5** presents the experimental results and parameter selection. We also compare different methods with each other. The best performance we can get is around 87% F-measure. We give a conclusion of the thesis and future work in **Chapter 6**.

Chapter 2

Overview of the energy minimization framework

Many computer vision problems can be posed as a labeling problem which involves assigning a label to each pixel. The task of brain tumor is definitely a labeling problem where we need to assign tumor pixels and healthy tissue pixels with different labels. Energy minimization framework has long been a popular way to convert the labeling problem to the problem of minimizing the labeling energy. The energy minimization framework usually involves two steps: energy function construction and energy minimization. Energy function provides a way to map the solution to a real number, and it measures the goodness of a solution. A good solution should map to a lower energy and a bad solution should map to a higher energy. The second step energy minimization is a process of searching the optimal solution in the solution space.

2.1 Segmentation as a Labeling Problem

Many vision problems can be posed as labeling problems in which the solution to a problem is a set of labels assigned to image pixels or features. Image denoising, stereo correspondence, restoration or smoothing can all be modeled as labeling problems [23], where the goal is to assign each pixel a label that is associtated with a value (true intensity or disparity) based on the observation (the input).

To describe a labeling problem, one needs a set of labels and a set of sites, which is usually the set of all image pixels or all volume voxels in case of a 3D medical volume.

Let

$$\mathcal{P} = \{1, 2, ..., n\}$$

be a set of pixels. The inter-relationship between pixels is maintained by a so-called neighborhood system. For example, in a two dimensional image, a pixel with its top, bottom, left and right pixels are neighboring pixels. The set of all neighboring pixel pairs is usually denoted by $\mathcal{N}$.
Let

$$\mathcal{L} = \{1, 2, ..., m\}$$

be a set of possible labels. A label usually represents intensities, disparities or semantic properties. In this thesis, since we are performing binary image segmentation, our label set is $\mathcal{L} = \{0, 1\}$ where 0 stands for the background, or healthy tissue, and 1 stands for the brain tumor, or object tissue.
Let f_i denotes the label assigned to pixel i, and let f be the set of all pixel-label assignments, that is

$$f = \{f_1, ..., f_n\}$$

If all pixels take values from the same label set, then the set of all possible labeling is

$$\mathbb{F} = \mathcal{L} \times \mathcal{L} ... \times \mathcal{L}$$

The total number of different labeling could be as huge as $\mathcal{L}^{\mathcal{P}}$. Among all the possible labelings, there are only a few of that are optimal in terms of some properly defined criterion, which measures the goodness (or inversely, the cost) of the solution. The criterion is encoded in the energy function in energy minimization framework, which is addressed in next section.

2.2 Energy function

Energy function maps a possible solution to a real number and provides us a way to measure the goodness of a solution. In general, a good energy function maps a desirable solution to a lower energy value and undesirable solution to a high energy value. In computer vision, an energy function usually takes the following form

$$E(f) = E_{data}(f) + \lambda \cdot E_{prior}(f) \tag{2.1}$$

where $E_{data}(f)$ called data term or regional term. It usually measures the disagreement between the data d and the labeling f. The term $E_{prior}(f)$ measures the disagreement between the labeling and prior knowledge. The constant λ controls the relative importance between *data energy* and *prior energy*. One can encode any prior knowledge in the energy function. For example we can assume that the set of pixels are such that most nearby pixels have the same

label, or we can incorporate various shape priors [24] on the set of assigned labels. One can add different priors by adding the energy of those terms into the energy function. The most commonly used energy function in computer vision consists of data term and smoothness term

$$E(f) = E_{data}(f) + \lambda \cdot E_{smooth}(f) \tag{2.2}$$

where the smoothness prior assumes that image quantities to be estimated varies smoothly everywhere or almost everywhere except at boundaries where it may change abruptly [23]. In case of binary energy segmentation we have only two labels. So smoothness prior encourages most neighboring pixels to have the same labels, thus minimizing the boundary length, where a boundary is a place where label changes. Usually, the data term takes the following form

$$E_{data}(f) = \sum_{p \in \mathcal{P}} D_p(f_p) \tag{2.3}$$

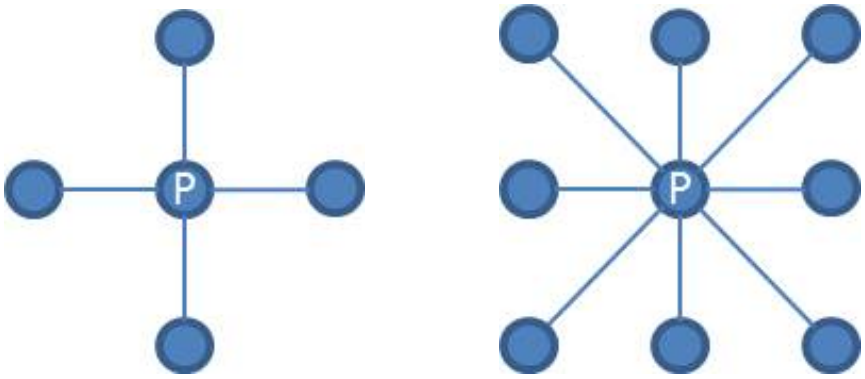
where $D_p(f_p)$ measures how the label f_p fits into the observation at pixel p. This form assumes that observations at each pixel are independent, this assumption is reasonable for most image analysis problems [23]. The data energy forces the solution to be close to the observation. In most cases, it depends on the level of the noise of the observation.
A popular formulation of the smoothness term in computer vision is of the form

$$E_{smooth}(f) = \sum_{\{p,q\} \in \mathcal{N}} V_{\{p,q\}}(f_p, f_q) \tag{2.4}$$

where $\mathcal{N}$ is the neighborhood system that describe how pixels interacts with each other. Mostly, the 4- or 8- neighborhood system are used. Figure 2.1 illustrates the two neighborhood systems. As we assume that the labeling should be spatially coherent, the smooth term gives penalty to labeling where neighboring pixels have different labels. As we also encourage discontinuity in intensity or image features across different labels, we want the smooth term only penalizes the labeling that assigns different labels to similar neighboring pixels. Normally, the more similar two neighboring pixels are, the larger we have to pay for assigning them to different labels.

2.3 Optimization with graph cuts

During the past years, graph cuts have been increasingly popular in the field of computer vision due to its efficiency in solving a wide variety of low-level computer vision problems, such as image restoration, stereo correspondence, image segmentation and many other computer vision problems that can be formulated in term of energy minimization [23]. The application of graph



(a) 4-neighborhood system (b) 8-neighborhood system

Figure 2.1: Representation of 4- and 8- neighborhood system in images

cuts in computer vision was first introduced by Greig et al [10] in binary image restoration. In [10], Greig et al. shows how graph cuts can find the exact maximum of a posterior probability of a degraded binary image. Since then, the graph cuts theory has received great interest from researchers to make it more general and efficient. In this section we address on the graph cuts theory and its application to energy minimization.

2.3.1 Overview of graph cuts

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be a weighted graph where $\mathcal{V}$ is a set of vertices (nodes) and $\mathcal{E}$ is a set of edges which connect vertices in $\mathcal{V}$. An $s-t$ cut $C = (\mathcal{S}, \mathcal{T})$ is a partition of $\mathcal{V}$ such that $s \in \mathcal{S}$ and $t \in \mathcal{T}$, where s and t are two distinguished vertices called the terminals. Usually, s is called a *source* and t is called a *sink*. More specifically, a cut C is a subset of edges $C \subseteq \mathcal{E}$ such that when edges in C are removed from the graph $\mathcal{G}$, $\mathcal{V}$ is partitioned into two disjoint sets $\mathcal{S}$ and $\mathcal{T}$. In figure 2.2, the thickness represents the weight of an edge. A possible cut for this graph is shown with dashed curve, which partitions vertices on the left of the cut to *source* and vertices on the right to *sink*.

The cost of a cut C is defined as

$$|C| = \sum_{e \in C} w_e \tag{2.5}$$

A minimum cut is a cut with the minimum cost. The max-flow min-cut theorem states that the minimum cost (capacity) of a cut in graph $\mathcal{G}$ is equal to the maximum amount of flow passing from the source to the sink [13]. Based on max-flow min-cut equivalence, one can find the minimum cut with a large number of fast max-flow algorithms. For solving the max-flow problem, there are Ford-Fulkerson Algorithm [13], push-relabel algorithm [15].

In [5], Boykov et al. presented two algorithms based on graph cuts that efficiently find a local

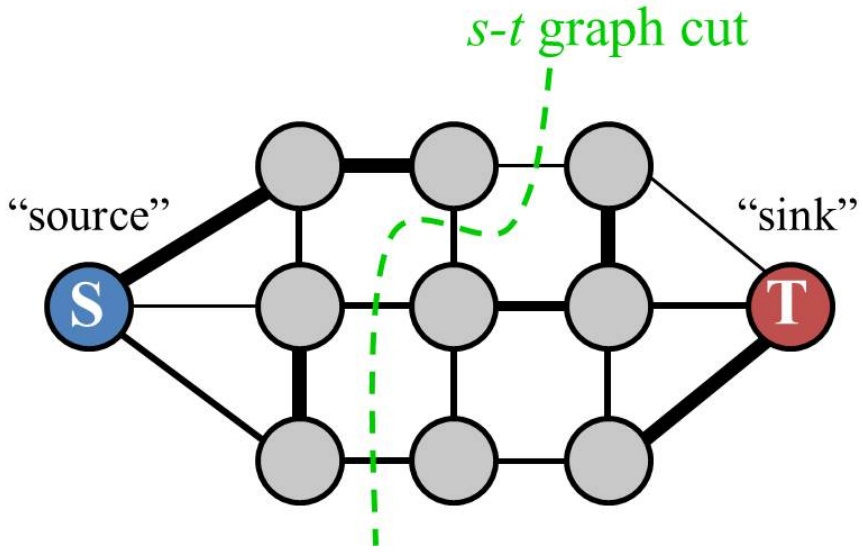


Figure 2.2: A s-t cut on graph with two terminals $\mathcal{S}$ and $\mathcal{T}$. [Image credit: Yuri Boykov]

minimum with respect to two types of large moves, namely α *expansion* move and $\alpha - \beta$ *swap* move. These algorithms find good approximate solutions by iteratively decreasing the energy on appropriate graphs. They also prove that the expansion algorithm finds a solution within a known factor of the global minimum and the swap algorithm handles more general energy functions. What's more, their algorithms are significantly faster than other standard algorithms when applying to computer vision problems. The work of this thesis is based on the implementation of their algorithm.

2.3.2 What energy can be optimized via graph cuts

The energy minimization problem can be solved via graph cuts by constructing a specialized graph for the energy function to be minimized such that the minimum cut on the graph also minimizes the energy. However, not all the energy function can be solved by graph cuts. In [16], Kolmogorov proved that for binary labeling problems, if the energy function is submodular, then the global optimum energy can be obtained by computing the minimum cut on a properly constructed graph.
The energy function is submodular if

$$V_{pq}(0,0) + V_{pq}(1,1) \leq V_{pq}(0,1) + V_{pq}(1,0) \tag{2.6}$$

where $\{0,1\}$ is the set of labels and V_{pq} is the smooth cost for neighboring pixels p, q.

2.4 Binary image segmentation with graph cuts

In this thesis, we are doing binary image segmentation where our task is to extract tumor from the healthy tissue. In this section, we address on the popular energy function for binary image segmentation and the construction of the corresponding graph.

2.4.1 Energy function for binary image segmentation

In the case of binary image segmentation, only two labels are used. $\mathcal{L} = \{0, 1\}$ where 0 stands for background, or in our case, healthy tissue, and 1 stands for foreground, or tumor in our case. Our energy is based on the most commonly used energy function before adding other constraints

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \lambda \cdot \sum_{\{p,q\} \in \mathcal{N}} V_{pq}(f_p, f_q) \tag{2.7}$$

A commonly used penalty function for smoothness term in binary image segmentation takes the following form

$$V_{pq}(f_p, f_q) = w_{pq} \cdot \delta(f_p, f_q)$$

where

$$\delta(f_p, f_q) = \begin{cases} 1 & \text{if } f_p \neq f_q \\ 0 & \text{otherwise} \end{cases}$$

Therefore in our application, we have

$$V_{pq}(0, 0) = 0$$
$$V_{pq}(1, 1) = 0$$

so our energy function satisfy the submodularity, thus can be optimized with graph cuts. More detailed formulation of our energy is addressed in **Chapter 4**.

2.4.2 Energy minimization with graph cuts

This section address on how to construct a graph so that the result of min-cut corresponds to the global optimal solution of the energy function. To apply graph cuts in the energy minimization framework, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ whose capacity of minimum cut $|C|$ is equal to the optimal energy $E(f^*)$ should be constructed.

Let $p \in \mathcal{P}$ represents a pixel, the set of nodes $\mathcal{V} = \mathcal{P} \cup \{s, t\}$. s and t are the terminal nodes source and sink respectively. Each pixel is a node in the graph and it connects to the source and the sink, and the links are called *t-link* and are denoted by $\{p, s\}$ and $\{p, t\}$. Edges between

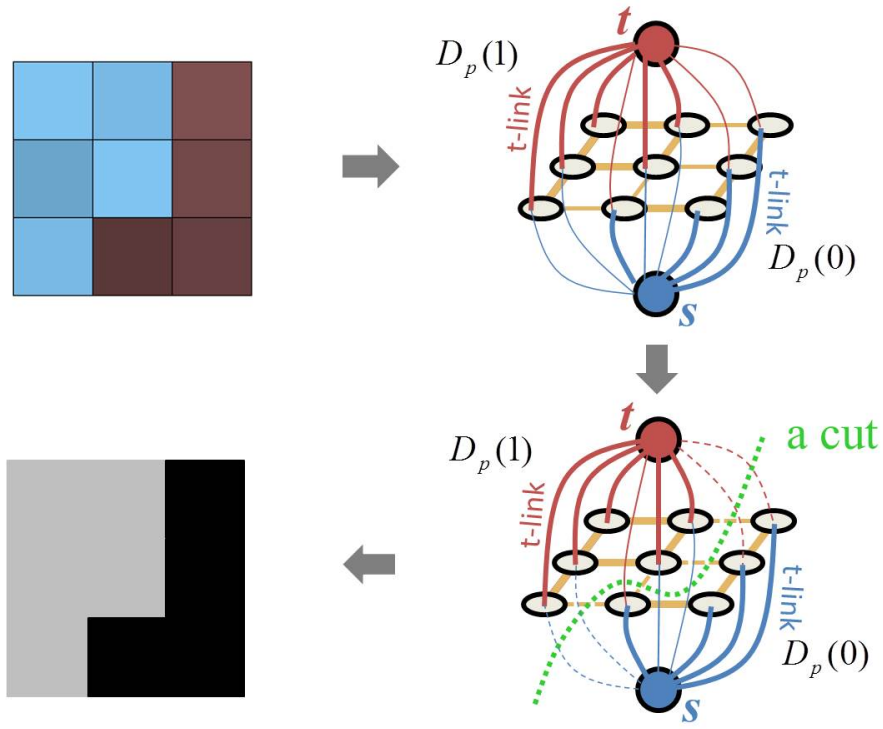


Figure 2.3: A simple bianry segmentation for a 3×3 image. (top-left): Image to be segmented; (top-right): Graph with pixels as nodes and s,t as terminals; (bottom-right) A cut for the graph, dashed links are cutted edges; (bottom left): Segmentation result. [Image credit: Yuri Boykov]

neighboring pixels are called *n-link* and is denoted by $\{p, q\}$. Therefore, the set of edges in the graph is

$$\mathcal{E} = \mathcal{N} \cup \{\{p, s\}, \{p, t\}\} \tag{2.8}$$

A graph with energy in Equation 2.7 can be constructed by assigning the edges with the following values

Table 2.1: Weights for edges in the graph

edge	weight	for
$\{p, s\}$	$D_p(1)$	$p \in \mathcal{P}$
$\{p, t\}$	$D_p(0)$	$p \in \mathcal{P}$
$\{p, q\}$	w_{pq}	$\{p, q\} \in \mathcal{N}$

An toy example of binary segmentation is shown in figure 2.3.

Chapter 3

Related Work

Image segmentation has long been a fundamental and well-studied problem in computer vision. Many algorithms have been proposed for accurate, efficient segmentation. Segmentation algorithms can be roughly divided into two groups, according to their goals. In the first group are the algorithms that try to segment an image into patches so that each patch has coherent color, texture, etc. These algorithms do not, in general, explicitly model the likelihood of a patch to correspond to some object in the world. In the second group are the algorithms that try to segment an image into patches corresponding to objects or object parts. For the algorithms in the first group, they segment an image according to the cues of color, boundary, texture, etc. Segmentation task of the algorithms in the second group is much harder because they have to model object appearance, which could be a complex combination of colors, boundaries and textures. Therefore, algorithms in the second group usually rely on learning object appearance from a labeled dataset.

In this thesis, we are interested in segmenting the input into two regions, the foreground and the background. This is usually referred to as binary image segmentation and there are many algorithms addressing this problem. Inspired by the success of interactive graph cut segmentation, we build a algorithm that adds star shape constraint, multi-region constraint and volume constraint to the basic formulation. In this chapter, we will give a general summary of work related to our approach in terms of interactive binary graph cut based segmentation.

3.1 Interactive graph cuts segmentation

Interactive segmentation has been a successful approach to extract objects of interest out from the background. Interactive segmentation is popular in many domains since automatic segmentation is just too difficult to achieve, due to the ambiguity of the image data, image noise and imaging artifacts, etc. [6].

The first graph cuts based interactive segmentation was proposed by Boykov and Jolly in 2001 [6]. It requires user to impose hard constraints for segmentation by indicating certain pixels as seeds that are absolutely have to be part of the object (marked by "$\mathcal{O}$") and certain pixels as seeds for the background (marked by "$\mathcal{B}$"), i.e. they absolutely have to be part of the background. In their work, $\mathcal{L} = \{0, 1\}$ where 0 stands for object of interest and 1 stands for the background. The energy function they use is also the most popular one

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \lambda \cdot \sum_{\{p,q\} \in \mathcal{N}} V_{pq}(f_p, f_q) \tag{3.1}$$

where they set hard constraint to pixels marked as seeds in data term

$$D_p(1) = \begin{cases} K & \text{if } p \in \mathcal{O} \\ 0 & \text{if } p \in \mathcal{B} \end{cases}$$

and

$$D_p(0) = \begin{cases} K & \text{if } p \in \mathcal{B} \\ 0 & \text{if } p \in \mathcal{O} \end{cases}$$

where $\mathcal{O}$ and $\mathcal{B}$ denote the subsets of pixels marked as object and background, and K is a large constant.

The marked pixels also provide a cue for the appearance of the object and background. In [6], Boykov and Jolly make double use of the user provided seeds to get the appearance model of object and background. They use intensities of pixels marked as seeds to get histograms for object and background intensity distributions: $\Pr(I|\mathcal{O})$ and $\Pr(I|\mathcal{B})$. Then they set the data cost D_p as negative log-likelihoods of the intensity probability

$$\begin{aligned} D_p(1) &= -\ln \Pr(I_p|\mathcal{O}) \\ D_p(0) &= -\ln \Pr(I_p|\mathcal{B}) \end{aligned}$$

In their work, the second term in Equation 3.1 is

$$V_{pq}(f_p, f_q) = w_{pq} \cdot \delta(f_p, f_q) \tag{3.2}$$

where

$$\delta(f_p, f_q) = \begin{cases} 1 & \text{if } f_p \neq f_q \\ 0 & \text{otherwise} \end{cases}$$

and

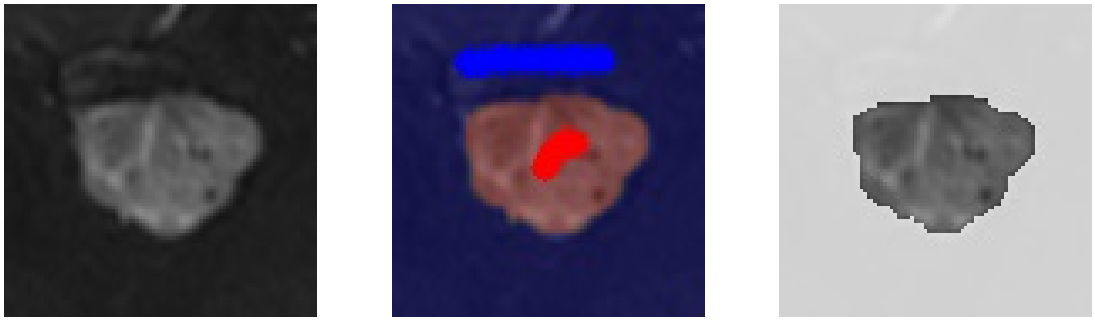
$$w_{pq} \propto exp(-\frac{(I_p - I_q)^2}{2\sigma^2}) \cdot \frac{1}{dist(p, q)} \tag{3.3}$$

The function used in Equation 3.3 measures the dissimilarity between two neighboring pixels and aligns object boundary to large contrast edges. When the difference between intensity of p and q is much smaller compared to σ, this function penalizes a lot. However, if pixels are very different, i.e. $|I_p - I_q|$ is much larger compared to σ, the penalty is small. Intuitively, this function corresponds to the distribution of noise among neighboring pixels of an image and σ can be estimated as "camera noise" [6].

With the above energy function, one can construct a graph by assigning the weights according to the following table

Table 3.1: Weights for edges in graph.

edge	weight	for
$\{p, q\}$	$\lambda \cdot w_{pq}$	$\{p, q\} \in \mathcal{N}$
$\{p, \mathcal{S}\}$	$-\ln \Pr(I_p \mid \mathcal{B})$	$p \in \mathcal{P}, p \notin \mathcal{O} \cup \mathcal{B}$
	K	$p \in \mathcal{O}$
	0	$p \in \mathcal{B}$
$\{p, T\}$	$-\ln \Pr(I_p \mid \mathcal{O})$	$p \in \mathcal{P}, p \notin \mathcal{O} \cup \mathcal{B}$
	0	$p \in \mathcal{O}$
	K	$p \in \mathcal{B}$



(a) A slice of tumor (b) User input. (c) Segmentation result

Figure 3.1: An example of Boykov's interactive segmentation: red is foreground brush and blue is background brush.

Note that further user editing is possible after one segmentation. The user editing acts as a correction of current segmentation by adding more seeds for the object and background or changing the previously marked seeds. A globally optimum segmentation can be efficiently computed after the user editing. Thus users can get any desired segmentation as they want, with enough interaction.

3.2 Grabcut

Grabcut [20] introduced by Rother et al. is another graph cuts based interactive segmentation system. Different from the method proposed by Boykov et al. [6], grabcut uses a rectangle containing the object as the user interaction, which might require fewer clicks, depending on the image. In addition, instead of fixing the appearance model for the foreground/background, the grab cut energy optimizes over the appearance as well as the segmentation. This leads to an NP-hard problem, which grabcut addresses by block coordinate descent, iterating the estimation of segmentation using graph cuts with fixed appearance, and then, optimizing the appearance, keeping the segmentation fixed. Because of applying graph cuts iteratively, the grabcut approach is called, sometimes, iterative graph cuts.

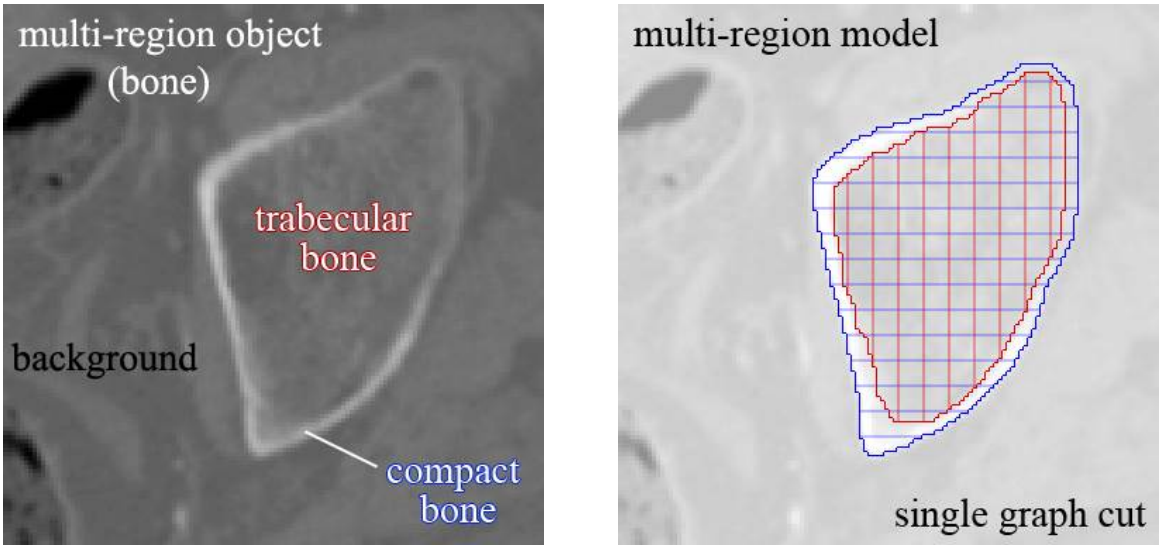
In grabcut, the initial user input is a rectangle which completely contains the object (see Figure 3.2). The rectangle gives an incomplete initial labeling of the images with label ”object” inside and ”background” outside. Appearance models of object and background, which are specified by Gaussian mixture models of pixels' color, are computed from the initial labeling. It is assumed that pixels outside the rectangle are background, therefore this part of image is hard constrained to the label ”background”. There is no hard constraint inside the rectangle because pixels inside can either be background or object. After the initial segmentation is performed, the labeling is updated and the appearance models are re-estimated. This is a typical EM-style algorithm which involves 3 steps: 1, initialize the appearance model; 2, segment the image with current appearance model; 3, update the appearance model with the segmentation and iterate step 1 and 2 until convergence. Further user corrections are also possible in grabcut.

(a) Image with green rectangle as user input (b) Segmentation result after 3 iterations

Figure 3.2: An example of grabcut segmentation.

3.3 Multi-region object segmentation

Graph cuts based multi-region framework is introduced by Delong et al. [11] with the motivation of extracting objects with spatially distinct regions. However, most data modeling methods, such as intensity histogram used in [6] and Gaussian mixture model used in grabcut, ignore the spatial distribution of colors within an object, and might fail to get a segmentation with coherent parts. Delong et al. proposed a method to encode geometric interactions between distinct region+boundary models. In our dataset, we could also see some tumors with such structure and this is our motivation of employing multi-region constraint in our application.
The left of Figure 3.3 is one example for distinct region+boundary object from Delong et al.'s paper. We could see that the boundary of the bone is much brighter than the interior. With normal graph cuts segmentation, it's hard to segment the gray inside with the bright outside together unless we incorporate other terms such as ballooning or encourage edges to go from bright to dark in this particular case. The result of using multi-region model is shown on the right of Figure 3.3, we can see that the boundary and the interior of the bone have been successfully segmented into two regions.



(a) Bone (b) Segmentation

Figure 3.3: Object with distinct region+boundary appearance and the segmentation using multi-region constraint. Figure from [11].

In [11], three geometric interactions are proposed to help to deal with multi-region object associated problem.

Containment. Region B must be inside region A, perhaps with repulsion force between boundaries.

Exclusion. Regions A and B cannot overlap at any pixel, perhaps with repulsion force between boundaries.

Atraction. Penalize the area $A - B$, exterior to B, by some cost $\alpha > 0$ per unit area. Thus A will prefer not to grow too far beyond the boundary of Y.

We will explain the multi-label segmentation, multi-region energy function and the construction of the graph in the rest of this section.

3.3.1 Multi-label segmentation

Before going into the detail of multi-region object segmentation, we have to introduce the multi-label segmentation in this section. This is because the task of multi-region object segmentation is modeled as a mulit-label segmentation. In general, an $n-$region model potentially has 2^n corresponding labels.

To illustrate multi-label segmentation with graph cuts, let us consider the example with only two pixels in Figure 3.4. To employ the graph cuts in segmentation with n labels, one approach is to construct a graph with $n - 1$ layers. Each layer contains the same number of nodes, i.e. the number of pixels to be labeled in the image. Each column represents nodes associated with one pixel. Nodes in the same layer are constrained under the similar smoothness term to the binary segmentation. With a linear label cost function (see Figure 3.4), each node connects to the node in the same column in next layer. We call such links inter-layer links and the cost for such a connection is specified by the linear label cost on the left of Figure 3.4. The more two label differ, the larger is the penalty. The max-flow algorithm allows us to find a minimum cut on the graph that seperate the nodes into two disjoint subsets. The label of a pixel is determined according to which inter-link is cut.

To perform multi-region segmentation, a directed graph consisting of an ordered set of layers, with one layer per region, is constructed. Each layer by itself has the same structure as in the binary graph cut problem formulation. The inter-layer arcs are specified by the geometric interactions introduced in next section.

3.3.2 Multi-region energy

In Delong's multi-region framework, the label set $\mathcal{L}$ denotes the set of region indices. There are $|\mathcal{L}| \times |\mathcal{P}|$ binary variables, and they are indexed by $\mathbf{x}_p^i$ over pixels $p \in \mathcal{P}$ and over regions $i \in \mathcal{L}$. $\mathbf{x}_p^i = 1$ denotes that the pixel p is interior to region i. The notation $\mathbf{x}_p$ denotes a vector of all variables that correspond to pixel p, one for each of the $|\mathcal{L}|$ regions, and the notation $\mathbf{x}^i$

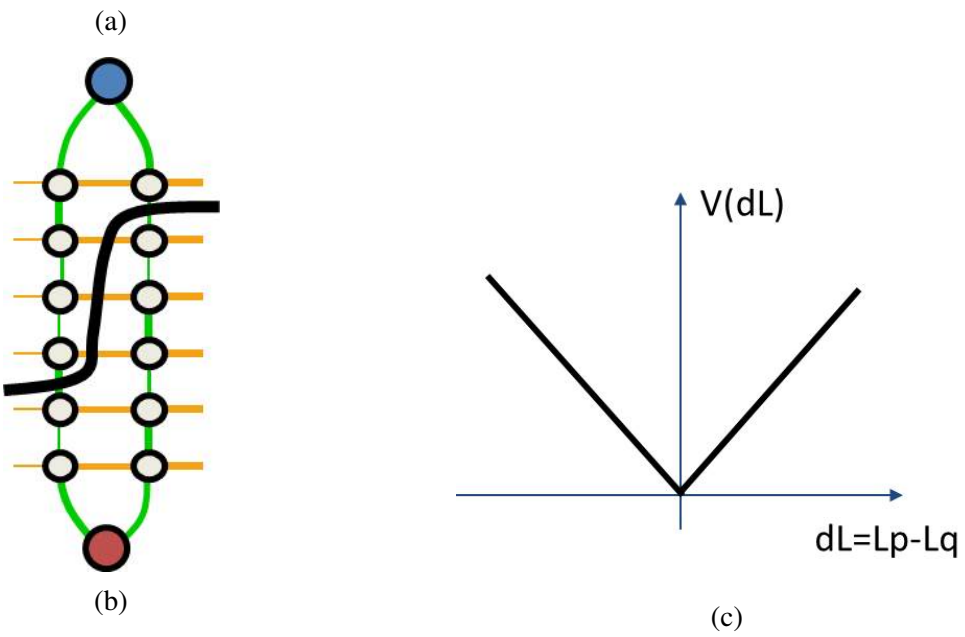


Figure 3.4: (a): Graph for multi-label segmentation with two pixels. Each column represents nodes associated with one pixel. The weights of links from one layer to another are specified by the label cost on the right. (b): Label cost for label difference. Image credits: Yuri Boykov.

refers to all variables of a particular region i. $\mathbf{x}_p = \mathbf{0}$ means the pixel p is not interior to any region, therefore p is considered "background".
The multi-region energy takes the overall form

$$E(\mathbf{x}) = \sum_{p\in\mathcal{P}} D_p(\mathbf{x}_p) + \sum_{i\in\mathcal{L}} V^i(\mathbf{x}^i) + \sum_{\substack{i,j\in\mathcal{L}\\ i\neq j}} W^{ij}(\mathbf{x}^i,\mathbf{x}^j) \tag{3.4}$$

where D_p defines a cost of pixel p for every *combination* of regions. The smoothness term in Equation 3.4 consists of an inner-region interaction term V^i and an inter-region interaction term W^{ij}. The inner-region interaction term V^i is the standard smoothness term for variables representing the region i and it takes the following form

$$V^i(\mathbf{x}^i) = \sum_{\{p,q\}\in\mathcal{N}^i} V^i_{pq}(\mathbf{x}^i_p,\mathbf{x}^i_q)$$

where each neighborhood $\mathcal{N}^i$ defines nearest-neighbor grid connectivity.
The inter-region interaction term W^{ij} encodes all geometric interactions between region i and j and is indexed over both region pairs (i, j) and pixel pairs (p, q)

$$W^{ij}(\mathbf{x}^i,\mathbf{x}^j) = \sum_{\{p,q\}\in\mathcal{N}^{ij}} W^{ij}_{pq}(\mathbf{x}^i_p,\mathbf{x}^j_q)$$

$\mathcal{N}^{ij}$ is the set of all pixels pairs (p, q) at which region i is assigned some geometric interaction with region j. $(p, p) \in \mathcal{N}^{ij}$ is allowed because it refers to separate variables of different regions.

3.3.3 Geometric interaction

Recall that there are three geometric interactions in multi-region framework: containment, exclusion and attraction. Consider Figure 3.5 with the energy terms listed. The basic "i contains j" interaction means region j has to be inside region i. An example is shown in Figure 3.5 that B is interior to A, which means A contains B. If p is interior to region j, then all pixels q that defined under the inter-region neighborhood system N^{ij}_{pq} should be interior to region i. The term $W^{ij}_{pq}(0, 1) = \infty$ prohibits any cut that violates this constraint. We can restrict $\mathcal{N}^{ij}$ to index over only pixel pairs (p, p), therefore if p is interior to region j, then p has to be interior to region i because region j is inside region i. So we have $W^{ij}_{pp}(0, 1) = \infty$ to prohibit labeling p as interior and exterior to region i at the same time. The attraction interaction penalizes the area difference between i and j which is rational because generally the boundary region would not be too large compared to the inside of an object. We do not provide details on exclusion term in Table 3.2 because it is not submodular and we do not need it for the work in this thesis. Details of transforming this term to a graph-cut optimizable term can be found in the original paper [11].

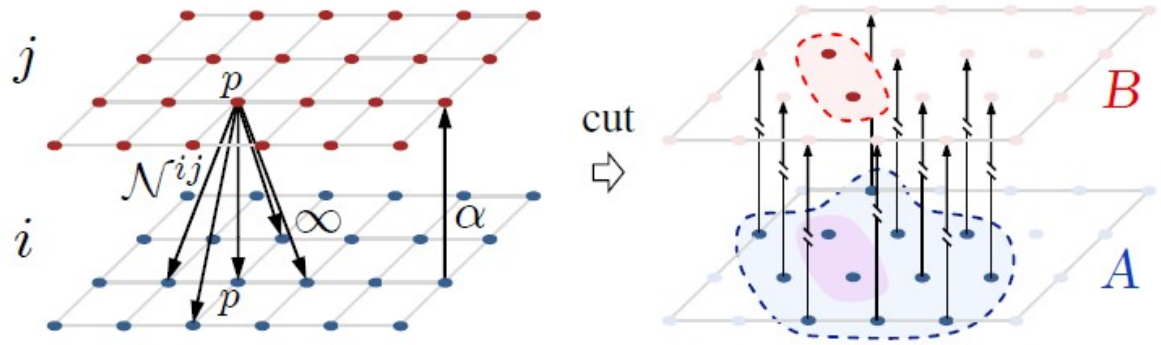


Figure 3.5: Left: Graph for region layers $i, j \in \mathcal{L}$, each layer uses 4-neighborhood system. The figure shows a subset of inter-region neighborhood $\mathcal{N}^{ij}_p$ and arcs for containment and attraction interaction. The ∞-cost arcs enforce a 1-pixel margin between region boundaries. The α-cost arcs attract the outer boundary by penalizing only the area $A - B$. Right: A cut shows how the interior and boundary of object are separated through the binary labeling. Figure from [11]

Table 3.2: Geometric interaction terms. Table from [11].

i contains j		
$\mathbf{x}_p^i$	$\mathbf{x}_q^i$	W_{pq}^{ij}
0	0	0
0	1	∞
1	0	0
1	1	0

i excludes j		
$\mathbf{x}_p^i$	$\mathbf{x}_q^i$	W_{pq}^{ij}
0	0	0
0	1	0
1	0	0
1	1	∞

i attracts j		
$\mathbf{x}_p^i$	$\mathbf{x}_q^i$	W_{pq}^{ij}
0	0	0
0	1	0
1	0	α
1	1	0

3.3.4 Multi-region data term

Multi-region framework ultilizes the negative log-likelihood as the data term. One advantage of multi-region framework is that it splits the mixed data model into separate data models, each with a distinct data distribution (See Figure 3.6). Consider the case where the object has only two distinct region A and B, and A contains B, see Figure 3.6. With the appearance model, we can construct the data term using the weights below, where

Table 3.3: Weights for data term. Table from [11].

$\mathbf{x}_p^A$	$\mathbf{x}_p^B$	D_p
0	0	$-\log\Pr(Bg\|I_p)$
0	1	K
1	0	$-\log\Pr(A\|I_p)$
1	1	$-\log\Pr(B\|I_p)$

K is a large value that prohibits the configuration $x_p^A = 0$, $x_p^B = 1$. Generally, we set $W_{pp}^{AB} = \infty$. The three likelihoods are driven by the image data itself, with respect to each region.

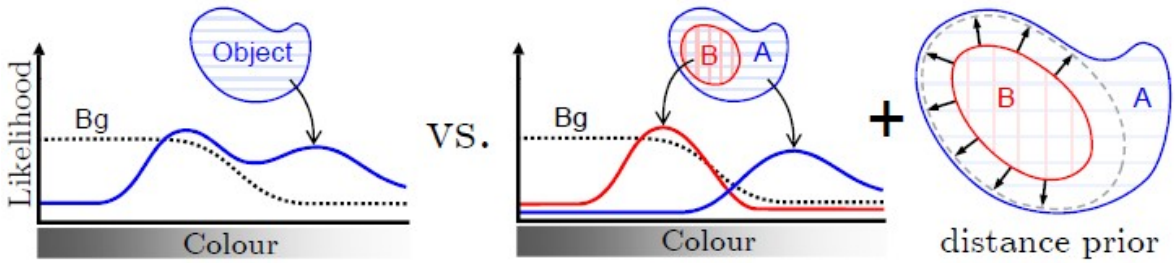


Figure 3.6: Left: Object model corresponding to Figure 3.3. Right: Object interior (B) and boundary (A) model corresponding to the final result in Figure 3.3. Figure from [11].

3.4 Star shape prior

Recall that the basic energy function in Equation 2.1 consists of a data term and a prior term. The prior term incorporates some prior knowledge about the object, such as shapes. So far, we have introduced smoothness prior, multi-region prior. Another possibility is a shape prior which reduces ambiguity by ruling out all segments that violate the prior.

Veksler proposed a generic shape prior for graph cut segmentation called star shape prior [24]. The star shape prior is based on simple geometric properties of an object instead of a shape of a specific object class. The definition given in the original paper [24] is: " A *star* shape is defined with respect to a center point c. An object has a star shape if for any point p inside the object, all points on the straight line between the center c and p also lie inside the object." The paper gives some examples with star shape in Figure 3.7. The star shape prior needs a user

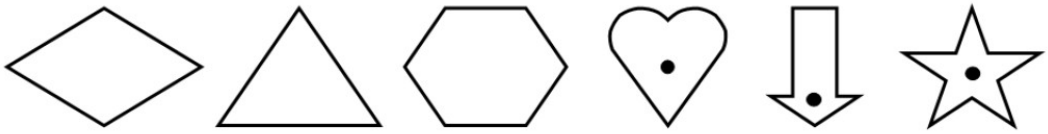


Figure 3.7: Star shape examples. First three shapes are star shapes with respect to any inside point as the center. Last three shapes are star shapes with respect to only some inside center points. Figure from [24]

input seed as the star center. In [24], the author claims that most that are star shaped, have more than one possible center, therefore, the user does not have to be very careful in choosing the center. In a special case of a convex object, the user can choose any point inside as the center. The advantage of the star shape constraint is that it can be directly incorporated in the graph cut segmentation [24], without adding much to the complexity of either implementation or the running time of the algorithm. The following part of this section will explain how to add the star shape constraint.

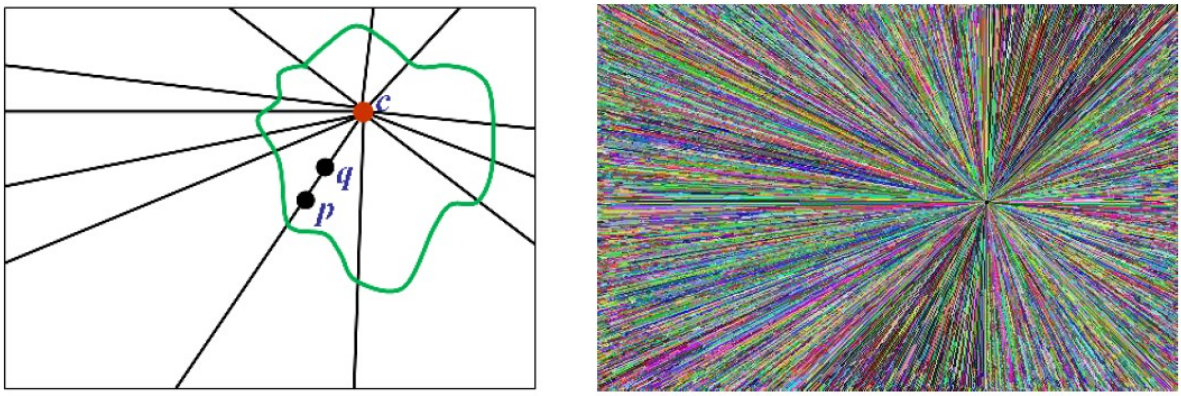
3.4.1 Star-shape constraint term

Let c denote the center of the star shape and assume it is known beforehand. To illustrate the idea of the star shape constraint, consider Fig. 3.8. Pixels p and q are on the line passing through c, and q lies between c and p. If p is assigned label 1, all the pixels between c to p (on the line) including q are also assigned 1. Therefore the star shape constraint term S_{pq} takes the following form

$$S_{pq}(f_p, f_q) = \begin{cases} 0 & \text{if} \quad f_p = f_q, \\ \infty & \text{if} \quad f_p = 1 \quad \text{and} \quad f_q = 0, \\ \beta & \text{if} \quad f_p = 0 \quad \text{and} \quad f_q = 1 \end{cases} \tag{3.5}$$

Note that the shape constraint in Equation 3.5 does not need to be placed between all pairs of pixels on a line passing through c. It is enough to put an S_{pq} only between neighboring pixels [24]. β can be set to negative values, if desired, and this encourages a longer segmentation boundary.

To add the star-shape constraint, one needs to discretize the image pixels into lines passing through the center c. The star shape algorithm starts from a random pixel p in the image and discretize the line starting from center c to p.



(a) (b)

Figure 3.8: (a) An example of a star shape is in green. The center of the star is marked with a red dot c. Let p and q be pixels on the line passing through c, and q lies between c and p. If p is labeled as the object, then q must be also labeled as the object; (b) Discretized lines are displayed with random colors. Figure from [24].

With the shape constraints, the energy function becomes

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \lambda \sum_{\{p,q\} \in \mathcal{N}} V_{pq}(f_p, f_q) + \sum_{\{p,q\} \in \mathcal{N}} S_{pq}(f_p, f_q) \tag{3.6}$$

where V_{pq} takes the same form as Equation 3.2.

3.4.2 Bias towards longer segment boundaries

A graph cut has a well known bias towards shorter boundaries unless we have a strong data term. The relative importance between data term and smoothness term is control by the parameter λ. Energy function with larger λ pays more for smoothness cost and results in smaller

segment. Conversely, energy with a smaller λ relies more on the power of data term, which is generally specified by appearance model of object/background. The amount of user input in star shape implementation can be really small: a single user marked seed. This single seed is for object and a pixel wide border of the image is assumed to be the background. This is not enough to construct reliable models for the object and background, therefore the data term has to be weighted low relative to the smoothness term.

In the absence of a strong data term, the result is more likely to shrink, thus a bias towards a longer boundary is needed. In [24], this bias is incorporated by setting β in Equation 3.5 to a negative number. The star shape term in Equation 3.6 acts as the penalty for the length of the segment boundary so that longer segmentation boundaries decrease the energy more as compared to the shorter boundaries. To get the optimal β, a ratio energy is defined as follows

$$E_{ratio}(f) = f_{weight} + \beta \cdot f_{length} \tag{3.7}$$

where f_{weight} is is the sum of w_{pq} weights on the segmentation boundary, and f_{length} is the length of the boundary. The optimal β can be found by searching for the minimum ratio region, where the optimum value of E_{ratio} is 0. Some examples of Veksler's star shape segmentation are shown in Figure 3.9.

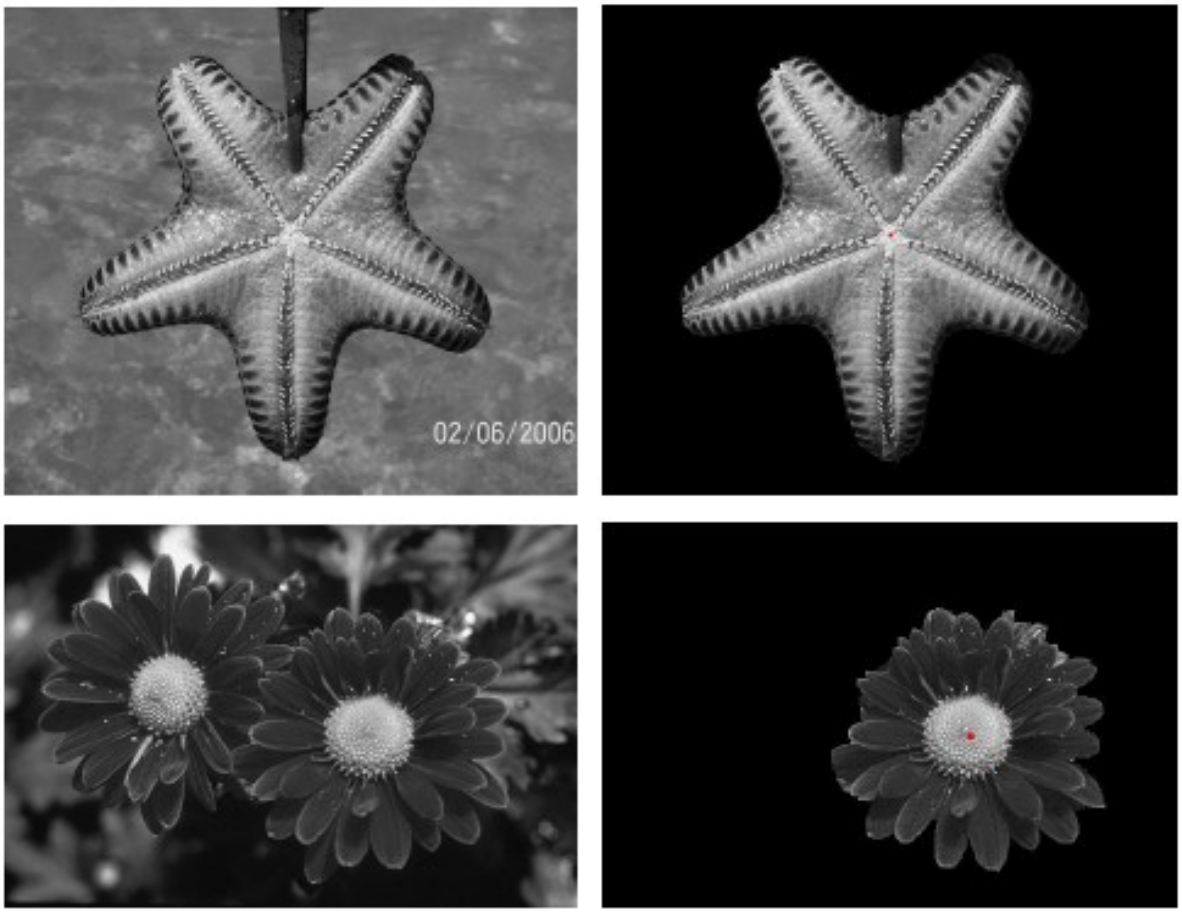


Figure 3.9: Left: Original images. Right: Segmentation results with star shape. Red dots are user marked centers for objects. Figure from [24].

Chapter 4

Interactive brain tumor segmentation

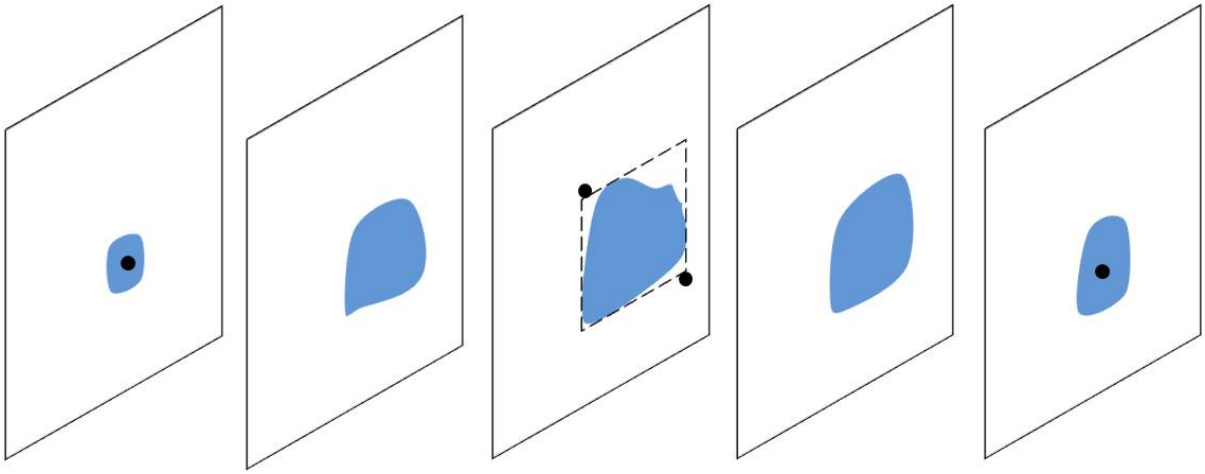
Brain tumor segmentation plays an critical role in diagnosis and treatment decision. In this thesis, we use a semi-automatic approach that requires user interaction to provide us rough information about the position and size of the tumor. Specifically, we ask the user to specify a rectangle that completely enclose the tumor in all slices. This requires 2 clicks, one for the top left corner and the other for the bottom right corner of the rectangle. In addition, we ask the user to locate the first and the last slice containing the tumor, and click roughly in the center of the tumor in these slices. This requires two more clicks. This type of user interaction is illustrated in Figure 4.1. This interaction should be simple for medical experts because they just locate the tumor instead of labeling all the tumor pixels slice by slice. In addition, it is very minimal in terms of user clicks, just four clicks are required. At the same time, these four clicks give a lot of information for our algorithm. We know the rough size of the tumor, and a rough location. While automatic segmentation might be the ultimate goal, it is a difficult task for the problem at hand, due to the following reasons:

1. It is difficult to automatically locate the position of the tumor inside the whole brain volume due to large variations in appearance and size of tumors among different patients.

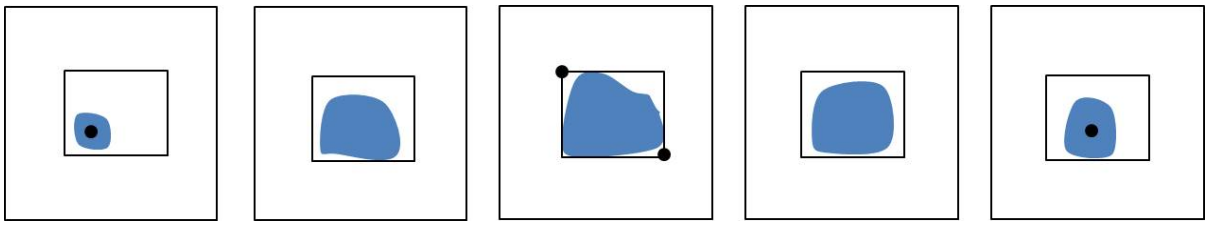
2. Even if a tumor can be localized successfully, automatic segmentation is still a hard problem because the appearance of the tumor and the background has a large overlap.

3. The boundaries between the tumor and the normal tissue are often fuzzy and thus make the task of reliable automatic segmentation more difficult.

In addition, a semi-automatic approach makes sense in this domain since a medical expert has to validate the results of a segmentation algorithm for possible errors. Thus our task is to design a program that is helpful for the medical expert to achieve an accurate segmentation faster.



(a) Black dots: user clicks. The two clicks in the central slice specify a rectangle that completely enclose the tumor in all slices. Other clicks specify the first and last slice respectively and are assumed to be the center.



(b) 3D bounding box from the four user clicks.

Figure 4.1: An example for user interaction and 3D bounding box obtained from user clicks.

We formulate the task of brain tumor segmentation in energy minimization framework and use the graph cut algorithm to find the solution. A common energy function that most often used for binary image segmentation with graph cuts contains a data term and a smoothness term. The data term encourages the foreground/background regions to obey the preferred appearance models, and the smoothness term encourages a smooth object boundary. In addition to these common terms, we also add and evaluate a star shape prior constraint [24] and volumetric ballooning. The star shape prior allows to incorporate a prior on the tumor region that it is connected and cannot have shapes widely different from convex. The ballooning helps to encourage the brain tumor region to be of more appropriate volume. We also noticed that the appearance of some tumors have some regularities, such as a darker region is often surrounded by a lighter region. We decided to take the advantage of it by incorporating the multi-region framework of [11]. This framework is more general than binary segmentation.
In this section, we explain our approach and our energy function in details for the basic energy, the star shape, the ballooning, and the multi-part components.

4.1 Overview of our approach

Figure 4.2 shows the flow chart of the segmentation procedure. For simplicity, we do not put star shape constraint and volume ballooning here. We begin with an MRI data with a 3D bounding box that completely contains the tumor (See Figure 4.1). The 3D bounding box is obtained from the 2D bounding box the user provided in one of the slices and the first and the last slice click. Since the outside of the bounding box contains only the healthy tissue, it does not need to be a part of the segmentation algorithm. Therefore we crop the input data only to contain the 3D box we constructed. The cropped data is the input data for our system. We approach optimization like in the grabcut [20] framework. Namely, we do not assume fixed foreground/background models but rather estimate them from the data. Thus, the appearance models are part of the variables that need to be estimated during segmentation. The bounding box gives us an initial approximate labeling of the volume with label object (tumor) inside the box and label background (healthy tissue) outside. We estimate the initial appearance model for object/background from the initial labeling. Specifically, for the foreground model, we take the histogram from the box, and for the background, we take pixels outside the box in a band of size 1/4 of the user provided box. Energy minimization is performed on the graph constructed based on current appearance model. Then we update the appearance model from the segmentation and iterate the graph cut algorithm to get a new result. We do not perform more iterations than two because the energy tends to converge after just two iterations. Just as in grabcut, the energy is guaranteed to decrease during the iterations.

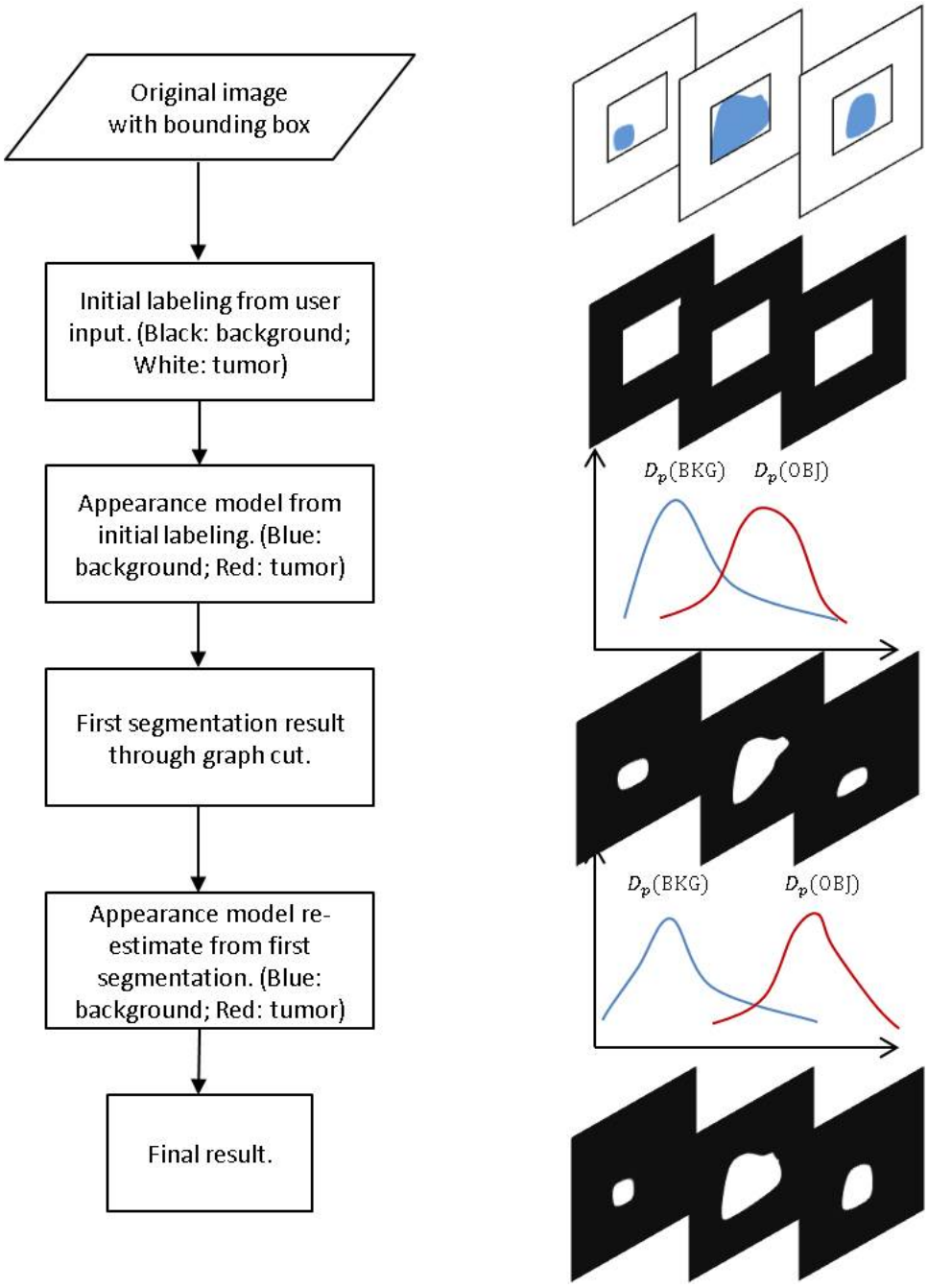


Figure 4.2: Algorithm flow of our binary segmentation.

4.2 Data term

Recall that the data term measures how well the labeling fits into the observed data. In binary segmentation, it is a cost function that measures the cost of assigning pixel p to "object" and "background". The most popular cost function for data term reflects on how the intensity of pixel p fits into a known appearance model of the object and background. We utilize the negative log-likelihood of the appearance model, therefore equation 2.3 is as follows

$$\begin{aligned} E_{data}(f) &= \sum D_p(f_p) \\ D_p(\text{BKG}) &= -log\,\Pr(I_p \,|\, \text{BKG}) \\ D_p(\text{OBJ}) &= -log\,\Pr(I_p \,|\, \text{OBJ}) \end{aligned} \tag{4.1}$$

where $\mathcal{P}$ is the set of all image pixels, f is a labeling of all image pixels, I_p is the intensity of pixel p, f_p is the label for pixel p. In our application, the label set $\mathcal{L} = \{\text{BKG}, \text{OBJ}\}$ where BKG represents background and OBJ represents object. The term $\log \Pr$ is called the likelihood function of appearance model which gives the probability of intensity I_p conditioned on different labels. And it measures the cost for assigning a label to a pixel. Large probability value means better fit to the label, therefore, in this case, the negative log function returns a small cost for assigning the label to a pixel. The negative log-likelihood has a major limitation in many applications, since it assumes the appearance models are known before the segmentation takes place. However, in our application, this is not a problem because we estimate the appearance models as part of the energy formulation, and we have a good initialization for the appearance models from the user provided box. Figure 4.3 shows how to construct part of the graph corresponding to the data term. In the rest of this section, we focus on different appearance modeling methods.

4.2.1 Global intensity histogram

An intensity histogram is the most common and one of the simplest ways to model region appearance. The normalized intensity histogram, that is intensity histogram divided by the number of pixels in the region being modeled, is an approximation of the probability distribution of the pixel colors within an object. If the range of intensities is not large, a histogram can be built on the raw intensity values. If the range of intensities is high, usually a binning of intensities is performed, both for computational efficiency and to make sure the histogram is not too sparse, which would be unreliable. In a binned histogram, the range of intensities is divided, usually equally, in the desired number of bins. Then, a histogram counts, for each bin, the number of pixels with intensities in that bin. If bins are used, then the normalized

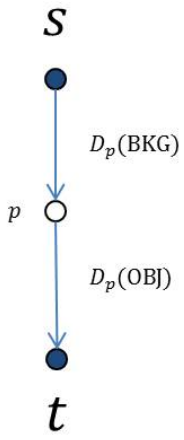
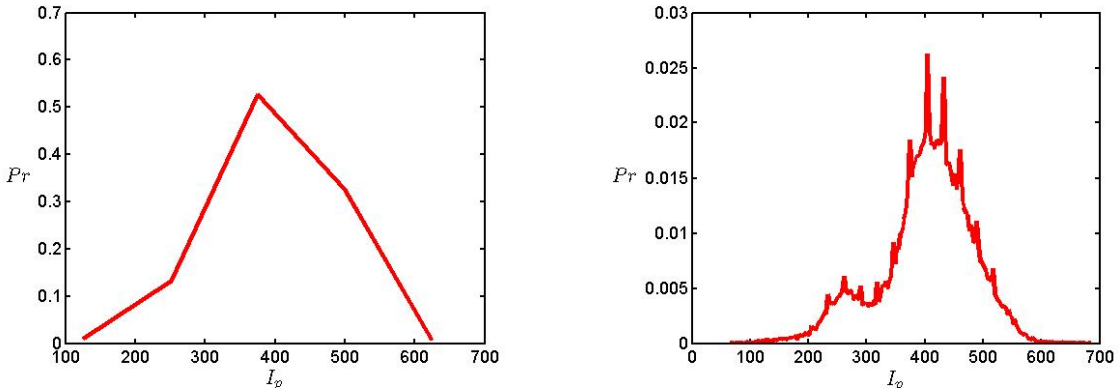


Figure 4.3: T-links weights for a pixel p in the graph.

histogram approximates the distribution of the center of each bin.

The number of bins is crucial in constructing a histogram. In figure 4.4, a histogram with a large bin number is too noisy while the one with a small bin number is over-smoothed compared to the true distribution.

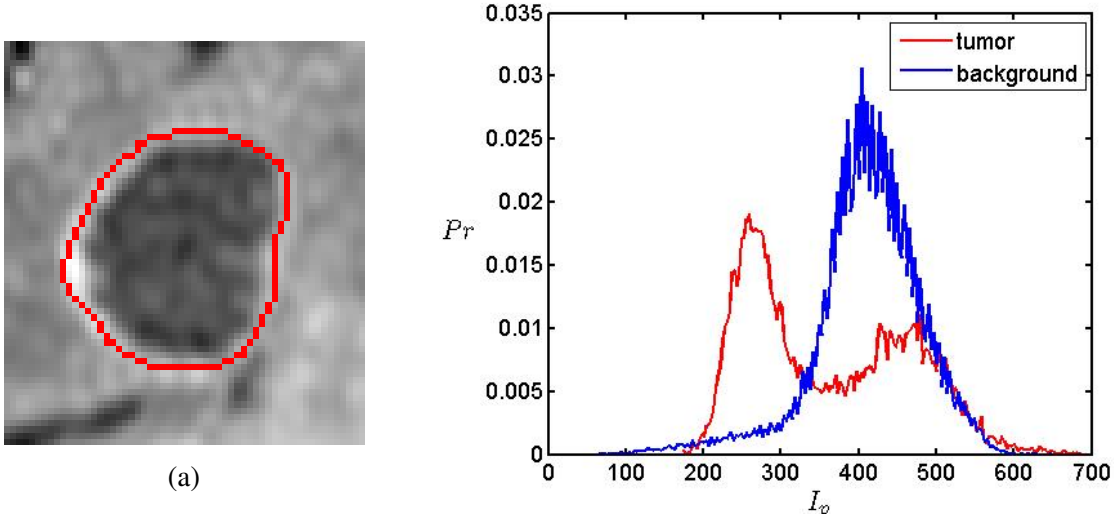


(a) Histogram with 5 bins

(b) Histogram with 200 bins

Figure 4.4: Histograms with different number of bins. *Pr* represents the probability, I_p represents the pixel intensity. (a): Histogram with too few bins that over-smoothed the true distribution; (b) Histogram with too many bins that results in noisy probability distribution.

In our application, we build an initial histogram of tumor and background regions by binning pixels inside the box and in a band outside the box respectively. Figure 4.5 shows the histogram of background and tumor for one volume.



(a) (b)

Figure 4.5: (a): One slice of the tumor, red curve indicates the boundary of tumor. (b):Global intensity distribution for tumor (red) and background (blue) *Pr* represents the probability, I_p represents the pixel intensity.

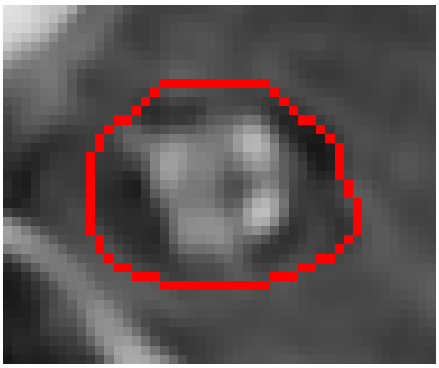
4.2.2 Local intensity histogram

Since the appearance of tumor can vary significantly across different volume slices, and tumor on slices that are far from the central slice only takes up a small proportion of the whole volume, the global intensity histogram usually fails to accurately capture the appearance of tumor on these slices. This motivates us to use a separate histogram for each slice. That is, each slice has its own appearance model for the foreground and the background. As before, we get the initial models from the initialization box. But now we take the projection of the 3D box on the particular slice for which modeling is performed. We call this approach is ”local histogram” since each slice has its own, local, histogram.

4.2.3 Parzen window density estimation

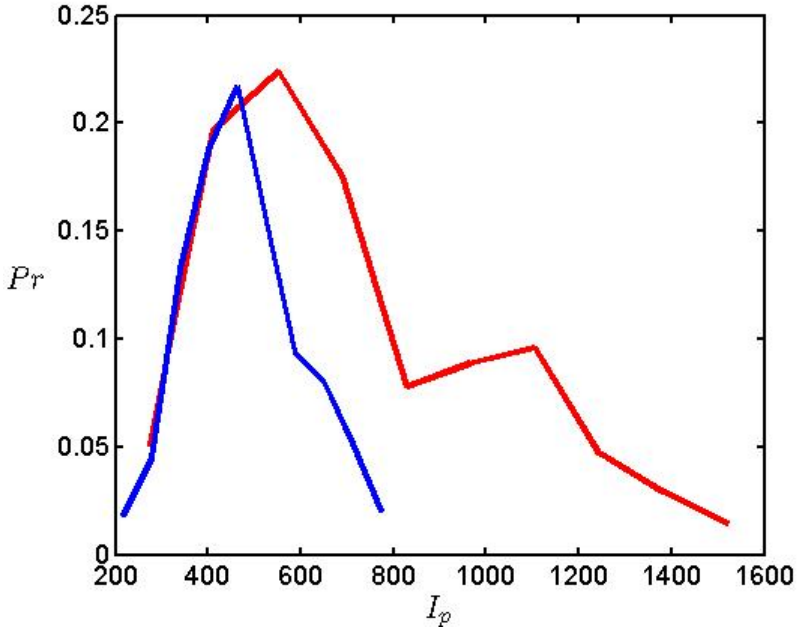
As can be observed from Figure 4.7, the intensity histogram can be quite noisy and thus we may benefit from smoothing it. A histogram is noisy if there are not enough samples per bin to give a reliable estimate. For example, in Figure 4.7, the image size is 47x39, which is rather small, but the number of distinct intensities is 572. This is too large of a range for a reliable estimate with 623 bins. To get a smoother distribution, we use kernel density estimation, also called the parzen window estimation.

Parzen window was proposed by Emanuel Parzen [18] in the early 1960s. It is a *non-parametric*



(a) One slice of tumor located at the center of the volume

(b) The same tumor on the slice that is more close to the end of the volume.



(c) Local histogram of tumor showed in (a) and (b). Red:(a); Blue: (b).

Figure 4.6: Local intensity distribution for tumor on different slices. *Pr* represents the probability, I_p represents the pixel intensity.

probability density estimation. Non-parametric means that no specific density is assumed for the underlying intensity data. Typically, the parzen window estimate is defined as

$$P(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_n}K(\frac{x-x_i}{h_n}) \tag{4.2}$$

where $(x_1, x_2, ..., x_n)$ is the data whose density function is to be estimated. $K(x)$ is the kernel function which is always symmetric but not necessarily positive function that integrates to one. $h_n > 0$ is the smoothing parameter called the bandwidth and is typically chosen based on the number of available observations n. The kernel function $K(\cdot)$ can take any form depending on the application. We use the Gaussian kernel, which is the most popular one to use with Parzen windows. The parzen window estimate with the Gaussian kernel becomes

$$P(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h\sqrt{2\pi}}\exp(-\frac{1}{2}(\frac{x-x_i}{h})^2) \tag{4.3}$$

The selection of bandwidth h is critical in parzen window density estimation, and it exhibits a strong influence on the resulting estimation. Typically, the choice of h is based on the standard deviation of the data. For Gaussian kernel, the recommended choice of bandwidth is [21]

$$h = (\frac{4\hat{\sigma}^5}{3n})^{\frac{1}{5}} \approx 1.06\hat{\sigma}n^{-1/5} \tag{4.4}$$

where $\hat{\sigma}$ is the standard deviation of the samples.

4.3 Smoothness term

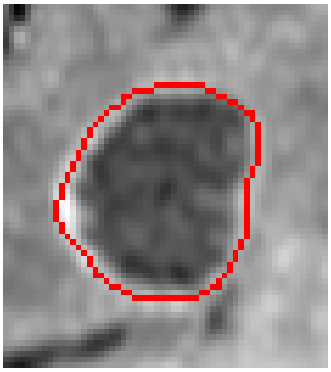
Smoothness or boundary term encourages spatially coherent labelings. A labeling is spatially coherent if most nearby pixels have the same label. That is the boundary, a place where labels switch, is encouraged to be of small length. This type of spatial coherence is appropriate for most natural objects, including tumors. Our smoothness term takes the following term

$$E_{smooth}(f) = \lambda \cdot \sum_{\{p.q\}\in\mathcal{N}} V_{pq}(f_p, f_q) \tag{4.5}$$

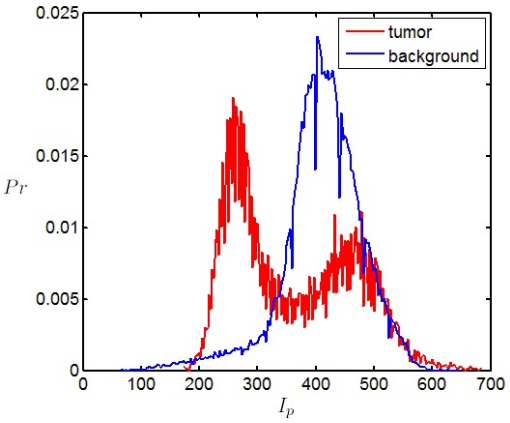
where

$$V_{pq}(f_p, f_q) = w_{pq} \cdot \delta(f_p, f_q) \tag{4.6}$$

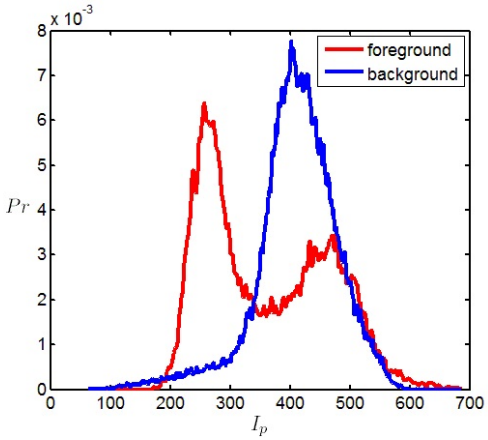
and



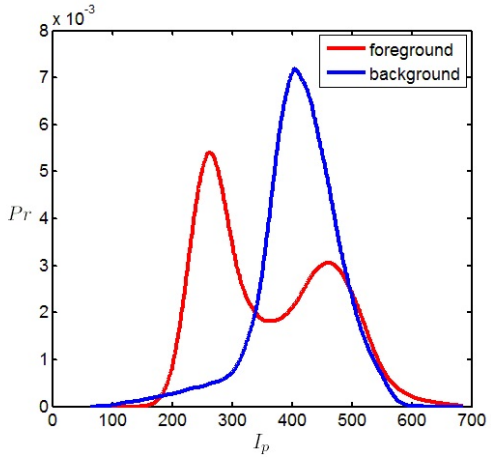
(a) One slice of tumor. Red contour is the boundary of the tumor.



(b) Global intensity histogram of the image



(c) Probability density with bandwith = $0.1h$



(d) Probability density with bandwith = h

Figure 4.7: Parzen window density estimation with different bandwidth *Pr* represents the probability, I_p represents the pixel intensity. We compute h according to formula 4.4.

$$\delta(f_p, f_q) = \begin{cases} 1 & \text{if } f_p \neq f_q \\ 0 & \text{otherwise} \end{cases}$$

If neighboring pixels are assigned the same label, there is no penalty. When $w_{pq} = constant$, the penalty for any pair of pixels to be assigned to different labels is the same and independent of how strong the intensity edge is between these pixels. To align the segmentation boundary with intensity edges, w_{pq} is typically a non-increasing function of $|I_p - I_q|$, where I_p is the intensity of pixel p. Normally, costs w_{pq} is based on local intensity gradient, Laplacian zero-crossing, gradient direction and other criteria. In this thesis, we use the following function [6]

$$w_{pq} = \exp(-\frac{(I_p - I_q)^2}{2\sigma^2}) \cdot \frac{1}{dist(p, q)} \tag{4.7}$$

The selection of σ is usually based on the level of noise of the image. We set σ as the average intensity difference between neighboring pixels in the image. This way, if pixels p and q have a larger than average intensity difference, then the cost w_{pq} will be relatively small. If pixels p and q have intensity difference smaller than average, w_{pq} will be more costly. Here $dist(p, q)$ is the distance between pixel p and q. Normally, for 4-neighborhood system, all neighboring pixels have the same distance 1, which means a distance of 1 pixel. In our application, because the volume is sampled with different sampling rate along the depth direction, compared to the x and y direction, the distance between pixels is not measured in pixels, but rather by the exact physical distance. We get the information about the exact physical distance from the input data, and this distance is different for different input volumes. Parameter λ in equation 2.2 controls the relative importance between the smoothness term and the other terms. Larger λ makes smoothness terms more important. In the limit, if λ is set to a too high value, the resulting segmentation will be either all pixels assigned to the foreground or all pixels assigned to the background.

We have different sampling rate along dimensions x, y and dimension z, $dist(p, q)$ for two pixels on the same slice is different from those on different slices. Therefore, we need two parameters to control the relative importance of the smoothness term, one is λ_{xy} for smoothness term on the same slice, the other is λ_z for smoothness term along dimension z. Then our smoothness energy becomes

$$E_{smooth}(f) = \lambda_{xy} \cdot \sum_{\{p,q\} \in \mathcal{N}_{xy}} V_{pq}(f_p, f_q) + \lambda_z \cdot \sum_{\{p,q\} \in \mathcal{N}_z} V_{pq}(f_p, f_q) \tag{4.8}$$

where $\mathcal{N}_{xy}$ is the 4-neighborhood system on a slice and $\mathcal{N}_z$ is the 2-neighborhood system for pixels on different slices. They are subsets of the 6-neighborhood system $\mathcal{N}$ we used in this

thesis, see Figure 4.8.

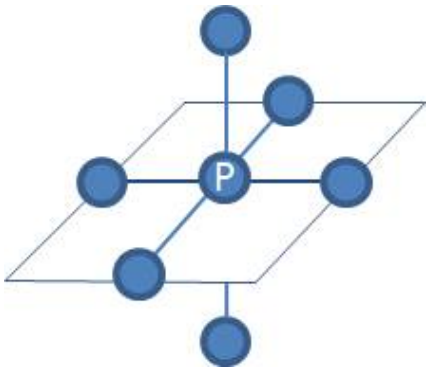


Figure 4.8: 6-neighborhood system. Each voxel has 4 connections to its immediate left, right, top, bottom neighbors within the slice, and also two connections to its closest neighbors in the previous and next slice.

4.4 Star shape constraint

So far, we have constructed the basic energy function containing the data term and the smoothness term. However, there is no connectivity information encoded in the energy function. The problem is that instead of getting one single segment, the segmentation result corresponding to the basic energy can give the foreground object broken into multiple components, object with holes inside, or object with unrealistic shape. The star shape constraint described in **Chapter 3** helps to rule out these implausible tumor object shapes.

The user provided bounding box gives us a initial labeling of the image, with label OBJ inside and BKG outside. Recall that the star shape prior had a parameter β that helps to incorporate a bias towards a longer segmentation boundary. We found that for our application, we do not need this bias. This is because we model appearance from the user provided box for the foreground and the outside of the box for the background. This gives us enough data to reliably estimate the appearance models. Thus we set β to zero and do not need to search for an optimal value of it, which saves a lot of computation. So our star shape term that we add to the energy function is:

$$S_{pq}(f_p, f_q) = \begin{cases} 0 & \text{if} \quad f_p = f_q, \\ \infty & \text{if} \quad f_p = 1 \quad \text{and} \quad f_q = 0, \\ 0 & \text{if} \quad f_p = 0 \quad \text{and} \quad f_q = 1 \end{cases} \tag{4.9}$$

We use the energy above to set star shape a hard constraint in our work. We have only ∞ cost for pixel pairs that violate the star-shape constraint and no additional cost for a segmentation

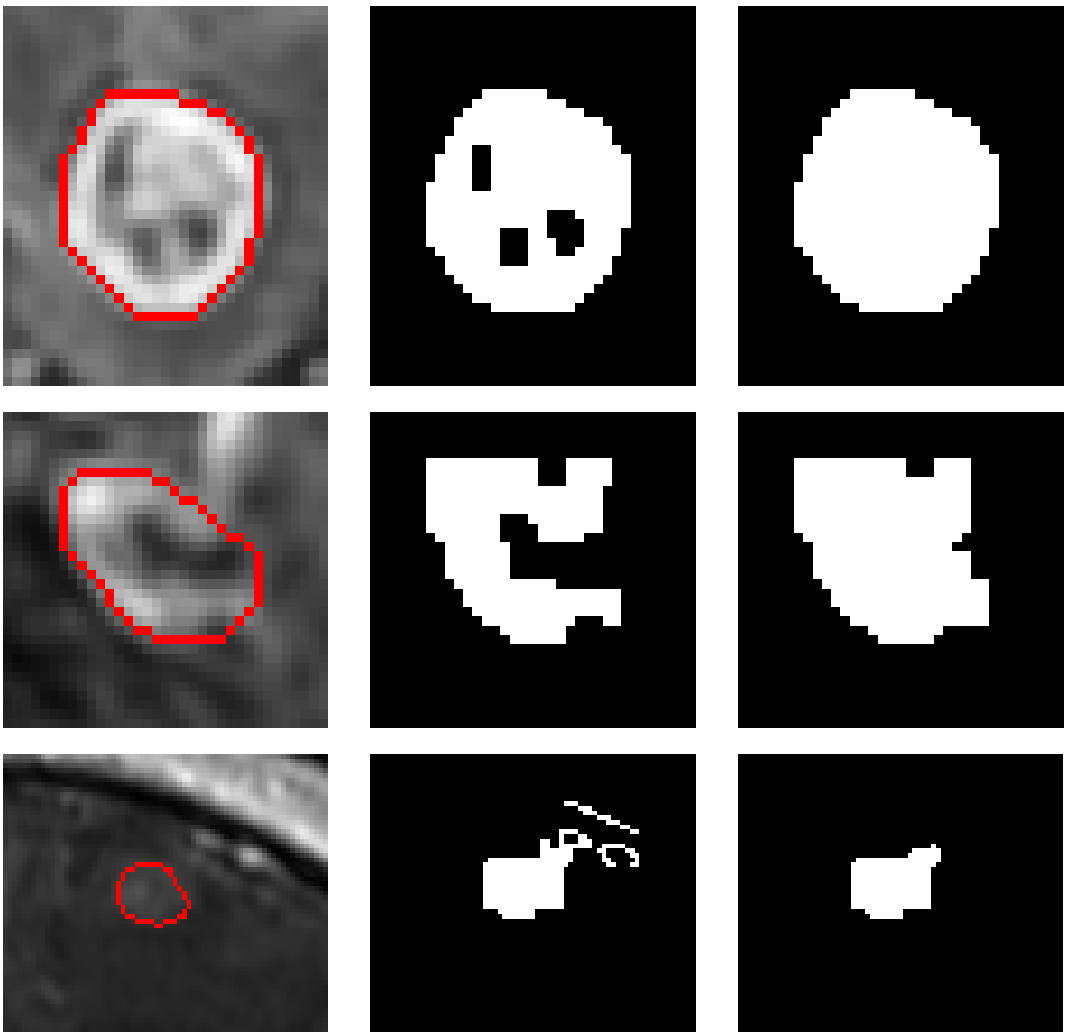


Figure 4.9: Left: a tumor slice. Middle column: (top) segmentation with holes inside. (middle) segmentation with unlikely shape. (bottom): segmentation with isolating regions. Right: segmentation with star shape constraint. Red is the ground truth boundary. White pixel stands for tumor and black stands for background

that does not violate the star shape constraints. The star shape constraint can also be a soft constraint by replacing ∞ to a finite value. The advantage of the star shape constraint is that it can be directly incorporated in the graph cuts segmentation and can be optimized in single cut. The benefit of incorporating it in our energy is a significant help to rule out disconnected foregrounds, foregrounds with holes, or of unlikely shapes that are far from convex. This is the motivation of including the star shape constraint in our algorithm. The following part of this section will talk about how to add the star shape constraint.

4.4.1 2D and 3D star shape constraints

The star shape prior needs a user input seed as the star center. Recall that we require the user to provide a bounding box that contains the tumor. We can make use of this information and set the center of the bounding box to be the 3D star shape seed. We discretize all lines passing through 3D star center and add star shape constraints to neighboring pixels on a line. However, 3D star shape does not necessarily lead to a 2D star shape in each slice. Consider Figure 4.10 as an extreme example. We can see from the figure that no pixel pair violates the 3D star shape constraint along the discretized lines, therefore the segmentation keeps the star shape property in 3D. However, the ring-like segmentation on first and last slice are obviously non-star shape wherever the seeds are.

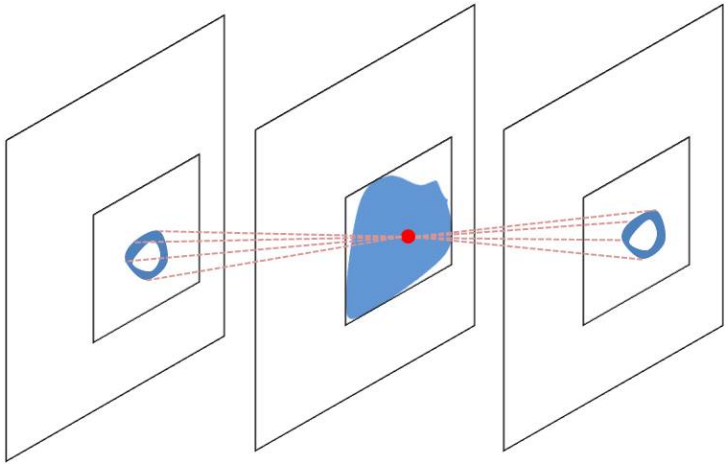


Figure 4.10: An extreme example of a star-shape in 3D that is not a star shape in its 2D projections. The red dot is the 3D star shape seed and dashed lines are discretized lines passing through star seed to pixels on first and last slice. 3-slices segmentation with ring-like shape in the first and last slice. This segmentation does not violate the 3D star shape constraint.

In our application, we found that most tumors are star shapes both in 3D segmentation and its 2D projections. Therefore, we need 2D star shape in each slice to prohibit not 2D-star shaped segmentations. In most cases, tumors are convex shapes so that we can choose any point inside

the tumor as the center for 2D star. To select 2D star shape centers, we make use of the user click on first and last slice. Recall that user provide us a bounding box in 3D with 4 clicks. We assume that the bounding box is tight and that the first and last slice clicks are in the center of the tumor in these slices. Then we project the line passing through the bounding box center to the user click in the first slice and take the intersection of this line with each slice as the center of the 2D star shape. The same process is repeated for the line connecting the 3D box center and the user click in the last slice. See Figure 4.11 for an illustration.

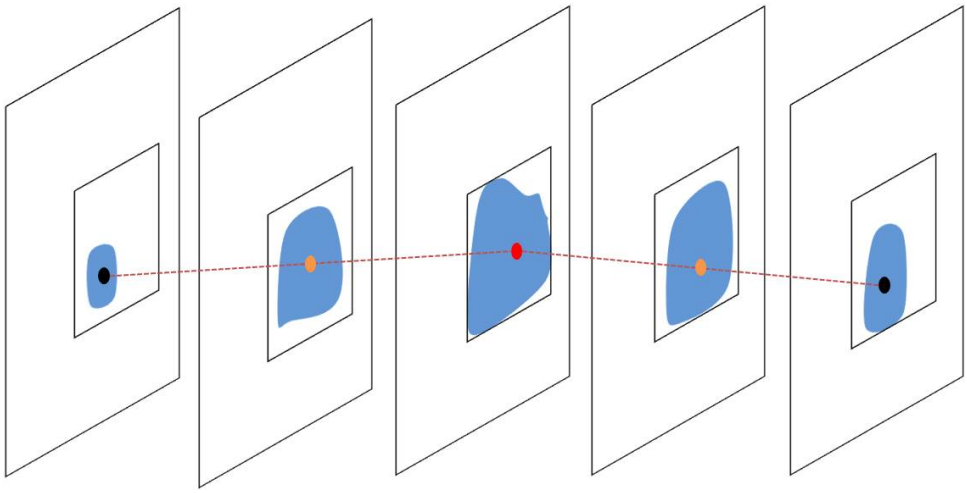


Figure 4.11: Star shape seeds for the whole volume. Red dot: 3D star shape center. Black dots: user clicks in first and last slices, and also the 2D star shape centers for first and last slice. Dark yellow dots: 2D star shape centers from the intersection of the lines with the 2D slices.

To add the star-shape constraint, we need to discretize the image pixels into lines passing through the center c. We start from a random pixel p in the image and discretize the line starting from center c to p. Different from the method used in [24], we use the well-known Bresenham line algorithm which is introduced in next section.

4.4.2 Bresenham line algorithm

The Bresenham line algorithm [7] is an algorithm which generates a set of points with integer coordinates to approximate a straight line between two given points. It is commonly used to draw lines on a computer screen as it uses very cheap operations in standard computer architectures. This section is a quick review of this algorithm. You can refer to [12] for more detailed explanation.

Let (x_0, y_0) and (x_1, y_1) be the endpoints of the line. To get a better approximation of the line, one has to always go through the dimension with larger distance because there are more sampling points in that dimension. The general equation of the line through the endpoints is

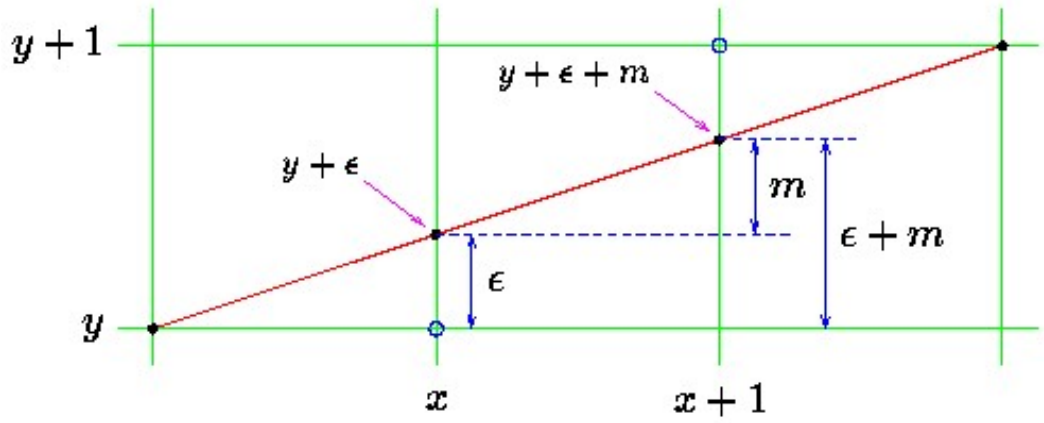


Figure 4.12: Example for plotting a line on screen. Red line: the line to be approximated; Black dot: real points on the line; Blue circle: plotted points; ϵ: error for y at x; m: slope of the line. (Figure from [3])

given by

$$\frac{y - y_0}{y_1 - y_0} = \frac{x - x_0}{x_1 - x_0} \tag{4.10}$$

When going through all possible x, we round the quantity given by Eq. 4.10 to the nearest integer

$$\begin{aligned} y &= \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) + y_0 \\ &= m(x - x_0) + y_0 \end{aligned} \tag{4.11}$$

where m is the slope of the line. Consider Fig. 4.12. For convenience, we always assume that the line goes down and to the right, which means $x_0 \leq x_1$ and $y_0 \leq y_1$. We also assume the line's horizontal projection $x_1 - x_0$ is longer than its vertical projection $y_1 - y_0$. Other cases can be easily transferred to this case. We have to compute the corresponding y for each $x \in x_0, x_0 + 1, ..., x_1$. The problem is equivalent to having plotted a point at (x, y), and choosing where to put the next point on the line. We have two options: (a) choose the point $(x + 1, y)$ or (b) choose the point $(x + 1, y + 1)$. Suppose that ϵ is the error of our choice at x, then $y + \epsilon$ is the real value at x. $y + \epsilon + m$ is the real value at $x + 1$. $\epsilon + m$ is the error for choosing y at point $x + 1$. We are seeking a point at y dimension that the error less than 0.5. Therefore, we choose

$$y = \begin{cases} y & \text{if } \epsilon + \text{m} < 0.5 \\ y + 1 & \text{otherwise} \end{cases} \tag{4.12}$$

The Bresenham algorithm repeats plotting the next points by updating ϵ to the new error at the

current point to

$$\epsilon = \begin{cases} \epsilon + m & \text{if } \epsilon + \text{m} < 0.5 \\ \epsilon + m - 1 & \text{otherwise} \end{cases} \tag{4.13}$$

Computers operate relatively slowly on fractional numbers like ϵ and m. Moreover, errors can accumulate over many floating-point additions. Bresenham algorithm fixes this problem by working with integers, which is faster and more accurate. To do this, we have to multiply all the fractional numbers (including the constant 0.5) by $(x_1 - x_0)$. Below is the integer Bresenham line algorithm we use in our application.

Algorithm 1 Bresenham line algorithm

```
function line(x0, x1, y0, y1)
  int deltax : = x1 − x0
  int deltay : = abs(y1 − y0)
  real error : =deltax/2
  int y := y0
   for x from x0 to x1
     error := error -deltay
     if error < 0 then
       y : = y + 1
       error := error +deltax
```

4.5 Volume ballooning

The edge and appearance cues usually get weaker as we go far from the tumor center. Even with the local appearance model, we are still not able to capture enough volume of the tumor on slices that are far from the center. Without the hard constraints, there will be empty segmentations on some slices.

Recall that our algorithm requires the user to provide us a tight bounding box. Therefore all of the slices should contain a non-zero amount of tumor. Moreover, the bounding box offers a cue for the tumor size. Therefore, we decided to include volume ballooning to encourage a larger foreground region. Figure 4.13 shows an example where we do not get a large enough foreground object at the end of the volume compared to the center of the volume.

We add a volumetric bias towards object to the data term by increasing the penalty of certain pixels to be assigned to the background (see Figure 4.15). We assume that pixels within a circle on a slice, where the circle is centered at the star shape seed, are more likely to be the object. Let $(1, 2, \ldots, n)$ index different slices of a volume, and $(R_1, R_2, \ldots, R_n)$ be the size of circle on different slices. The size of the circle on central slice R_{ct} is determined by the length of the

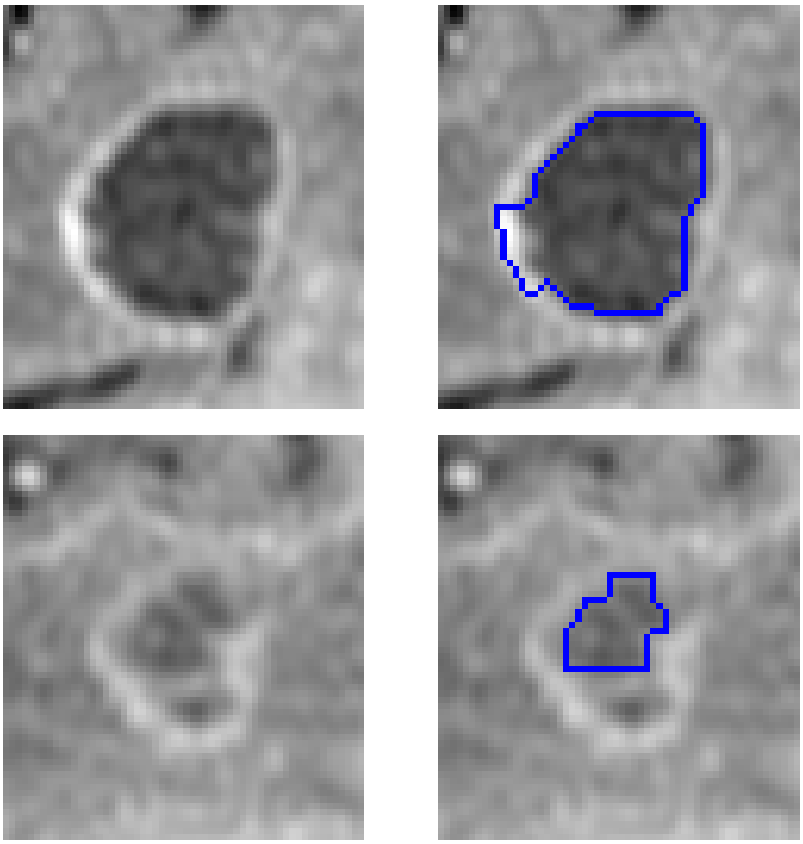


Figure 4.13: Left: two slices from the same volume, one from the central slice (top) and the other (bottom) is near the end of the volume. Right: the corresponding segmentation.

bounding box on that slice. We set it half of the smaller length of the bounding box. Radius of circles in the other slices decreases linearly from the center to the first and last slices. The circles in the first and last slices are of the same size, where the ratio between it and the size of central circle is a constant r, trained from the whole dataset.

$$\begin{aligned} R_1 &= R_n = rR_{ct} \\ R_i &= R_1 + \frac{|i - ct|}{ct - 1}(1 - r)R_{ct} \end{aligned} \tag{4.14}$$

where ct is the index of the central slice. The value of the volumetric bias is determined by a function of distance to the center. In our work, we use a function that decreases linearly with the distance to the circle center, and reaches zero, i.e. no bias at the circle circumference. That is

$$B_p = c(1 - \frac{d_p}{R}) \tag{4.15}$$

and

$$\frac{d_p}{R} \leq 1$$

where d_p is the distance from pixel p to the circle center, R is the circle radius of that slice, c is the bias value at the center. The function above is illustrated in Figure 4.14. We set the central bias the same for all the slices and train c and the circle ratio r on the whole dataset.

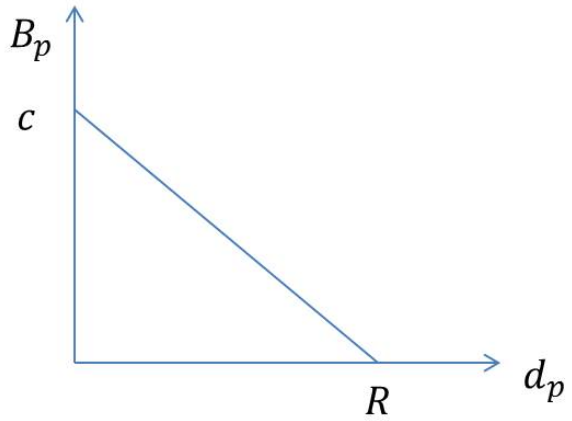


Figure 4.14: Linear function for volumetric bias. B_p is the bias value at pixel p. c is the bias value at the 2D star shape center. R is the radius of volumtric circle on the slice. d_p is the distance from pixel p to the center.

The volumetric bias can be easily implemented by adding additional cost on links $\{s, p\}$, as illustrated in the Figure 4.15. Figure 4.16 shows the comparison of segmentation with and

without volume ballooning.

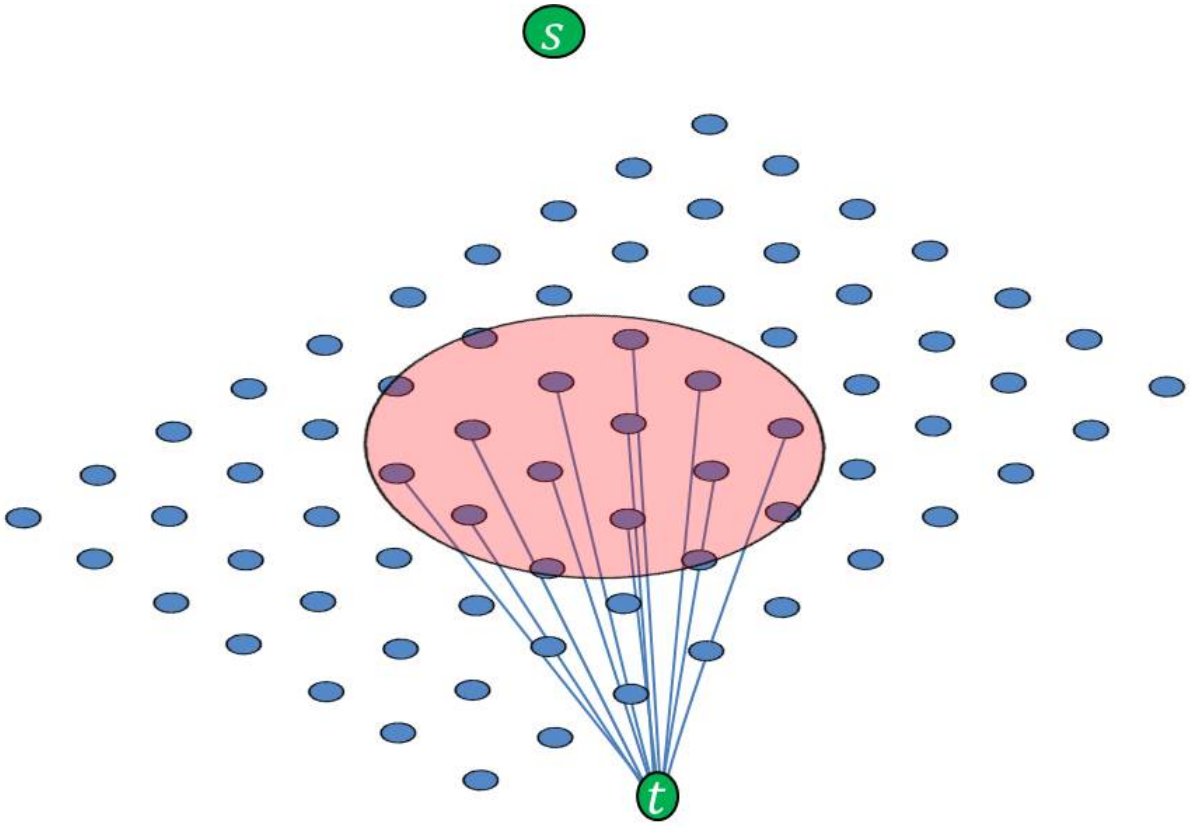
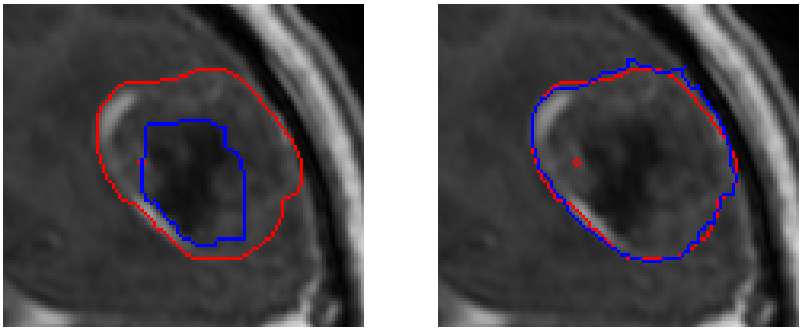


Figure 4.15: Graph for volume ballooning. Pixels inside the circle are biased to be the object.



(a) (b)

Figure 4.16: (a): segmentation without volumetric ballooning. (b): segmentation with volumetric ballooning. Volumetric ballooning gives a larger segmentation. Red contour is ground truth, blue contour is the segmentation.

4.6 Multi-region model

We also apply Delong's multi-region framework [11] in our application because we found many tumors contain ring-like boundaries. These are abnormal tissues we want to capture as well. Thus the tumor has two regions of distinct appearance: an interior tumor part (inside of the tumor), and an exterior tumor part (surrounding the inside part of the tumor). See Figure 4.17.

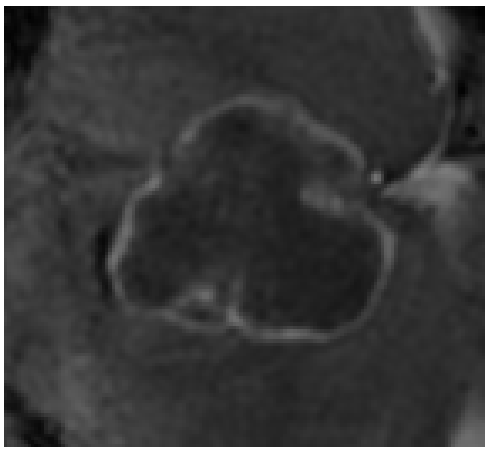


Figure 4.17: Tumor with distinct object interior (dark) and exterior (white).

The graph in multi-region framework is very different from the one in binary segmentation. It becomes a multi-labeling problem where we have to construct a multi-layer graph to encode the energy. In this section, we explain one special case in multi-region framework where only object exterior+object interior model is considered. We begin with the same input data as we do in binary segmentation. To perform multi-region segmentation, we have to build the appearance model for three regions, object interior, object exterior and the background. However, the initial labeling only separates the volume into two regions, one for tumor and the other for background. We fix the problem in a two-step approach: first, we take appearance of object from a smaller 3D box than the one provided by the user and perform the first segmentation. Second, we erode/dilate the segmentation to get a mask with 3 labels. We choose to erode or dilate depending on the divergence of the two regions after erosion and dilation. Then we build the appearance model for background/object interior/object exterior and perform the graph cuts algorithm again to get the final result. Figure 4.18 illustrates our boundary-object segmentation.

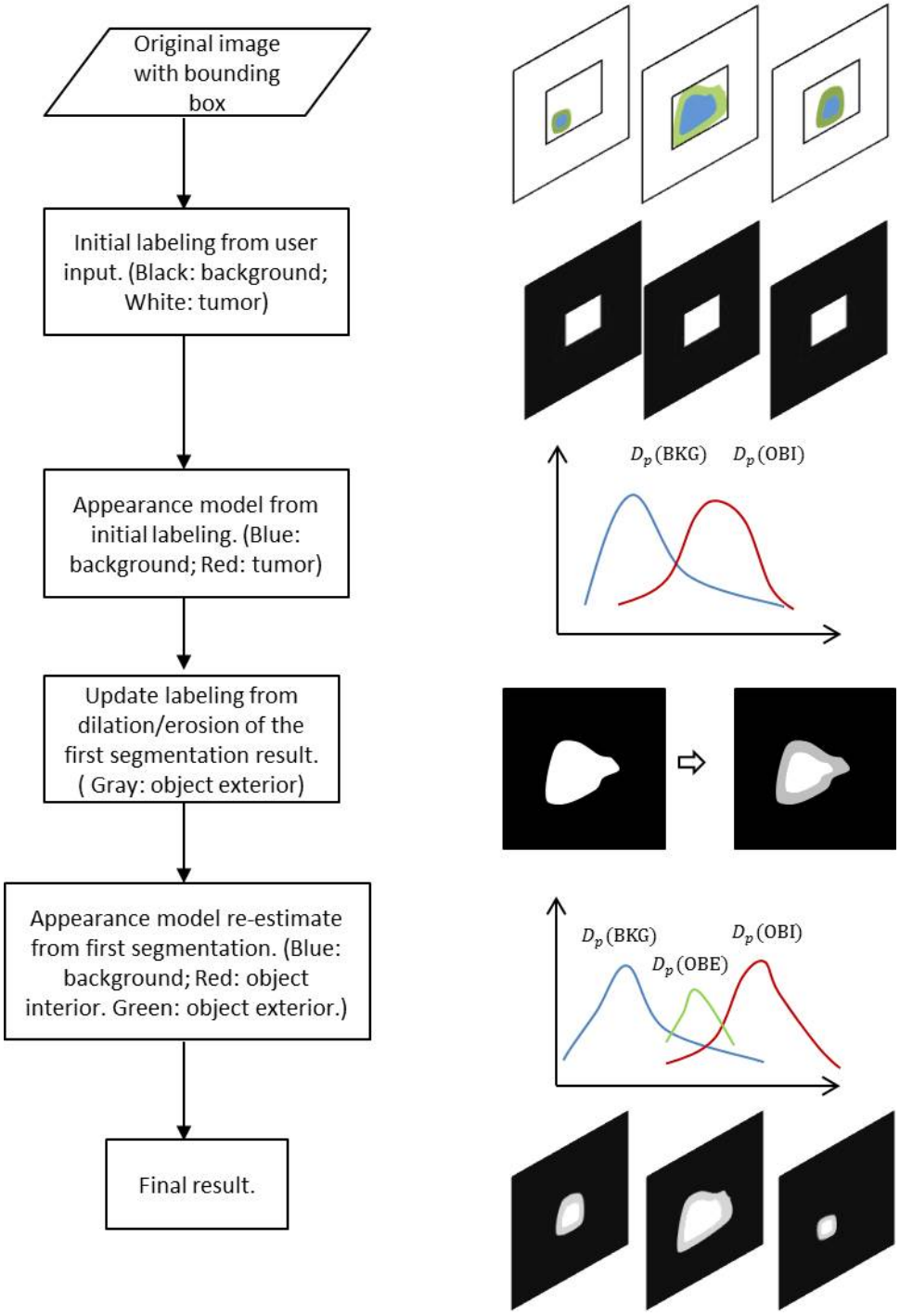


Figure 4.18: Flow chart for our boundary+object interior segmentation.

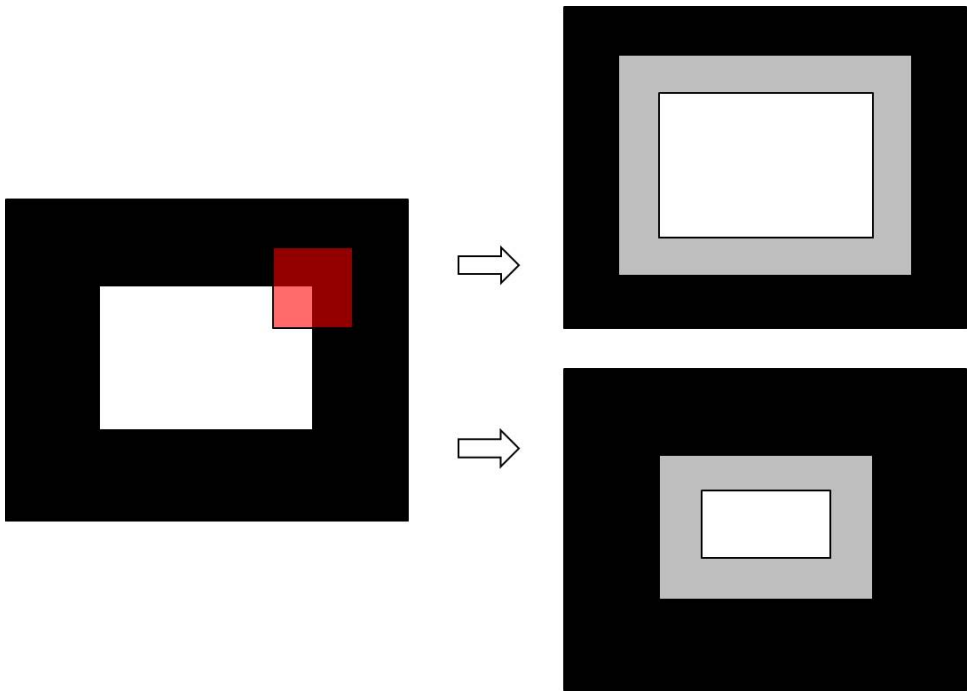


Figure 4.19: Left: segmentation result (white: object; black: background), red rectangle is the mask for dilation/erosion. Right: result of dilation (top) and erosion (bottom). Gray area is assumed to be the boundary.

4.6.1 Multi-region energy

We have 3 labels with each represents a region in the image, our label set becomes $\mathcal{L} = \{\text{BKG}, \text{OBE}, \text{OBI}\}$, with each value represents background, object exterior and object interior respectively. For simplicity, we do not include the star shape constraint and volume ballooning in our energy.

$$E = \sum D_p(f_p) + \sum_{\{p,q\} \in \mathcal{N}} V_{pq}(f_p, f_q) \tag{4.16}$$

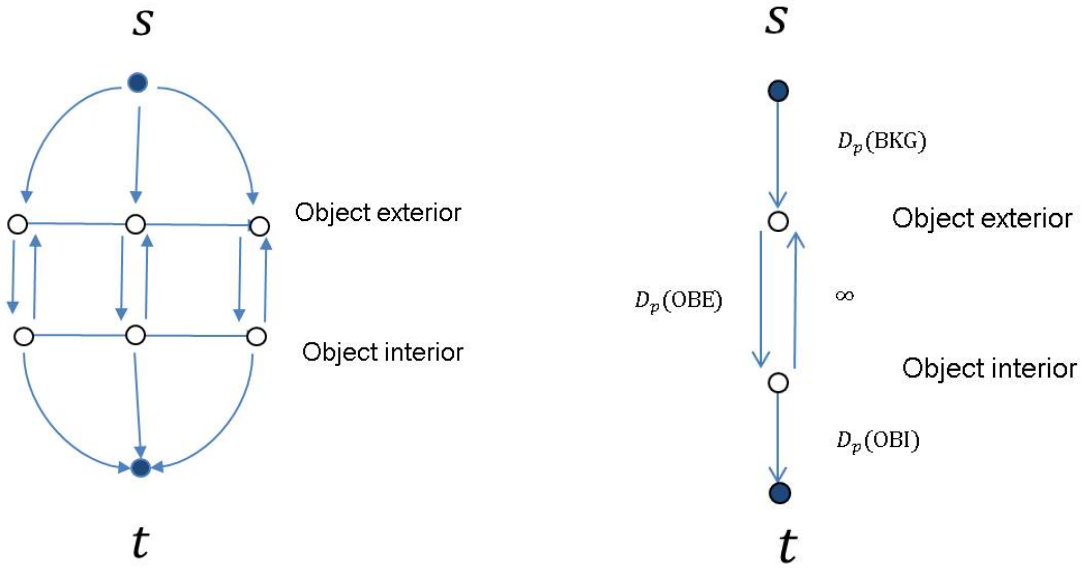
and

$$V_{pq}(f_p, f_q) = w_{pq} \cdot |f_p - f_q| \tag{4.17}$$

where we associate label BKG, OBE, OBI with real values 0, 1, 2 respectively. We assume that the background and object interior have more distinct difference than background and object boundary or object boundary and object interior. w_{pq} is the same as the one described in Section 4.3. We use negative log-likelihood as our data term.

4.6.2 Multi-region graph construction

In our application, we model a tumor as a two-region object, one for object exterior and the other for object interior. Therefore we do not need exclusion interaction. We only employ the containmentation interaction which means that the object exterior contains the object interior. We set the inter-region neighborhood $\mathcal{N}^{ij}$ to be the set of all pixels pairs (p, p) between two regions. This allows us segment a tumor without distinct exterior region and thus gives us more flexibility. Recall that multi-region segmentation is a multi-labeling problem, we have to construct a graph with multiple layers. Each layer represents a region of object. Consider Figure 4.20, the vertical links encode data term and the horizontal links encode smoothness term.



(a) Graph for multi-region segmentation.

(b) Graph weights for different links.

Figure 4.20: An example for graph construction of multi-region segmentation.

Figure 4.21 shows an example of multi-region segmentation on a 1×3 image. The inter-layer link of pixel z is cut so z is labeled OBE. We pay $D_p(OBE)$ for this label assignment. Besides the data cost, we pay smoothness cost for assigning different labels to neighboring pixels, therefore, the optimal energy for the graph in Figure 4.21 is

$$E^* = D_p(\text{BKG}) + D_p(\text{OBI}) + D_p(\text{OBE}) + 2w_{pq} + w_{qz}$$

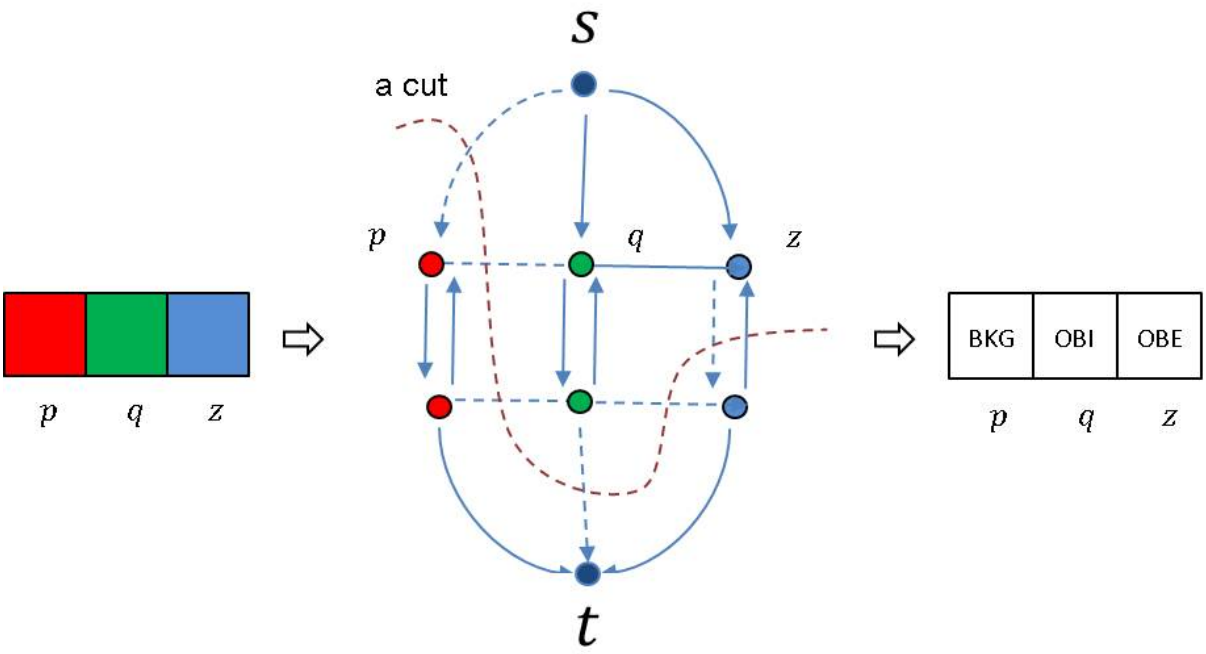


Figure 4.21: Left: 1×3 image. Middle: multi-region graph and the minimum cut for the graph. Right: the result labeling for the input image.

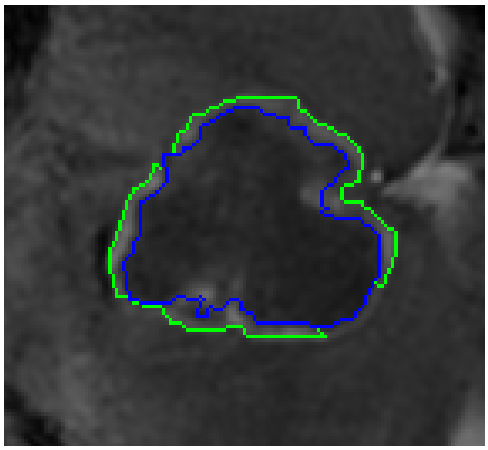


Figure 4.22: Multi-region segmentation for tumor shown in Figure 4.17. Pixels within the blue contour are object interior. Pixels in between blue contour and green contour are the object exterior. Pixels outside the green contour are the background.

Chapter 5

Experimental results

In this section we discuss our experimental setup, parameter selection, and the segmentation results, including the experiments on the ground truth provided by Aaron Ward and Glenn Bauman.

5.1 Implementation details

We try many options of our algorithm and make the best choice from all possibilities.

5.1.1 Simulation of user interaction

Since we have only the ground truth provided by the medical experts, we have to simulate user interaction, that is the two clicks specifying the 2D bounding box in one slice, and 2 clicks in the tumor center in the first and last slices. Our 2D box is obtained by extracting 4 edge points on x and y dimensions. The 4 edge points are enough to specify the 2D bounding box which completely contains the tumor in every slice. We randomly pick 2 voxels near the center of the tumor in first and last slice as 2 user clicks on these slices. Our 3D bounding box is very tight and the real user input might not be so precise, so we perform the sensitivity experiment of the bounding box in **Section** 5.2.4.

5.1.2 Parameter selection

There are four parameters that need to be provided before we run our algorithm. They are λ_{xy}, λ_z in Equation 4.8 for controlling the relative importance of the smoothness term, and r, c in Equation 4.14 and Equation 4.15 for determining the volumetric bias. In our work, we train these parameters from the whole dataset using cross-validation. Training all four parameters

together is too time consuming. Therefore we decided to train the smoothness parameters and the volume ballooning parameters separately. That is we first find the best smoothness parameters with the volumetric ballooning parameters set to zero (that is no ballooning), and then train for the ballooning parameters keeping the smoothness parameters fixed.
Our main statistics for evaluation is the F-measure, see Section 5.2.2. So our cross validation is based on F-measure. We train λ_{xy} and λ_z from 0.1 to 20 with 15 candidates for each parameter. To prepare cross validation experiments for λ_{xy} and λ_z, we compute and store the F-measure for each $(\lambda_{xy}, \lambda_z)$ for each volume. So there is an F-measure matrix for each volume.
Let the number of volumes to be k, v_{ij} be the number of votes of ith λ_{xy} and jth λ_z.

Algorithm 2 Cross Validation

1: **for** volume $v = 1, \ldots, k$ **do**
2: Take out all the f-measures corresponding to volume v.
3: Average the other f-measure matrixes and find the best $(\lambda_{xy}, \lambda_z)$, increase its vote v_{ij} by 1.
4: **end for**
5: Take the $(\lambda_{xy}, \lambda_z)$ with the most votes as our input.

The training for volumetric parameters is the same.

5.2 Experimental results

5.2.1 Image data

Our data is provided by Aaron Ward and Glenn Bauman from Robarts research, with 88 tumors from 64 scans of 27 patients. Some tumors are from different stages of the same tumor. The voxels are represented by 16-bit integers. Each volume has a ground truth provided by the expert. Ground truth is somewhat imprecise for some examples since these segmentations were obtained by blob-manipulation tools.

5.2.2 Evaluation methods

A commonly used evaluation method in image segmentation is the error percentage, which can be specified by

$$error = \frac{\text{FP} + \text{FN}}{\text{number of pixels to be labeled}}$$

where FP is the number of false positives, which means background pixels wrongly labeled foreground, and FN is the number of false negative, which means foreground pixels wrongly

labeled background. The sum of FP and FN is the number of pixels labeled incorrectly. The ratio between it and the number of pixels to be labeled is the error rate that measures the performance of the segmentation algorithm. The smaller the error, the better the segmentation. However, this measurement is highly sensitive to the bounding box size. Consider a larger and a smaller bounding box around a tumor. Suppose a segmentation algorithm labels exactly the same pixels in both boxes as the tumor pixels. We would like to say that these segmentations have the same accuracy. However, with the error metric described above, the larger box will have a smaller error, since the larger box has more pixels, and FN and FP are the same between the smaller and larger boxes. The problem here is that the error measure takes into consideration TN (true negative), that is the number of background pixels that are correctly labeled as background. TN are easier to classify, especially on the outskirt of the box, and by making the box larger, we take more of this pixels into consideration. Thus we need to use another measure for a more realistic evaluation.

A more appropriate evaluation method is F-measure, which is not based on TN. F-measure is based on precision and recall of the segmentation, where precision is the fraction of retrieved pixels that are tumor, and recall is the fraction of tumor pixels that are retrieved. The F_β score is a weighted average of the precision and recall with a positive real β, the general formula is

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

and

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

where β controls the relative importance between precision and recall. The advantage of F-measure over error percentage is that only the ground truth and the segmentation result will affect the score and it eliminates the influence of box size. Therefore, we use F-measure instead of the error percentage.

Parameter β is used to weigh the relative importance of recall and precision. For our evaluation, we treat precision and recall equally important, so we set $\beta = 1$. The formula of F-measure becomes

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{5.1}$$

The F1-measure we use is the same as dice score, which is commonly used in medical image processing.

5.2.3 Evaluation of the results

We tried different combinations of the constraints discussed in **Chapter 4**. By default, we use parzen window to compute the normalized global or local intensity distribution. Table 5.1 is the performance of different methods. Notice that the average F1-measure of 88 volumes is not necessarily equal to the value obtained by applying the average of recall and the average of precision to Equation 5.1. Table 5.1 shows that local histogram and star shape constraint always help improving the result compared to global histogram. Before adding the volumetric bias, multi-region segmentation framework yields the best result. Volume ballooning improves the results for all the approaches and all the approaches seems to perform almost equally well with volumetric bias. We obtain the best overall results with 0.8712 for multi-region segmentation with star shape constraint and volumetric ballooning.

Table 5.1: Performance of different methods. 'B', 'G', 'L', 'S', 'V' stands for binary segmentation, global histogram, local histogram, star shape constraint and volume ballooning respectively. Each item of the first column is a combination of different options.

method	recall	precision	FM
BG	0.0.7373	0.8963	0.7936
BL	0.7540	0.8984	0.8039
BGS	0.7768	0.9287	0.8284
BLS	0.7923	0.9202	0.8365
MGS	0.7872	0.9286	0.8360
BGSV	0.8718	0.8819	0.8698
BLSV	0.8736	0.8819	0.8709
MGSV	0.8775	0.8789	0.8712

5.2.4 Sensitivity of user interaction

We currently simulate user interaction by extracting a tight 2D bounding box from the ground truth. Clearly, a real user would not be as accurate in the placement of the bounding box. Because we ask the user to give a 2D bounding box that is large enough to completely contain the tumor in all slices, errors result from too small boxes are viewed as user error, therefore smaller boxes are not taken into consideration in sensitivity experiment. Specifically, we test the sensitivity of the algorithm to the user box placement as follows. We enlarge the length of the current 2D box by a ratio from 0.05 to 0.2. Suppose w is the width and h is the height of current box, and we want to enlarge it by adding rw to the width and rh to the height. In Figure 5.1, we enlarge the box at left, right, top and bottom by $dw1, dw2, dh1, dh2$ respectively. And $dw1 + dw2 = rw$, $dh1 + dh2 = rh$. In order to move the box center at the same time, we

use the following computation

$$dw1 = \text{rand}(0, rw)$$
$$dw2 = rw - dw1$$
$$dh1 = \text{rand}(0, rh)$$
$$dh2 = rh - dh1$$

where rand is a function for randomly picking up integers in an interval. By doing this, we can enlarge the box by a certain value and randomly move the box center at the same time. We test different box sizes on the approach that yields the best result, namely the multi-region framework with star shape constraint and volume ballooning.

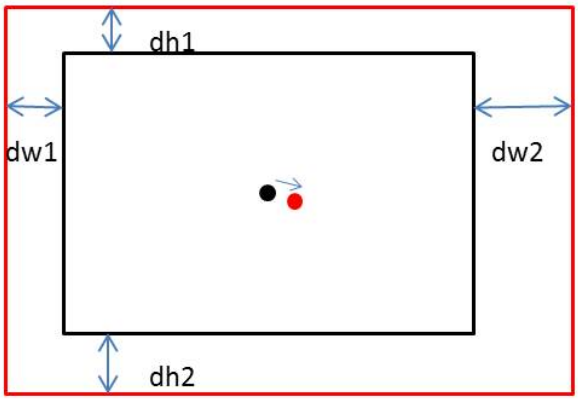


Figure 5.1: New 2D box (red) obtained by adding $dw1, dw2, dh1, dh2$ to the left, right, top and bottom of old 2D box (black) respectively. Black dot is the old box center and red dot is the new box center.

In general, the bounding box provided by the expert would not be as tight as the one we are using but it would not be too imprecise either. Therefore, we assume that a box whose length is 0.2 larger is the most imprecise box. We test boxes whose length is 1.05, 1.1, 1.15 and 1.2 of the box we obtained from the ground truth. That is, 4 'off' boxes for each volume. Figure 5.2 shows how the performance gradually changes as the box size gets larger compared to the tight bounding box obtained from the ground truth. Recall goes up because the circle of our volumetric ballooning gets larger and more pixels are biased to be the tumor. As a result, more pixels are labeled as tumor. Precision goes down sharply because we are labeling more non-tumor pixels to tumor. With a reasonable size of bounding box, our algorithm still achieve around 80% F-measure.

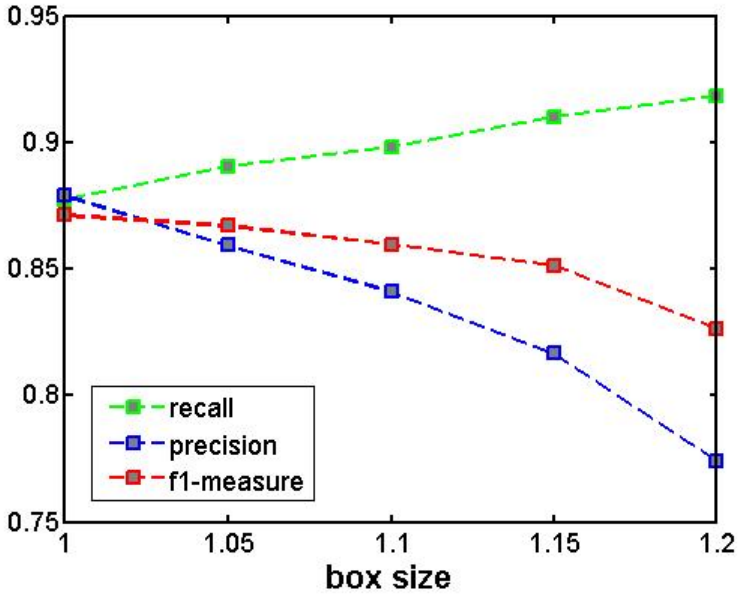


Figure 5.2: Performance for different box size. Horizontal axis is the ratio of the experimented box size to current box size.

5.2.5 Results

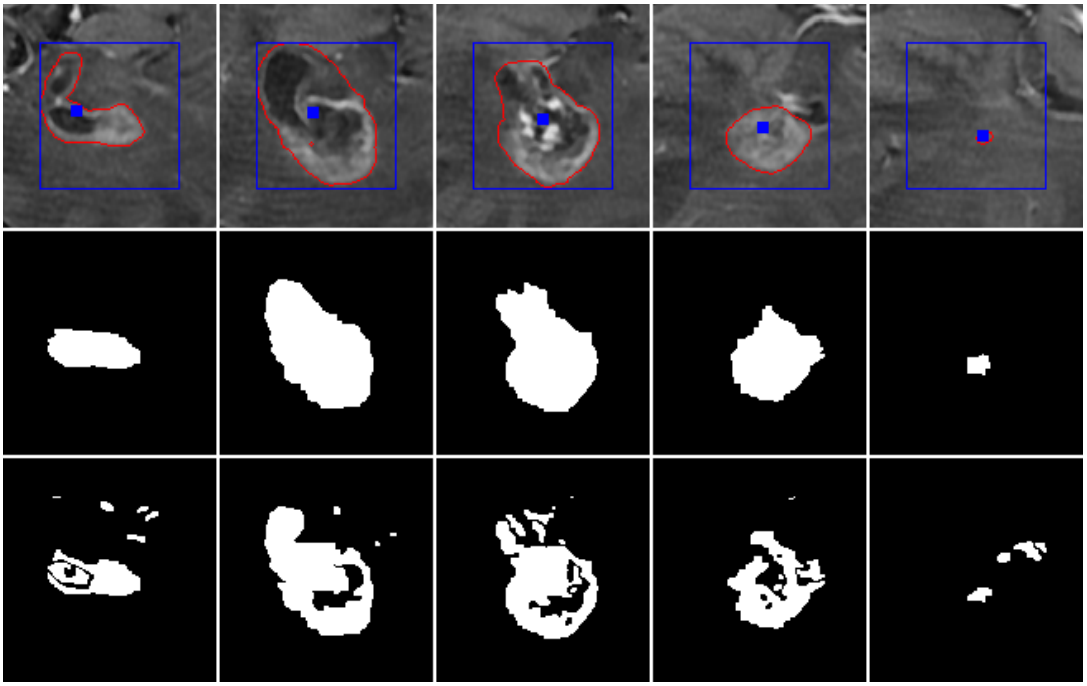
In this section, we display some typical results for different methods we have tried and compare them to each other. We first show the benefit of star shape constraint, then we incorporate it into all the subsequent experiments. We also show some examples where volumetric helps and examples where volumetric ballooning makes things worse. In addition, some failure examples are shown as well. Because there are too many slices for a volume, we can not display all of them. Therefore, we sample 5 slices from each volume.

BGS versus BG

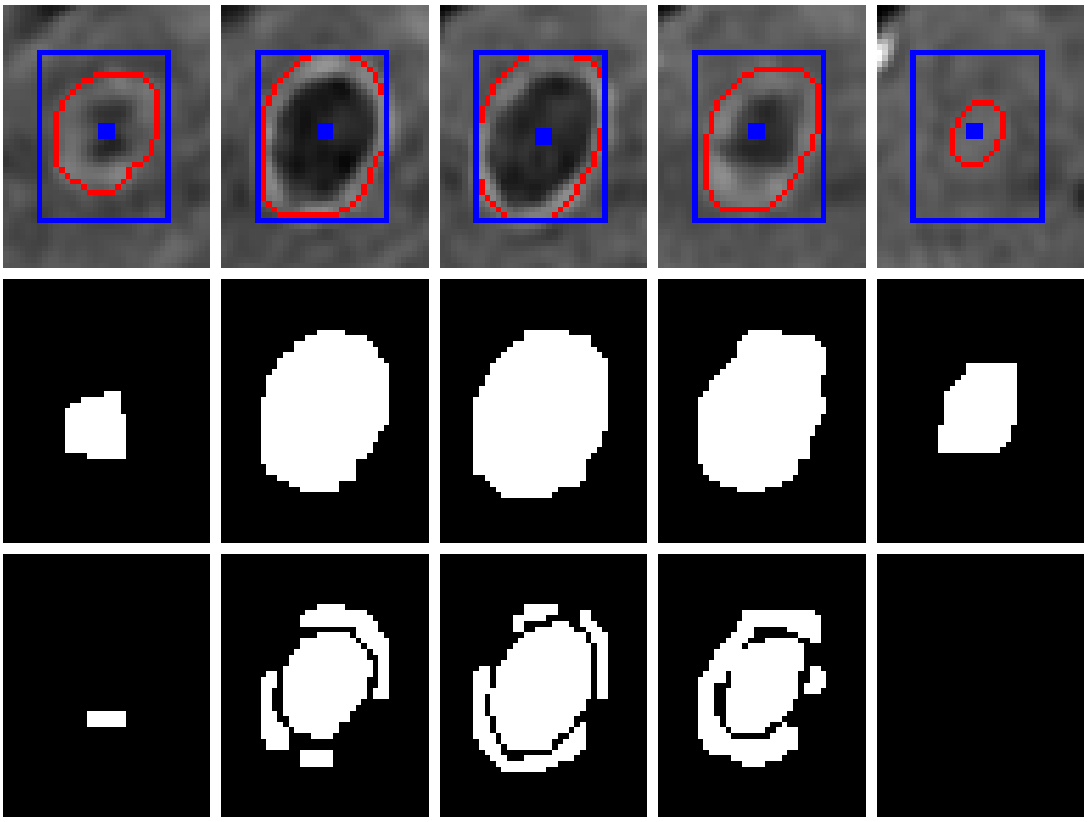
We show some results in binary segmentation with star shape and without star shape. Segmentation without star shape consists of many separate regions and star shape constraint helps prohibit this kind of segmentation, see Figure 5.3.

BGS versus BLS

From Table 5.1 we know that the overall performance of local histogram is better than that of global histogram. However, this is not always the case, there are still examples where global histogram would be a better choice. Figure 5.4 shows results using binary segmentation with local histogram, where each slice has its own background and foreground appearance model, are better than global histogram. Figure 5.5 shows an example where global histogram is better than the local histogram.

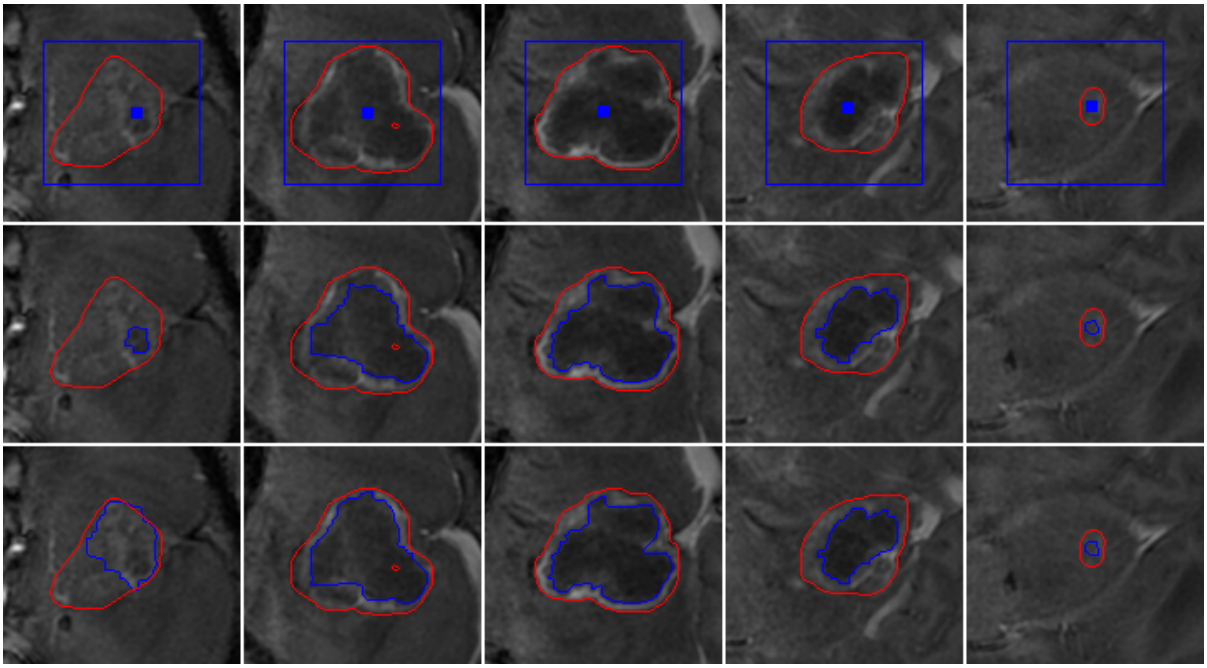


(a)

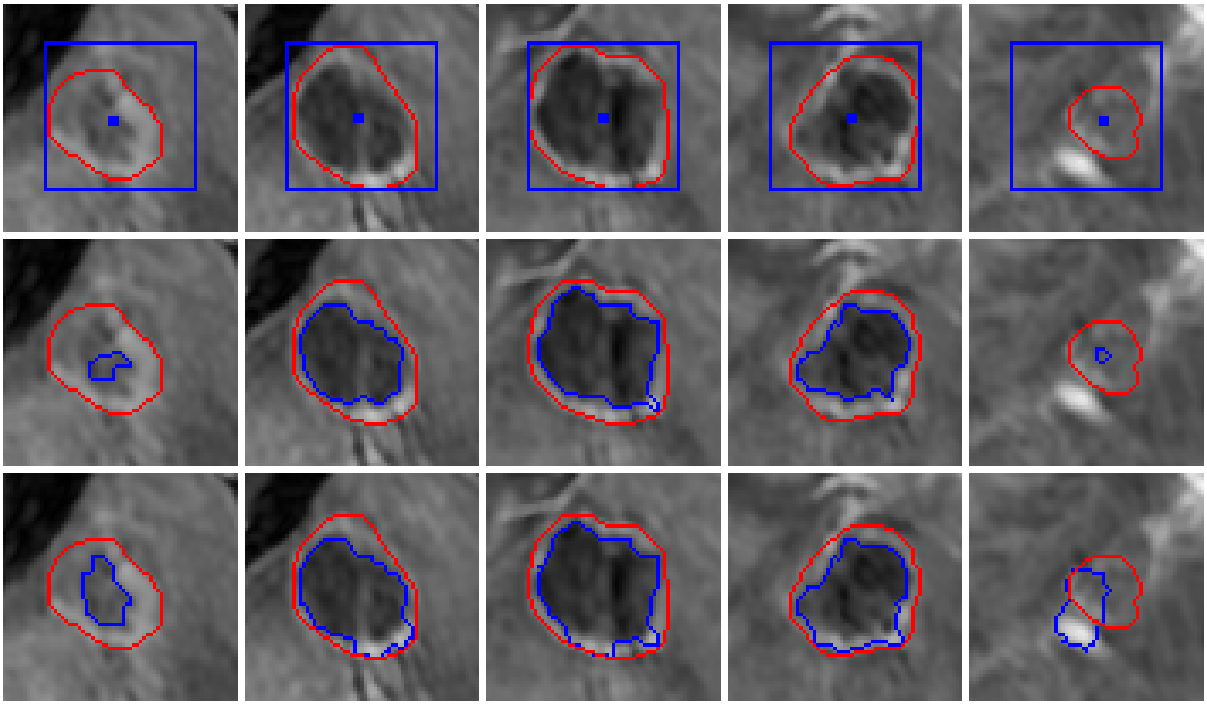


(b)

Figure 5.3: Comparison of results with and without star shape. Five volume slices are shown. For each example, the first row is the ground truth with bounding box and star shape seeds. The second row is result with star shape constraint and the third row is the result without star shape constraint.



(a)



(b)

Figure 5.4: Examples where local histogram works better than global histogram. Five volume slices are shown. For each example, the first row is the ground truth with bounding box and star shape seeds. The second row is result using global histogram and the third row is the result using local histogram.

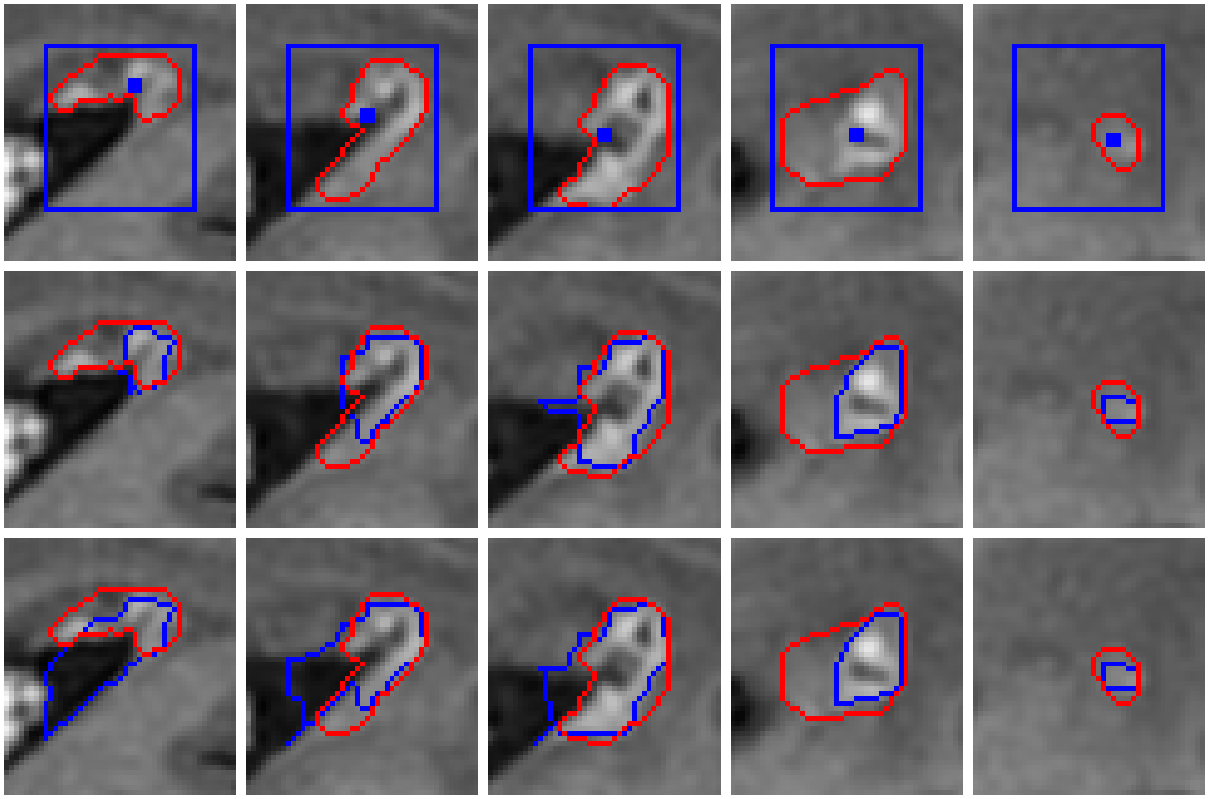


Figure 5.5: Examples where global histogram works better than local histogram. Five volume slices are shown. For each example, the first row is the ground truth with bounding box and star shape seeds. The second row is result using global histogram and the third row is the result using local histogram.

BLS versus MGS

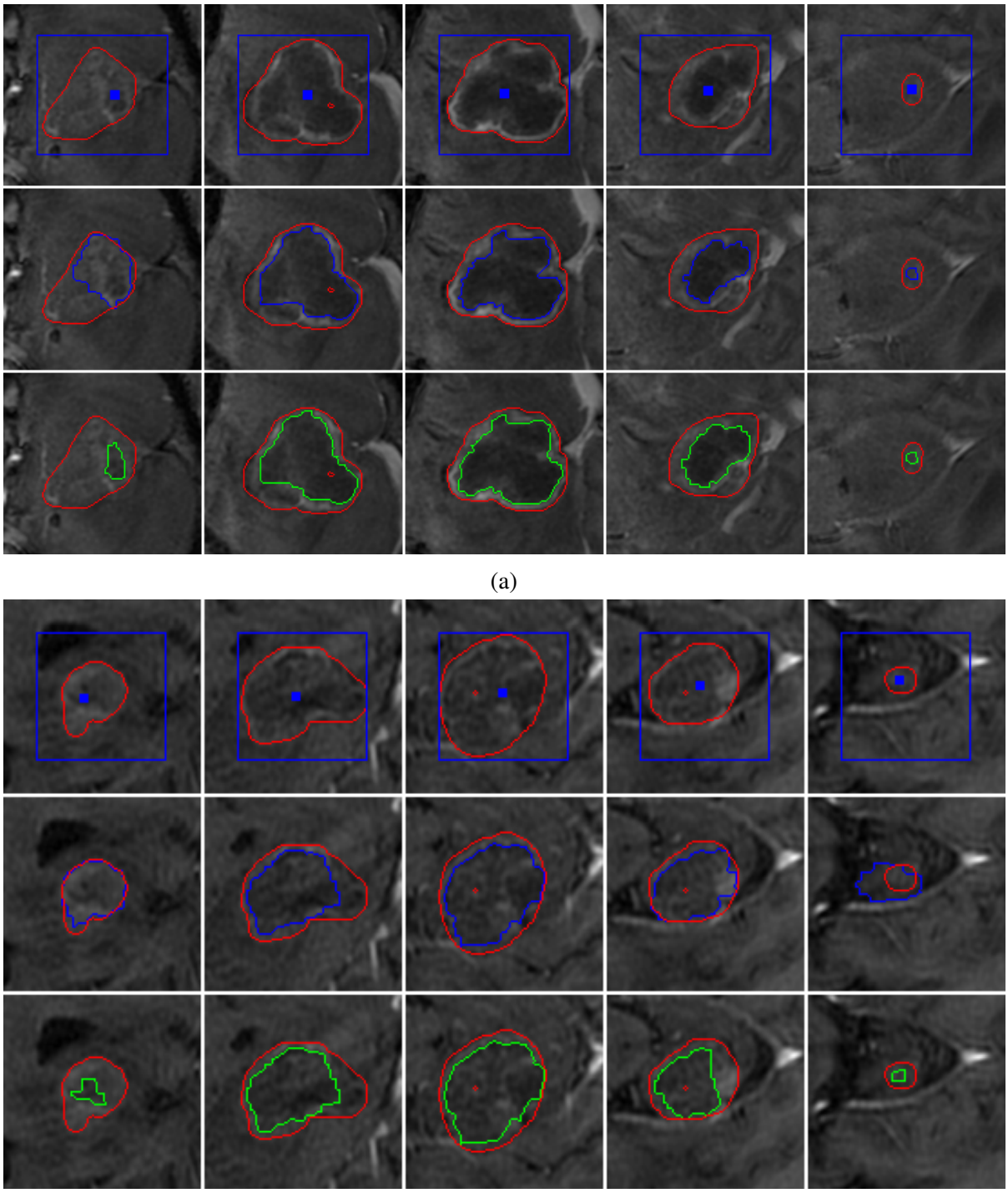
We compare the results obtained from multi-region segmentation framework to binary segmentation with local histogram. Multi-region segmentation framework allows modeling the tumor region as consisting of two parts, one part contained inside the other. See Figure 5.6 and 5.7.

MGS versus MGSV

The volumetric ballooning helps especially when some slices are undersegmented. See Figure 5.8. Although volumetric ballooning improve most segmentation results, it worsens the performance for some examples, especially when the results are already very good without ballooning. See Figure 5.9.

Failure cases

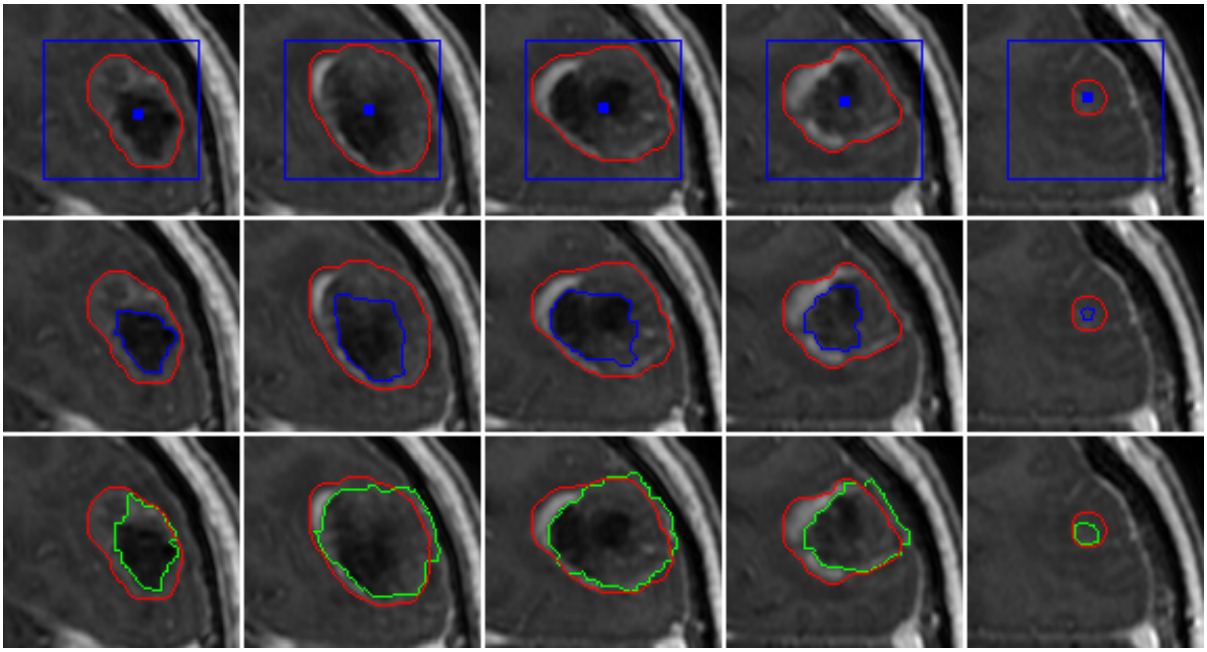
Even with so many options for solving the problem, there are still some volumes that are hard to segment. We show some of these examples in Figure 5.10 and Figure 5.11. The challenge of the first example in Figure 5.11 is the weak boundary and large overlap in appearance of foreground and background. Poor segmentations in other examples are mostly due to the weak edges between the tumor and healthy tissue.



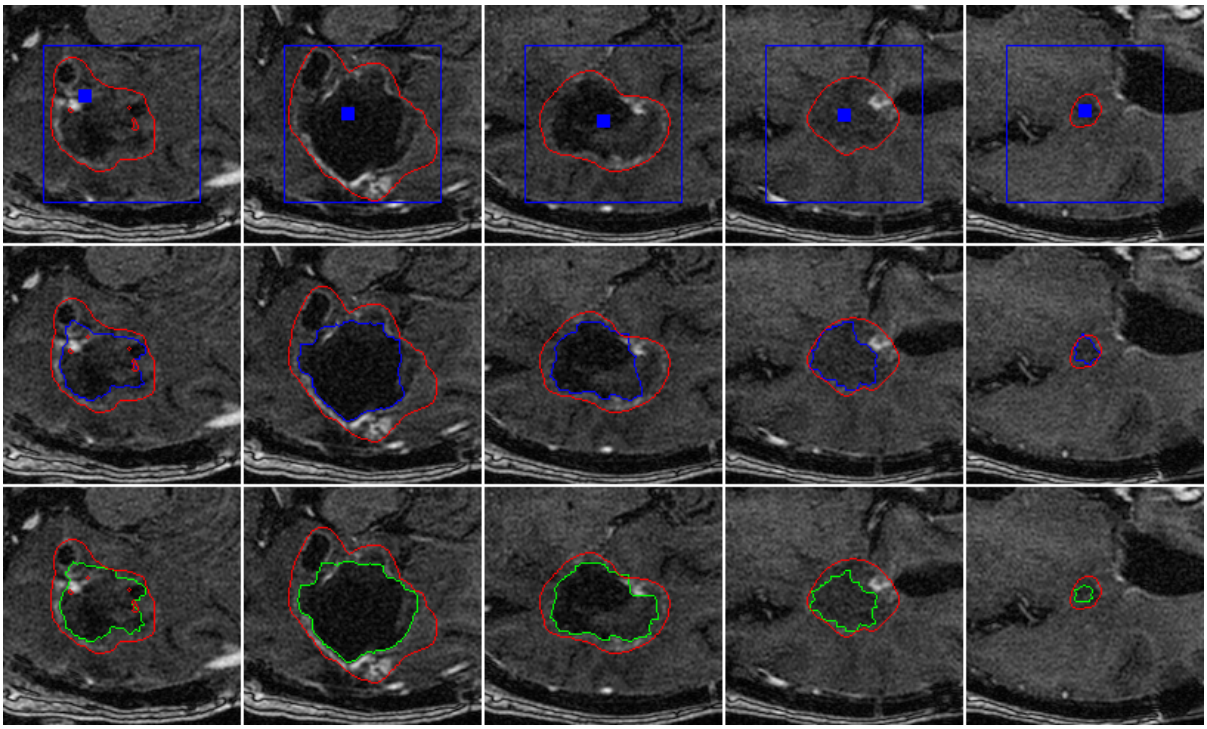
(a)

(b)

Figure 5.6: Examples where binary segmentation with local histogram works better than multi-region segmentation. Five volume slices are shown. For each example, the first row is the ground truth with bounding box and star shape seeds. The second row is result from binary segmentation with local histogram, where blue contour outlines the tumor boundary. The third row is the result from multi-region segmentation, where green contour outlines the boundary between tumor exterior and the background, blue contour outlines the boundary between the tumor exterior and the tumor interior. One contour means the tumor contains only one region.

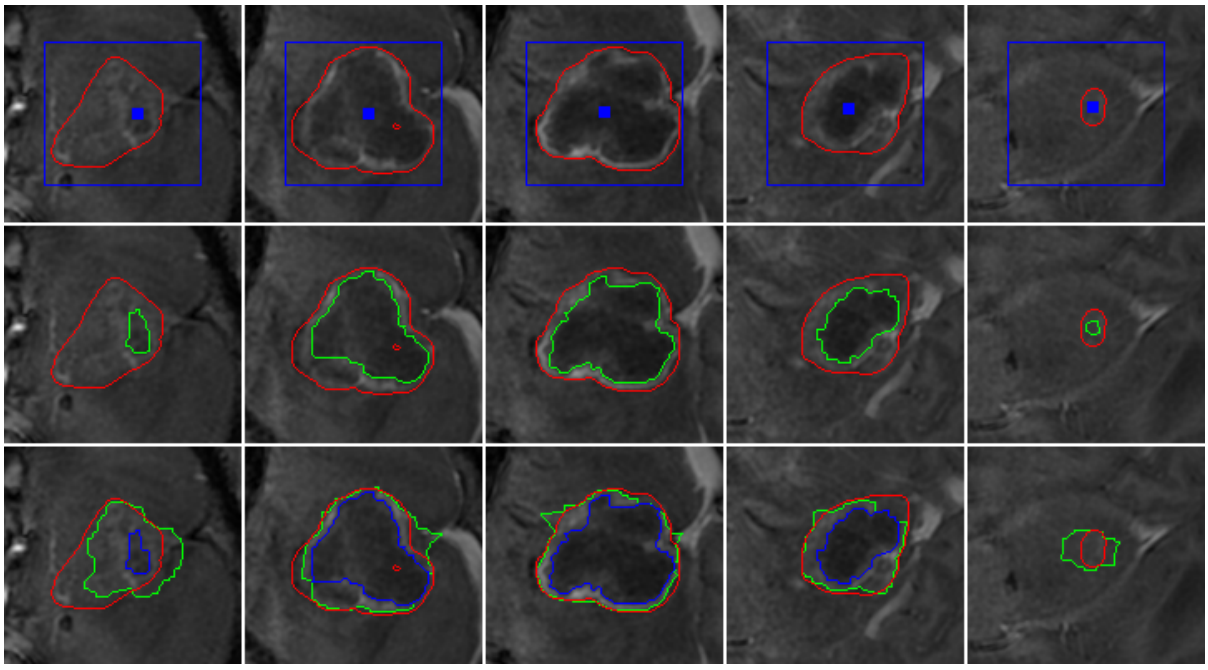


(a)

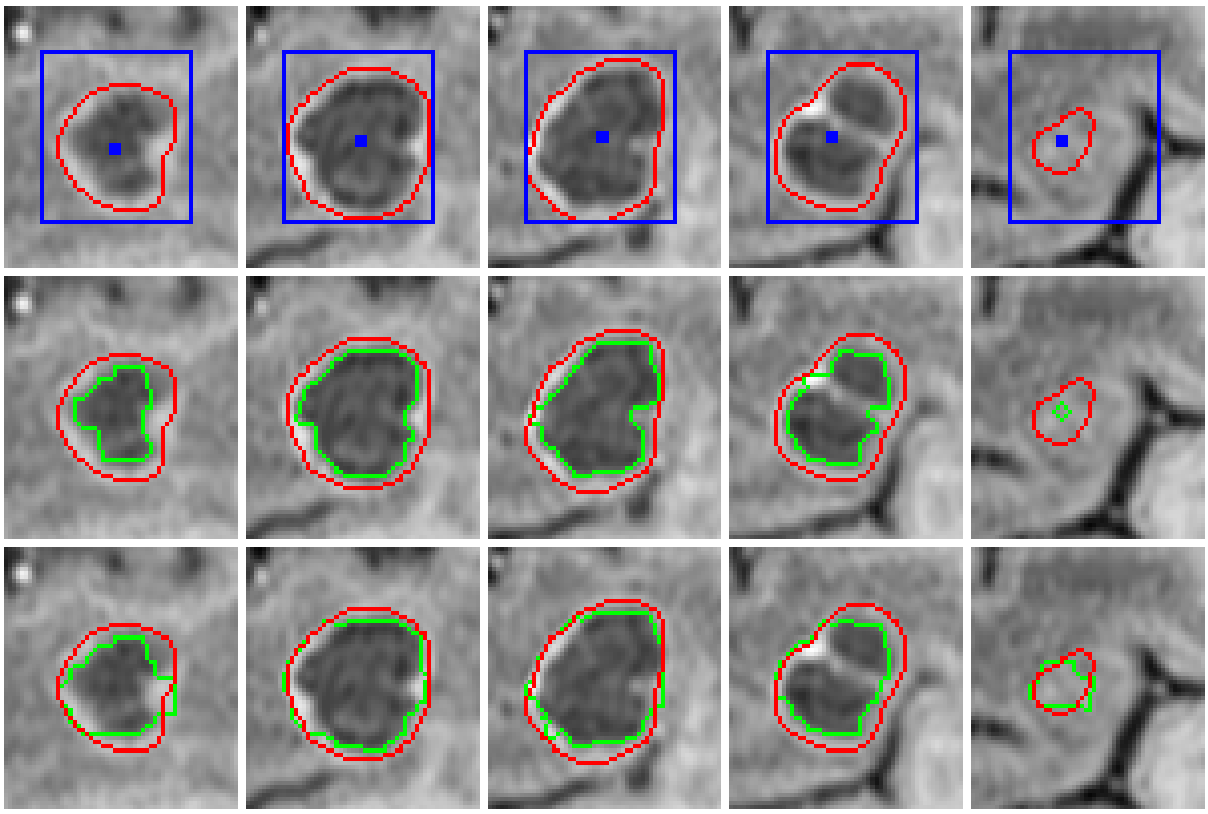


(b)

Figure 5.7: Examples where multi-region segmentation works better than binary segmentation with local histogram. Five volume slices are shown. For each example, the first row is the ground truth with bounding box and star shape seeds. The second row is result from binary segmentation with local histogram, where blue contour outlines the tumor boundary. The third row is the result from multi-region segmentation, where green contour outlines the boundary between tumor exterior and the background, blue contour outlines the boundary between the tumor exterior and the tumor interior. One contour means the tumor contains only one region.

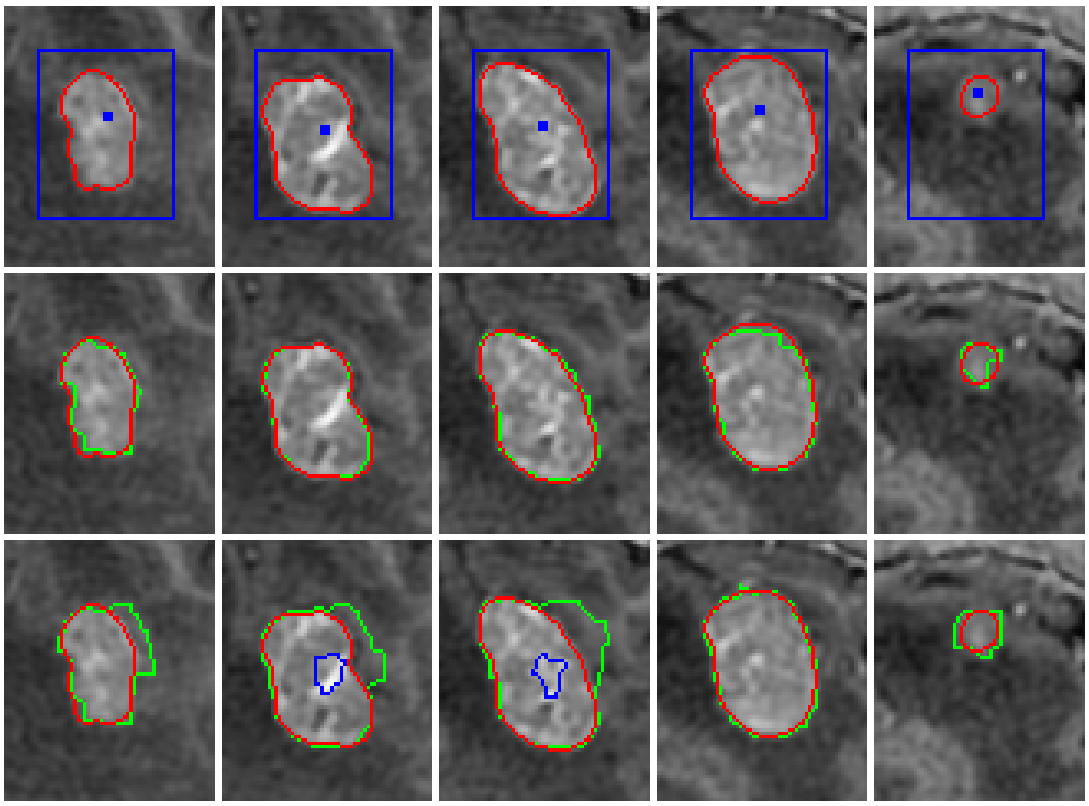


(a)

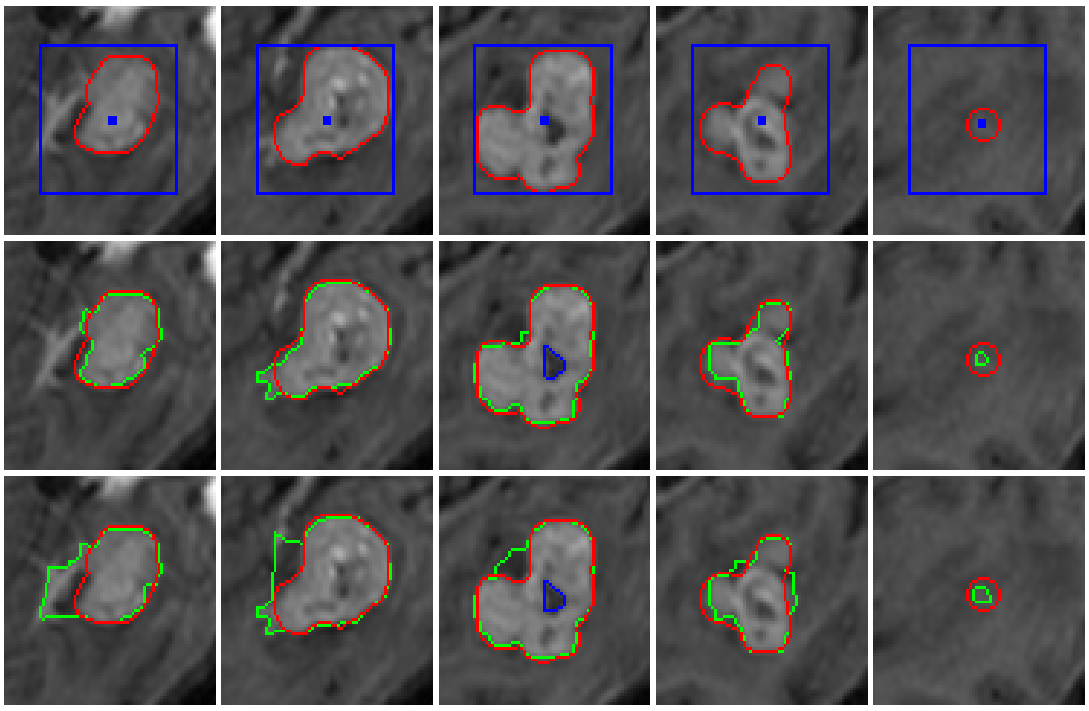


(b)

Figure 5.8: Examples where volume ballooning helps to get better results on multi-region segmentation framework. Five volume slices are shown. For each example, the first row is the ground truth with bounding box and star shape seeds. The second row is result multi-region segmentation without volume ballooning and the third row is the result from multi-region segmentation with volume ballooning. Green contour outlines the boundary between tumor exterior and the background, blue contour outlines the boundary between the tumor exterior and the tumor interior. One contour means the tumor contains only one region.

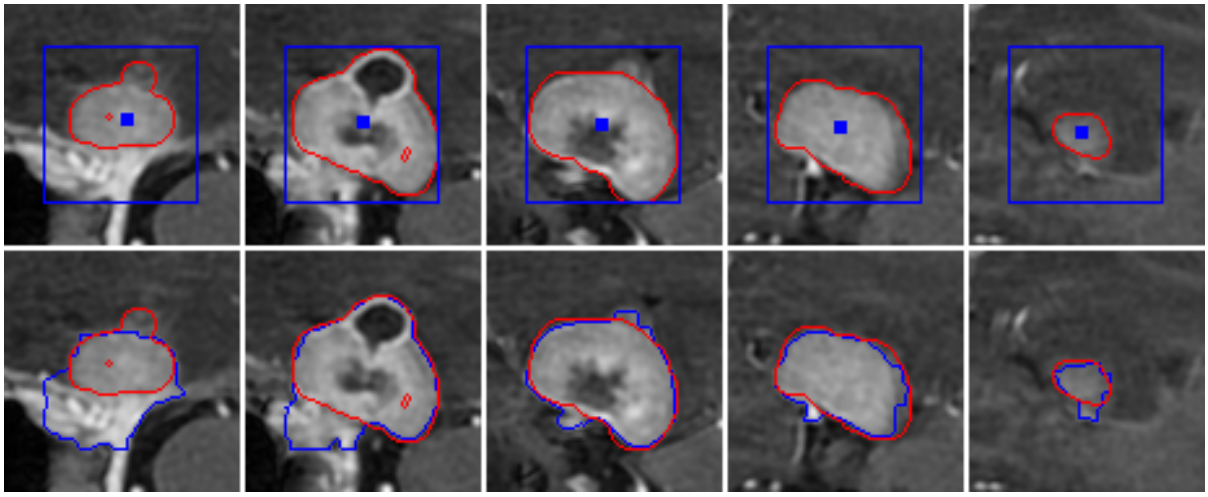


(a)

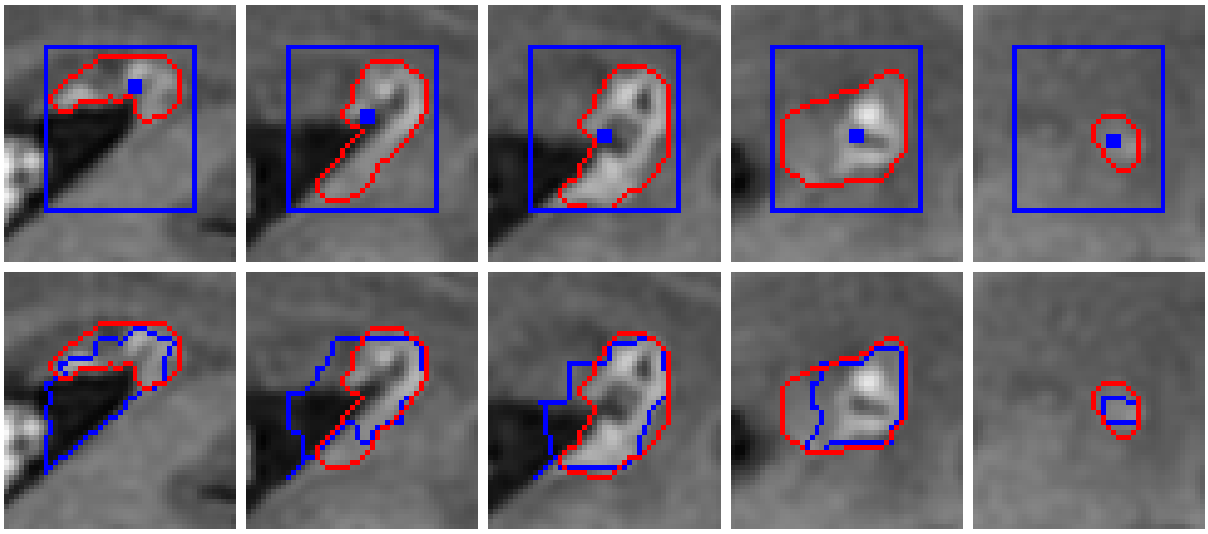


(b)

Figure 5.9: Examples where volume ballooning make segmentations worse. Blue dots are star shape centers. Five volume slices are shown. For each example, the first row is the ground truth with bounding box and star shape seeds. The second row is result multi-region segmentation without volume ballooning and the third row is the result from multi-region segmentation. Green contour outlines the boundary between tumor exterior and the background, blue contour outlines the boundary between the tumor exterior and the tumor interior. One contour means the tumor contains only one region.

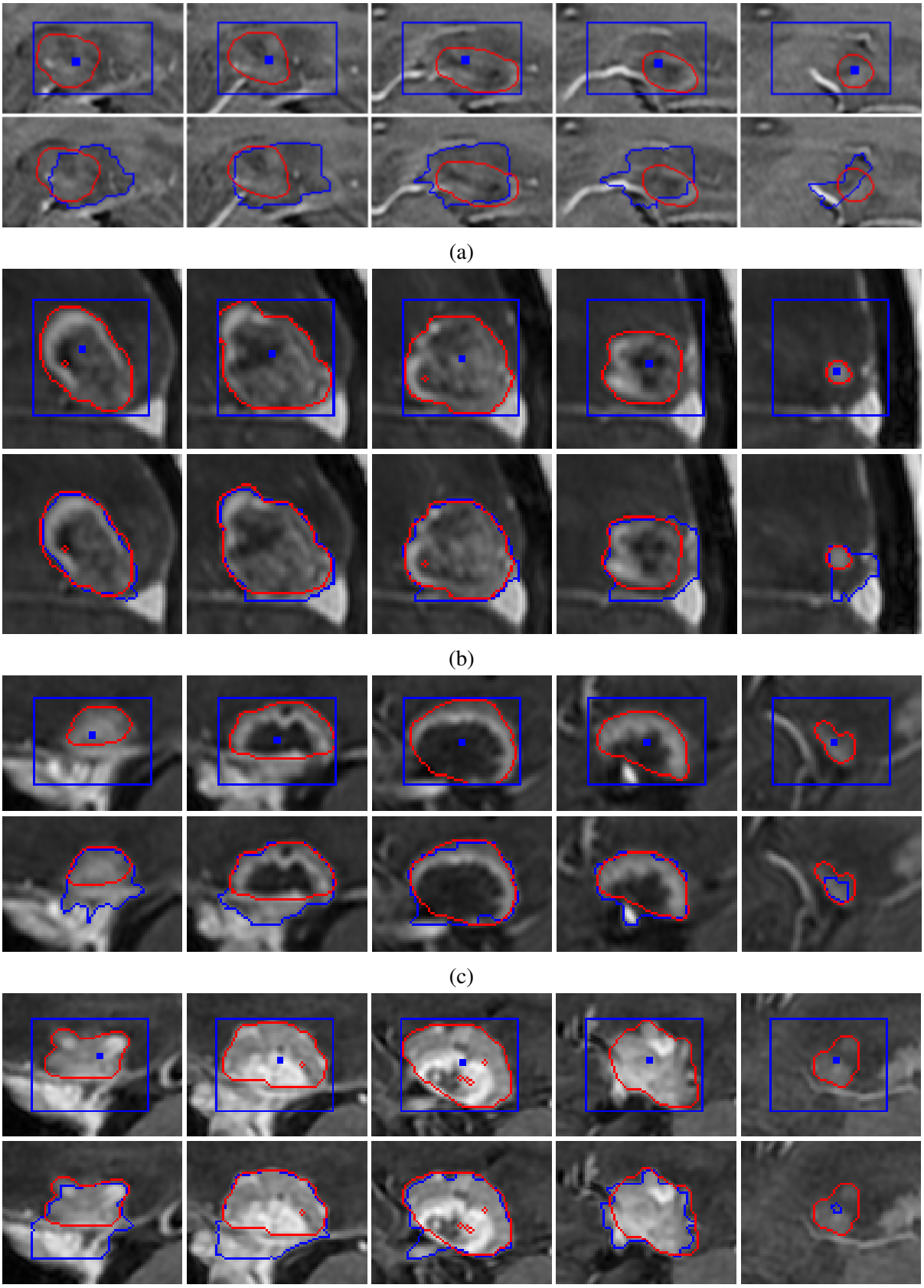


(a)



(b)

Figure 5.10: Two examples of inaccurate segmentation. Five volume slices are shown. For each example, the first row is the ground truth with bounding box and star shape seeds. The second row is the ground truth and our result.



(a)

(b)

(c)

(d)

Figure 5.11: Four examples of inaccurate segmentation. Five volume slices are shown. For each example, the first row is the ground truth with bounding box and star shape seeds. The second row is the ground truth and our result. Red contour is the boundary of ground truth and blue contour is the result of our segmentation.

Chapter 6

Future work and conclusion

6.1 Future work

This section we discuss some possible improvements that we can make to current method.

6.1.1 Background model

One of the largest difficulties of segmentation is that the foreground and background models often have a large appearance overlap. This problem can be made even worse if the regions from which we model foreground and background regions are chosen in a way that increases the overlap even more. For example, if we inadvertent take the background sample that is too large, it may contain objects that are close in the appearance to the tumor, and thus worsen the appearance overlap with the tumor. The initial background model is constructed from pixels outside of the bounding box, from a band that is 1/4 of the width of the user-provided box. The total volume of pixels used to model background is larger than that for the foreground. At the second iteration, to the pixels used for modeling the background we add all those pixels outside the tumor region segmented at the first iteration. Its volume is even larger than that of the initial background. Thus this large background could include some objects with similar appearance to the tumor and the overlap could be enlarged. One idea we can try is that adaptively select some patches from outside the bounding box and/or outside the segmentation boundary, in a way that maximizes the difference of these patches from the pixels interior to the center of the initial box or pixels inside the first segmentation boundary. By doing this, we could potentially reduce the overlap of background and foreground appearance models.

6.1.2 Adaptive appearance model binning

Another idea we plan to try that is also aimed at making the foreground/background models more distinct is as follows. Currently, for modeling appearance, we use intensity histograms smoothed with Parzen windows technique. However, this approach can smooth out very subtle foreground/background histogram differences, making segmentation more difficult. Histogram binning can produce smoothing effects similar to that of Parzen windows. We plan to explore adaptive histogram binning. That is the idea is to search for bins, not necessarily uniform, that make foreground and background models more distinct. This is, of course, chicken-and-egg problem, since we do not know the foreground/background segmentation, which is our main goal. Thus, we plan to incorporate adaptive binning into our segmentation energy, so that we are searching both for the appropriate binning and segmentation at the same time.

6.1.3 Adaptive volumetric ballooning

Examining our results, we can conclude that for the large part, segmentations are rather accurate even without volumetric ballooning. Volumetric ballooning is needed mostly for the slices that are further away from the volume center, where appearance tends to be more similar to the background and intensity edges are weaker, compared to that in the middle slices. This prompted us to design volumetric ballooning. However, the way we currently implement volumetric bias sometimes is too strong in the middle slices, which makes their segmentations sometimes much larger compared to the ground truth. Moreover, our volumetric bias only relates to the size of the bounding box, which can be inaccurate.
Instead of adding volumetric bias everywhere, we could first try segmentation without the bias, and then examine the results. If some slices are clearly under-segmented (i.e. empty tumor region), we then could add volumetric bias only to these slices, that is, adaptively. This scheme would require some measure of undersegmentation. For example, we could define the minimum expected tumor area for the slices as a relationship to the area of the middle slice, since the middle slice usually has a reliable segmentation. These ratios could be learned from the ground truth, or set by hand.

6.1.4 Segmentation as a tracking problem

One of the main challenges of segmentation is that in many cases appearance between the middle slice (i.e. where user provided the bounding box) changes significantly from that of the other slices. This is similar to the well-known ”drift” [14, 25, 19] problem in motion tracking, where the tracked target can change the appearance significantly between the frames. We

can explore the techniques developed for the drifting problem to our segmentation problem. However, our segmentation often changes abruptly in two nearby slices. We can take the idea of tracking into our system and try to get smoothly changed segmentation in different slices.

6.1.5 System options and further editing

Although we have tried different constraints in graph cuts segmentation, their combination is not necessarily the ideal scenario for all the different tumor cases. However, we can make each of these constraints optional in our system so that users can try different constraints and select their subset that yields the best result. In addition, we plan to incorporate further editing with foreground/background seeds into the final system so that the user can get the segmentation of the desired accuracy level.

6.2 Conclusion

We have developed a system that can segment the tumor from the healthy tissues accurately with the minimum user assistance. Our system requires only 4 clicks with 2 clicks specifying a rectangle containing the tumor on all slices, and the other 2 clicks specify the first and last slices. Our system utilizes the basic graph cuts based binary segmentation and also brings in new terms such as star shape constraint and volumetric ballooning. Star shape constraint helps to achieve near convex segmentation and volumetric ballooning helps to get segmentation with desired volume size. We also try multi-region framework to capture tumors with distinct exterior and interior appearance.

Bibliography

[1] http://cs.brown.edu/~pff/segment/.

[2] http://docs.gimp.org/ca/gimp-tool-iscissors.html.

[3] http://www.cs.helsinki.fi/group/goa/mallinnus/lines/bresenh.html.

[4] Serge Belongie, Chad Carson, Hayit Greenspan, and Jitendra Malik. Color-and texture-based image segmentation using em and its application to content-based image retrieval. In *Computer Vision, 1998. Sixth International Conference on*, pages 675–682. IEEE, 1998.

[5] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

[6] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112. IEEE, 2001.

[7] Jack E Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems journal*, 4(1):25–30, 1965.

[8] Mark A Brown and Richard C Semelka. *MRI: basic principles and applications*. Wiley. com, 2011.

[9] Dana Cobzas, Neil Birkbeck, Mark Schmidt, Martin Jagersand, and Albert Murtha. 3d variational brain tumor segmentation using a high dimensional feature set. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[10] B. Porteous D. Greig and A. Seheult. Exact maximum a posteriori estimation for binary iamges. *Journal of the Royal Statistical Society, Series B*, pages 51(2):271–279.

[11] Andrew Delong and Yu Boykov. Globally optimal segmentation of multi-region objects. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 285–292. IEEE, 2009.

[12] Colin Flanagan. The bresenham line-drawing algorithm. *Helsingin Yliopisto*, 2009.

[13] L. Ford and D. Fulkerspn. *Flows in Networks*. Princeton University Press, 1962.

[14] Juergen Gall, Bodo Rosenhahn, and H-P Seidel. Drift-free tracking of rigid and articulated objects. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[15] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum flow problem. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, number STOC '86, pages 136–146, New York, NY, USA, 1986. ACM.

[16] Vladimir Kolmogorov and Ramin Zabin. What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):147–159, 2004.

[17] Eric N Mortensen and William A Barrett. Interactive segmentation with intelligent scissors. *Graphical models and image processing*, 60(5):349–384, 1998.

[18] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[19] Ali Rahimi, L-P Morency, and Trevor Darrell. Reducing drift in parametric motion tracking. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 315–322. IEEE, 2001.

[20] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 309–314. ACM, 2004.

[21] Bernard W Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

[22] Josef Sivic, Bryan C Russell, Alexei A Efros, Andrew Zisserman, and William T Freeman. Discovering objects and their location in images. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 370–377. IEEE, 2005.

[23] Olga Veksler. *Efficient graph-based energy minimization methods in compute vision*. PhD thesis, Cornell university, 1999.

[24] Olga Veksler. Star shape prior for graph-cut image segmentation. In *Computer Vision–ECCV 2008*, pages 454–467. Springer, 2008.

[25] Qing Wang, Feng Chen, Wenli Xu, and Ming-Hsuan Yang. Online discriminative object tracking with local sparse representation. In *Applications of Computer Vision (WACV), 2012 IEEE Workshop on*, pages 425–432. IEEE, 2012.

Curriculum Vitae

Name:	Liqun Liu
Post-Secondary Education and Degrees:	University of Western Ontario London, ON, CA 2012 - Present, M.Sc. in computer science Xi'an Jiaotong University Shaanxi, China 2008 - 2012 B.E. in Automation
Honours and Awards:	WGRS, Western University, 2012 -2013 Outstanding Graduate, Xi'an Jiaotong University, 2012
Related Work Experience:	Teaching Assistant, Western University, 2012 -2013 Research Assistant, Computer Vision Group, Western University, Sep. 2012 - Jan. 2014