April 2016

Interactive Brain Tumor Segmentation with Inclusion Constraints

Danfeng Chen
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

Abstract

This thesis proposes an improved interactive brain tumor segmentation method based on graph cuts, which is an efficient global optimization framework for image segmentation, and star shape, which is a general segmentation shape prior with minimal user assistance. Our improvements lie in volume ballooning, compactness measure and inclusion constraints.

Volume ballooning is incorporated to help to ”balloon” segmentation for situations where the foreground and background have similar appearance models and changing relative weight between appearance model and smoothness term cannot help to achieve an accurate segmentation. We search different ballooning parameters for different slices since an appropriate ballooning force may vary between slices.

As the evaluation for ”goodness of segmentation” in parameter searching, two new compactness measures are introduced, ellipse fitting and convexity deviation. Ellipse fitting is a measure of compactness based on the deviation from an ellipse of best fit, which prefers segmentation with an ellipse shape. And convexity deviation is a more strict measure for preferring convex segmentation. It uses the number of convexity violation pixels as the measure for compactness.

Inclusion constraints is added between slices to avoid side slice segmentation larger than the middle slice problem. The inclusion constraints consist of mask inclusion, which is implemented by an unary term in graph cuts, and pairwise inclusion, which is implemented by a pairwise term. Margin is allowed in inclusion so that the inclusion region is enlarged.

With all these improvements, the final result is promising. The best performance for our dataset is 88% compared to the previous system in [25] that achieved 87%.

Keywords: Interactive segmentation, graph cuts, star shape, volume ballooning, ellipse fitting, convexity, inclusion

Acknowledgement

First of all, I would like to express my sincere gratitude to my supervisor Dr. Olga Veksler who is a great mentor and brings me to computer vision this interesting field. I really appreciate her patience when explaining methods to me and when looking through the experimental results. She can always spot the problems in the results when I was too careless to notice. And every time when we get stuck she can always come up with new ideas to help us move forward. Her attitude and dedication to work and research has influenced me a lot. Without her guidance, this thesis cannnot be accomplished.

Second, I want to appreciate those who gave me significant help with my study and thesis. I am so grateful to Dr. Lena Gorelick who helped to integrate convexity into my project and carefully looked into the code to find the underlying problem. And also Dr. Yuri Boykov, who gave me a deep understanding of the algorithms in computer vision through his detailed explanation in class. Their passion and expertise is what I would always look up to. I also want to thank Dr. Aaron Ward for providing us with valuable advice for improving our approach and the extension in 3D Slicer. In addition, I really appreciate the "distant" help from Ipek Oguz, Steve Pieper and other developers in 3D Slicer community when I was transforming the code to 3D Slicer extension. Their help is really important for applying our approach into practical medical image processing software.

Next, I would like to give my gratefulness to other members in our vision group. Thanks Xinze Liu, Egor Chesakov, Meng Tang and Jiaqi Zhou for being such good labmates and offering help and discussing with me when I encounter problems. I want to give special thanks to Liqun (Rachel) Liu who did the prior work of this thesis. She always patiently answers my question and gives me useful suggestions. Thanks all of my other friends as well, I really enjoy hanging out and having fun with you.

Last but not the least, I want to thank my family, especially my great parents, for giving me continuous encouragement, support and love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Image analysis for brain tumor studies is an essential task for medical diagnosis, patient monitoring and treatment planning. The segmentation is crucial for monitoring tumor growth or shrinkage in patients during therapy and tumor volume measurements. And it also plays an important role in surgical planning or radiotherapy planning, where not only the tumor has to be outlined, but also surrounding healthy structures are of interest[2]. The manual segmentation of brain magnetic resonance imaging (MRI) can be complex and tedious work. Manual analysis is time-consuming and prone to errors as well due to the large amount of data. These facts make computerized brain tumor segmentation method desirable.

Computerized brain tumor segmentation methods usually fall into two categories: automatic segmentation and interactive segmentation. In fully automatic segmentation methods, an algorithm is developed to determine the segmentation of tumor without any human. The main challenge in automatic segmentation is obtaining accurate appearance models for the object of interest. One can obtain an appearance model through training on manually labeled data. However, quite often a large amount of training data is needed to obtain sufficiently accurate models. Large amounts of labeled data may be unfeasible to obtain. Another issue is lack of interpretability and transparency in the automatic segmentation process, and it restricts its wide acceptance among the practitioners (radiologists, neurologists and other medical experts) for daily clinical use[15].

Interactive segmentation, on the other hand, effectively combining both computers and humans' expertise, offers an attractive approach to overcoming the limitations of fully automatic methods. Rather than relying on large training data set, it requires users' initialization like specifying object of interest or adding "seeds" indicating the labels of pixels. Moreover, the interactive segmentation also allows users to evaluate the result and edit the final segmentation if it is not satisfactory. Iterative segmentation process can be repeated until no further modification is needed.

To develop a more accurate brain tumor segmentation method for daily clinical practice, an interactive segmentation approach will continue to be popular. This thesis takes a previously successful interactive brain segmentation system [25] and improves it further in several directions.

1.1 MRI and Brain Tumor

The image data used in this thesis is acquired with MRI technology. A magnetic resonance imaging (MRI) scan is a radiology technique that uses magnetism, radio waves, and a computer to produce images of body structures[7]. The image and resolution produced by MRI is quite detailed and can detect tiny changes of structures within the body. Detailed MR images allow physicians to evaluate various parts of the body and determine the presence of certain diseases.

MR image of a brain is a 3D scan of a human brain or a sampling of the brain structure in three different dimensions. In Figure 1.1, we show a 3D MRI volume of a patient. The 3D MR image can be also be thought of as a 3D volume composed by a series of slices in one direction. The intensity of a pixel is proportional to the nuclear magnetic resonance signal intensity of the contents of the corresponding volume element or voxel of the object being imaged[21]. The size of the element determines the spatial resolution, or the scale of detail that can be distinguished in an image. Voxel/pixel sizes may vary, depending on imaging parameters, magnet strength, the time allowed for acquisition, and other factors[11].

1.2 Challenge for Brain Tumor Segmentation

The difficulty for brain tumor segmentation mainly lies in the fact that brain tumors usually have a great variety in size, shape, and in image intensities. Some tumors also deform surrounding structures and appear together with edema or necrosis that changes intensity properties of the nearby region [30]. Below we enumerate some of these challenges as well as give examples from our dataset.

Diversity in Size, Location and Shape

The size of the brain tumor varies from patient to patient. Even the tumor within the same patient has size change in different treatment stages. The same goes for tumor location. The variance of tumor size and location can be easily noticed when going through the 3D MRI brain volume. As for tumor shape, it is more likely to be seen in 2D projection images. Figure

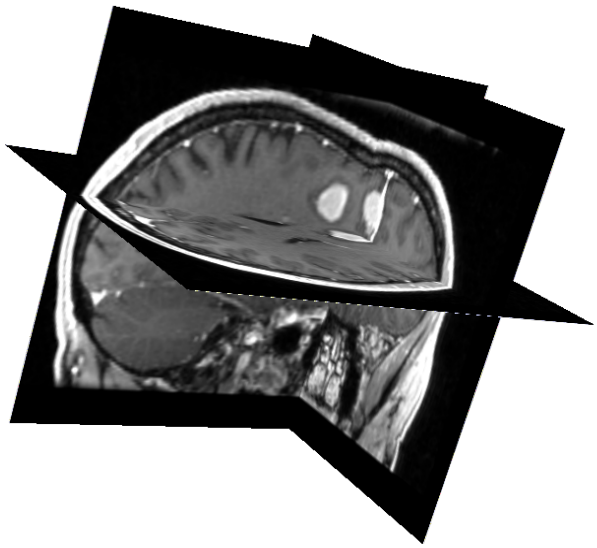


Figure 1.1: A MRI brain volume in 3D

1.3 shows three tumors with different shape. The diversity of brain tumor in size, location and shape makes it difficult for automatic segmentation to have satisfying segmentations.

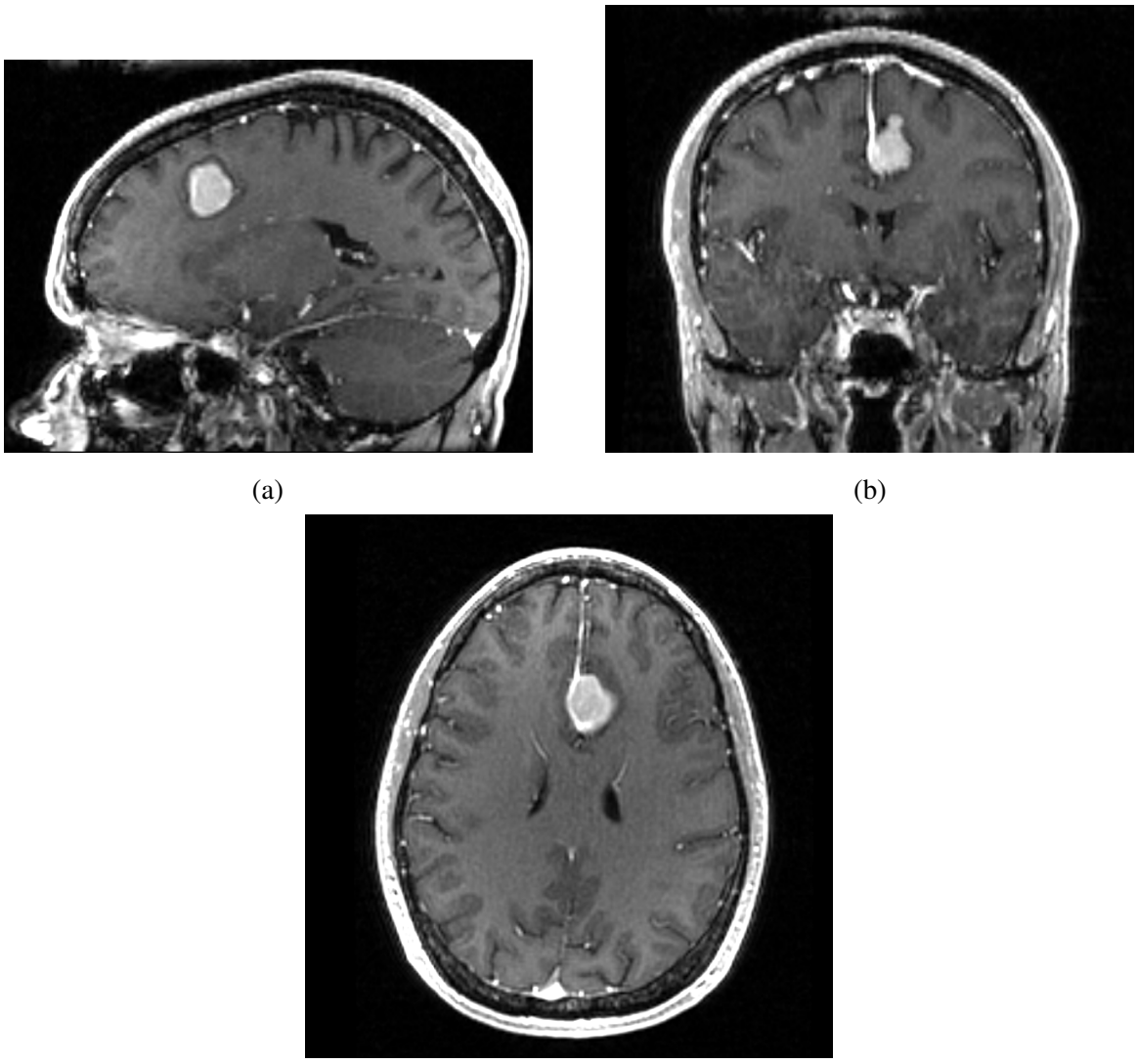
Inconsistency in Appearance

Brain tumors are usually inconsistent in appearance. Some tumors have higher intensities than the background while others may be darker. In some cases, a tumor can appear virtually indistinguishable from its background, which makes segmentation even in interactive setting very challenging. Figure 1.4 lists these three kinds of tumors.

Complexity in Structure

Brain tumors are complex in structure, especially when there are lots of surrounding tissues. Connections to surrounding structures often blur the tumor boundaries. Therefore, it is hard for the segmentation algorithm to capture the actual tumor region. Figure 1.5 shows examples where tumor has complex surrounding tissue and blurry boundary.

Challenges in brain tumor segmentation motivate research for improving segmentation methods. In this thesis, an improved brain tumor segmentation approach will be introduced.



(a) (b)

(c)

Figure 1.2: Brain MRI slices in 3 different dimensions

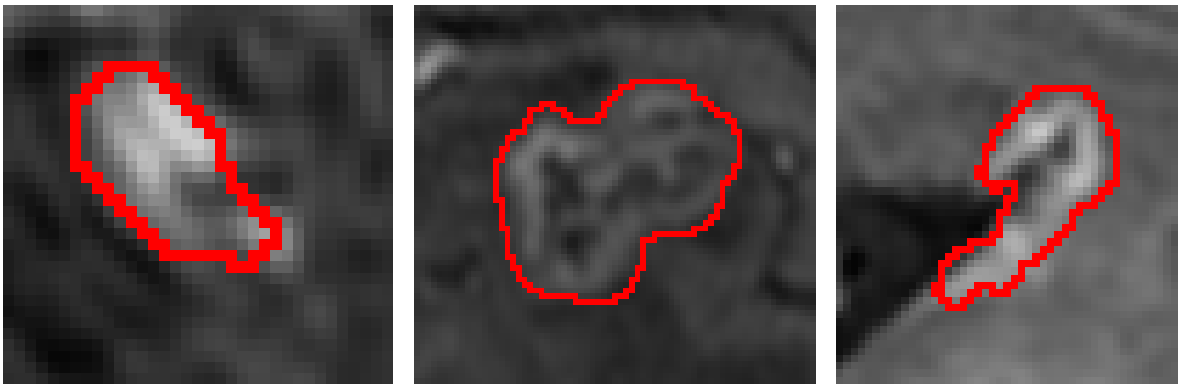


Figure 1.3: Examples of tumors in different shapes.

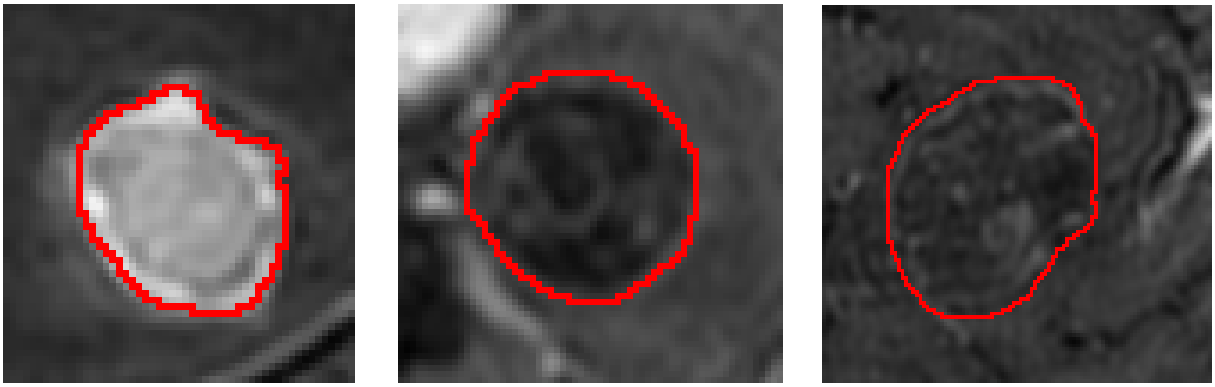


Figure 1.4: Examples of tumors with different appearance.

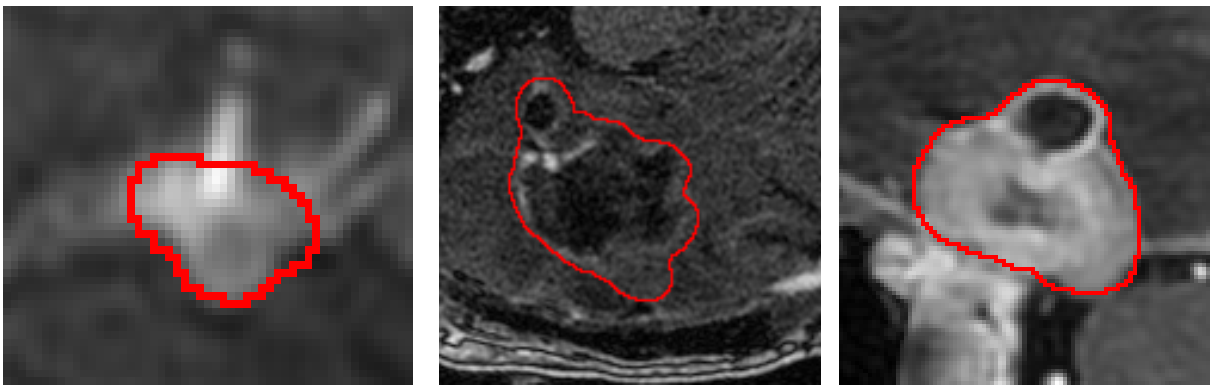


Figure 1.5: Examples of tumors with different appearance.

1.3 Overview of MRI-based Brain Tumor Segmentation

In [2], they summerize the state of art segmentation methods for brain MRI scans. The common procedure of automatic or semi-automatice brain tumor segmentation include four steps, preprocessing, feature extraction, segmentation and postprocessing. In preprocessing stage, certain image enhancement approaches such as denoising, intensity normalization and skull-stripping can be used to improve image quality. With enhanced image, common image features like image intensity, local image texture, alignment-based features and intensity gradient can be either computed from one single modality or from multi-modal images. Different features can be used alone or combined together for brain tumor segmentation.

Segmentation algorithms can be categorized into region or edge based methods and classification and clustering methods according to the features they adopt. In region or edge based segmentation, there are methods make use of deformable models based on regional characteristics or edge detection in the image. For example, [37] employed a fluid vector flow model to evolve a contour toward the broundary and [33] adopt content based intensity and texture patterns to evolve an active contour toward tumor boundary. [19] proposed a fuzzy-connectedness algorithm making use of region properties but no spatial constraints. Other region or edge methods like [20] and [31] also utilize multi-modal images for segmentation.

Most brain tumor segmentation methods proposed so far are based on cliassification or clustering because they can handle multi-modal datasets easier. These methods ususally use both voxelwise intensity and local textures. The general idea is to decide for every voxel which class it belongs to according to its feature vector. Classification requires training data to learn a classification model, based on which new instances can be labeled. Clustering, on the other hand, works in an unsupervised way and groups data based on certain similarity criteria [36]. Simple voxelwise classification or clustering methods are remarked not make use of complete image information. Therefore, additional constraints such as neighborhood regularization and shape and localization constraints are added in some methods. Neighborhood constraints are often imposed using a random field regularization method, while shape constraints are mostly handled by deformable models [2].

Apart from the traditional segmenation techiques, there is also atlas-based segmentation methods relying on registration. Atlases can be used to restrict the tumor location and also for generative classification models. They have advantage when segmenting tissues and structures surrounding the tumor. But traditional techniques are more flexible, more easily adapted to handle multiple modalities simultaneously and to treat individual tumor subcompartment.

1.4 Our Approach

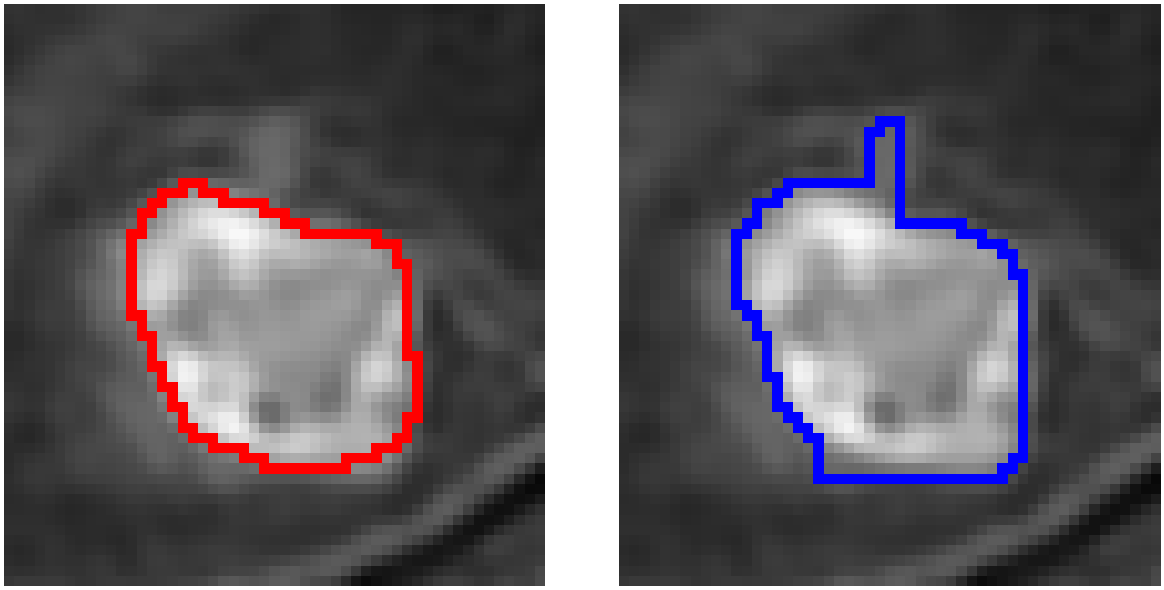
In this thesis, we introduce an interactive brain tumor segmentation approach with inclusion constraints. Our approach is an improvement over the interactive segmentation algorithm proposed in [25]. One of the main advantages of [25] is that it requires minimum user assistance. To be more specific, the user only needs to provide four clicks, two clicks at the center of two end slices and another two clicks in the middle slice where there is a largest tumor region, to specify a bounding box for the initialization of the segmentation.

The basic segmentation framework of this thesis is still using graph cut energy minimization method with the frequently used data term, smoothness term and star shape prior. Data term, also called regional term, contains regional appearance information. Smoothness term, or boundary term, encodes the coherence information among image pixels. Star shape prior restricts the segmentation to have a shape constraint with respect to a center within the object. Besides the basic framework, additional constraints and measurements are developed in this thesis. The main improvements we have are as follows

- Introduce a search for optimal parameter when performing volume ballooning.
- Develop a measure of compactness that helps to achieve higher accuracy.
- Introduce inclusion constraints between neighboring slices that help to achieve more accurate segmentation.

To reduce the shrinking effect of the graph cut algorithm, we introduce a ballooning bias. Volume ballooning adds ballooning bias to the energy function so that the pixel can get a bonus if it is assigned to the foreground. Volume ballooning is encoded as an unary term in energy function and can be easily integrated into the energy optimization framework. We combine the ballooning bias with the local information of each slice. Therefore the ballooning is adjusted according to the characteristic of each slice.

The parameter searching process requires an appropriate measure of segmentation quality. Thus, different compactness measure methods are discussed in our work to get the segmentation closer to an actual tumor shape. A novel compactness measure using ellipse fitting utilizes the ellipse shape property of brain tumor. It measures the deviation of the segment from an elliptical shape. Ellipse fitting is a simple geometric computation and does not increase computational complexity of the original algorithm. One drawback of the ellipse fitting is that it allows non-convex segmentation while most tumors are usually close to convex objects, as shown in Figure1.6. To prefer convex objects more strongly, we adopt the convexity deviation as another compactness measure. The main idea is to count the number of pixels violating



(a) Ground truth (b) Segmentation with ellipse fitting

Figure 1.6: Ellipse fitting allows non-convex segmentation.

convexity in a binary labeling and select the one with the smallest number. There is a fast method to implement this using dynamic programming which only requires linear computational time. Therefore there is still not too much additional computational complexity for the original algorithm.

The ground truth of our data set gives us a 3D characteristic of brain tumor that the tumor region usually decreases from the middle to the side, as displayed in Figure 1.7. Hence, an

Figure 1.7: Tumor region decreases from middle slice to end slices. The red contour is the ground truth and the slice with yellow rectangle is the middle slice.

additional inclusion constraint is imposed between neighboring slices so that the segmentation of the slice far from the middle should be within some regional restriction. Inclusion constraints include two parts, the mask inclusion, corresponding to the unary data term, and the pairwise constraint inclusion, implemented by a additional binary constraint term. The new inclusion constraints in our approach can be optimized by graph cut algorithm requiring no extra computational complexity.

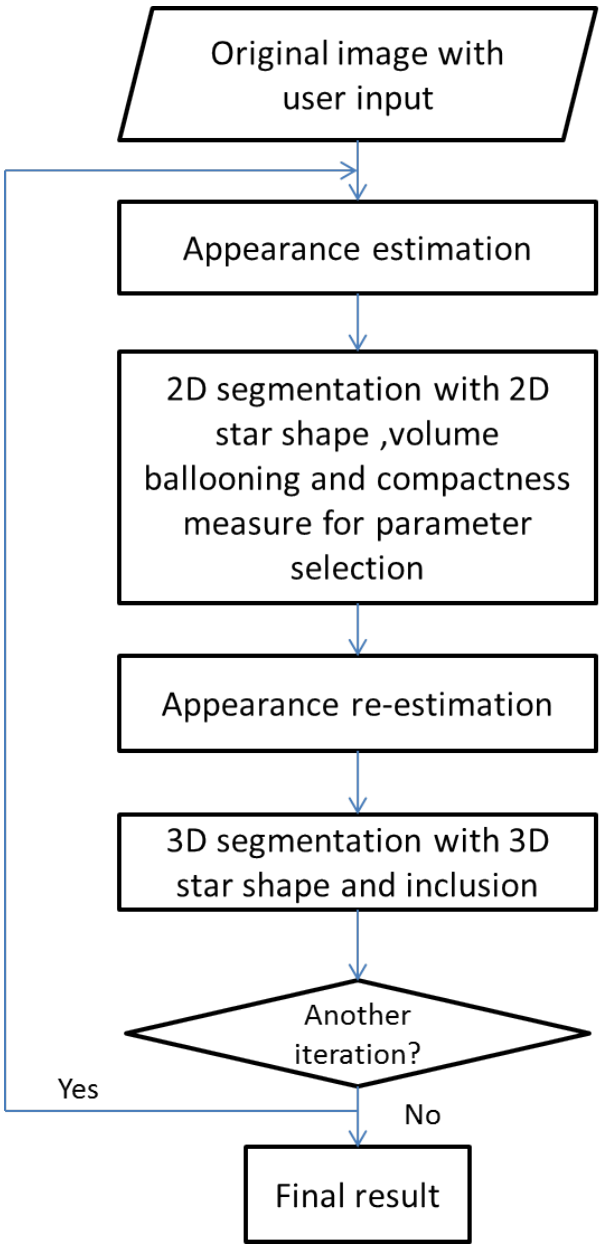


Figure 1.8: The flow chart of the algorithm.

1.5 Outline of This Thesis

This thesis is grouped as follows: **Chapter 2** is a review of the related works. It includes *graph cut energy minimization framework, interactive graph cut segmentation, grab cut, star shape prior, 3D slicer* and most importantly the prior work on interactive brain tumor segmentation. **Chapter 3** introduces the details of the improvements we have based on the previous work, mainly lie in volume ballooning, compactness measure of segmentation and inclusion constraints between slices. **Chapter 4** is all about the experimental results. It also explains about the parameter selection process, result evaluation method and result comparisons. In **Chapter 5**, we give a conclusion for this thesis and plan the future work.

Chapter 2

Related Work

2.1 Energy Minimization Framework

Interactive segmentation methods can be classified into two groups contour based and region based. Contour based approaches such as snakes [22], deformable template [38], methods computing the shortest path [27] cannot be easily generalized to images of higher dimensions and are often limited to local minimum[3]. Even though the segmentation for 3D images can be obtained by segmenting 3D volume slice by slice, the boundaries in each slice are independent which can result in spatial incoherence.

Region based techniques like region growing and split-and-merge [18] build the segmentation based on information inside the segments rather than at the boundary which can lead to "leak" in places with weak boundary.

There are also methods combining the benefits of both region and contour information. These methods associate segmentation with energy minimization problem. There is usually an energy function maps the solution to a real number reflecting the solution goodness. We minimize the energy function to obtain the optimal segmentation.

2.1.1 Energy Function

Segmentation, like many other computer vision tasks, can be thought of a labeling problem in which each pixel is assigned a label in some finite set L. The goal is to find a labeling f that assigns each pixel $p \in P$ a label $f_p \in L$, where f is both piecewise smooth and consistent with the observed data. This problem can be expressed naturally in terms of energy minimization[5]. A standard energy function can be formulated as following.

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

In Equation 2.1,$E_{data}(f)$ is called data term. It measures the disagreement between current pixel and observed data. For instance, it may reflect on how the intensity of pixel p fits into a known intensity model. $E_{smooth}(f)$ is the smoothness term. It measures the piecewise smoothness. It can be interpreted as a penalty for a discontinuity between pixel pairs. The more similar the pixel pair is, the greater the penalty they get to have discontinuity. And λ can adjust the relative importance between data term and smoothness term.

In the standard energy function, there are only two terms, i.e. unary data term and pair-wise smoothness term. But one can always define new energy term adding to the basic one to have additional constraints.

2.1.2 Optimization with Graph Cut

Graph cut is a popular technique in computer vision for energy minimization and its application is not only limited to segmentation. It can also be applied to other computer vision problems like image restoration, stereo correspondence and other ones which can be formulated in terms of energy minimization[34].

The graph cut method can guarantee a global minimal solution for certain energy functions. The other important feature is that the energy function used in graph cuts can be easily extended to an ND setting. Moreover, graph cut framework is able to integrate other constraints like shape prior and other geometric restrictions. Hence, graph cut is adopted as the basic segment algorithm for this work.

Let $G = \{V, E\}$ be an undirected graph where V is a set of vertices while E is a set of weighted edges connecting the nodes in V. Apart from the normal vertices, there are two terminal vertices, the source s and the sink t. An s-t cut, $C = (S, T)$, is partition of the vertices such that all vertices are divided into two disjoint sets, $s \in S$ and $t \in T$. The minimum s-t cut problem is to find a cut C with the smallest cost.

Figure 2.1 shows a simple graph with its s-t cut. It is a 3 by 3 graph with two terminal vertices s and t. The thickness of the edges represent the corresponding weights. Thick edge stands for high weight while thin edge has rather low weight. The green dashed curve shows a minimal s-t cut on this graph. According to the theorem of Ford and Fulkerson [13], max-flow problem is equivalent to min-cut problem. From Figure 2.1, we can intuitively understand the connection between min-cut on a graph and the energy minimization over image labeling. If each edge's weight is in proportion with the energy, a cut with minimum cost corresponds to a labeling with the minimum value of this energy.

There are straightforward implementations of the standard graph cut algorithms, like Ford-Fulkerson augment-path algorithm [13] and push-relabel algorithm [14]. But they do not have

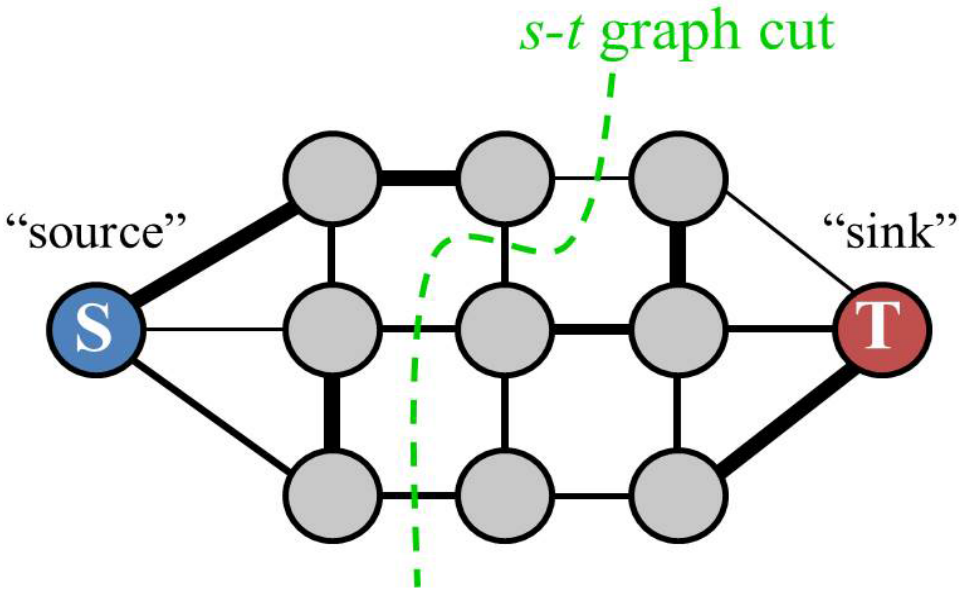


Figure 2.1: A *s-t* cut on graph with two terminals. [Image credit: Yuri Boykov]

good performance in terms of computational time. The max-flow/min-cut algorithm of [4] can find the minimum cut in practice in linear time for the types of graphs arising in computer vision and our work is based on it.

2.1.3 Binary Segmentation with Graph Cut

Our focus for this thesis is separating the brain tumor from other surrounding tissue. Thus, the goal is to do binary segmentation. Each pixel needs to be assigned with a label from label set $L = \{0, 1\}$, where 1 means the object and 0 represents the background. The energy function for binary segmentation should look like follow

$$E(f) = \sum_{p \in P} D_p(f_p) + \lambda \cdot \sum_{p,q \in N} V_{pq}(f_p, f_q) \tag{2.2}$$

where P is the set of all pixels in the image, N is the standard 4 or 8-connected neighborhood system on P, and pixel pairs (p, q) are neighbor pixels in corresponding neighborhood system. $f_p \in L$ here represents the label assigned to pixel p.

In Equitation 2.2, $V_{pq}(f_p, f_q)$ is the penalty for neighbor pixels when they have different labels. Typically,

$$V_{pq}(f_p, f_q) = \omega_{pq} \cdot \delta(f_p, f_q) \tag{2.3}$$

where

$$\delta(f_p, f_q) = \begin{cases} 1 & \text{if } f_p \neq f_q \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

For pairwise term, when neighbor pixels are assigned the same label, there is no penalty for them. Otherwise they should pay for having different labels. This means

$$V_{pq}(0,0) = 0 \tag{2.5}$$

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

Consider a simple 3 by 3 image as an example for binary segmentation. As seen in Figure 2.2, first a graph based on the original image needs to be constructed. Each pixel corresponds to a node in the graph. Apart from all the nodes in the image, there are two extra terminal nodes, the source s and the sink t, which correspond to the set of labels that can be assigned to the pixels. The edges connecting the nodes are all with weights. There are two kinds of edges in the graph. One is between neighbor pixels, usually called *n-links*. The weights of *n-links* reflect to the penalty for discontinuity between pixels which is derived from the pair-wise smooth term in the energy function. The other kind of edge connects the pixel node and the terminal node, often called *t-links*. The costs of *t-links* represent the similarity between each pixel and the terminal label derived from the data term in the energy function.

Utilizing graph cut to optimize the energy containing in the graph built previously, a minimum cut, separating the pixel nodes in the image into two disjoint parts, can be obtained. The two separated parts indicate the binary segmentation, as shown in the result in Figure2.2.

Although graph cut is for energy minimization problem, not all kinds of energy functions can be optimized by it. Kolmogorov [24] proved that the energy function can only be optimized by graph cut when it is submodular. In binary label case, the energy function needs to satisfy

$$V_{pq}(0,0) + V_{pq}(1,1) \leq V_{pq}(1,0) + V_{pq}(0,1) \tag{2.7}$$

Here, $0, 1$ is the label set and V_{pq} is the pair-wise term in energy function with the respect of pixel p and q. Otherwise, the energy optimization is a NP hard problem. It is easy to see from Equation 2.3 that the energy function for binary segmentation satisfies the submodular regulation. Therefore, it can be optimized by graph cut.

2.2 Interactive Segmentation Approach

Interactive segmentation is a popular approach when the appearance model of the object is unknown prior to segmentation.

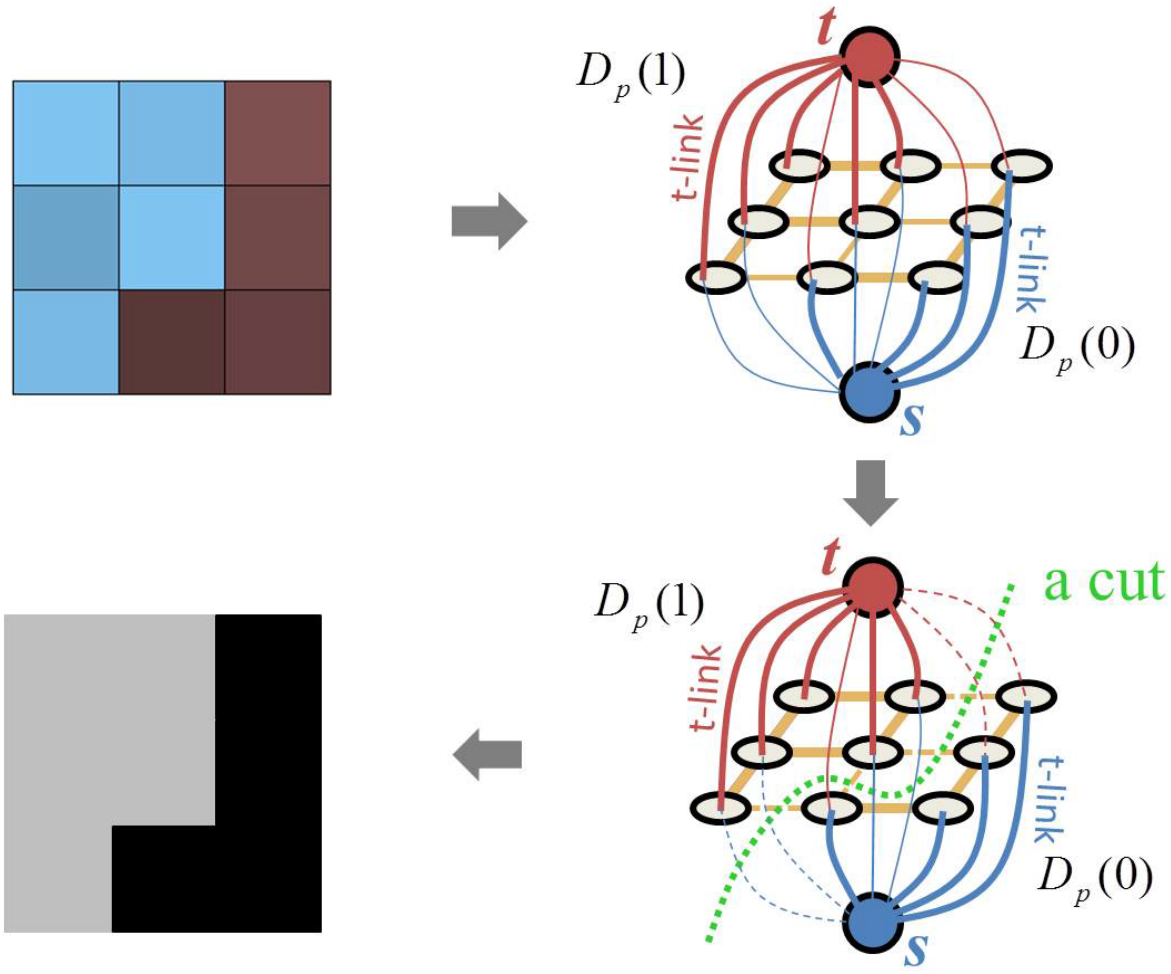


Figure 2.2: Binary segmentation for 3 by 3 image. (top-left): Original image; (top-right): Graph constructed based on original image with extra *s* and *t* terminals; (bottom-right) A minimal cut for the graph separating all pixels into two disjoint sets; (bottom left): Segmentation result, one color stands for one label. [Image credit: Yuri Boykov]

2.2.1 Interactive Graph Cut Segmentation

Interactive graph cut segmentation is first proposed by Boykov and Jolly in 2001 [6]. Their basic idea is to use user provided strokes as the hard constraints and then globally optimize the energy with the graph cut approach in [4]. The energy function they adopted is the standard one

$$E(f) = \sum_{p \in P} D_p(f_p) + \lambda \cdot \sum_{p,q \in N} V_{pq}(f_p, f_q) \tag{2.8}$$

Unlike other type of hard constraints such as intelligent scissors [27] and live wire [12] to indicate the segmentation boundary, the hard constraints users imposed in interactive graph cut approach do not need a very careful placement precisely on the object boundary by the user. The only requirement is that the object stroke is anywhere inside the object, and the background stroke is anywhere outside the object. More specifically, the "seeds" users added are imposed by the data term. Having the label set $L = \{0, 1\}$ where 0 stands for background (marked by B) and 1 stands for object (marked by O), the data term for hard constraints can be written like this

$$D_p(1) = \begin{cases} K & \text{if } p \in O \\ 0 & \text{if } p \in B \end{cases} \tag{2.9}$$

and

$$D_p(0) = \begin{cases} K & \text{if } p \in B \\ 0 & \text{if } p \in O \end{cases} \tag{2.10}$$

Here, O and B are subsets of pixels labeled as object and background respectively, and K is a large constant so that no cut will pass through the hard constraint link.

In [6], the user "seeds" are double used to get the appearance model of background and object. The pixel marked by the user provide intensity histograms for both background and object, which reflect their intensity distributions $Pr(I|O)$ and $Pr(I|B)$ respectively. The histograms can be regarded as the previous knowledge used to set the data term cost in energy function. The corresponding penalty can be represented by negative log-likelihoods, as below

$$D_p(1) = -lnPr(I_p|O) \tag{2.11}$$

$$D_p(0) = -lnPr(I_p|B) \tag{2.12}$$

The pairwise penalty in this approach depends on the intensity difference between neighbor pixels. Specifically, the smooth term is

$$V_{pq}(f_p, f_q) = \omega_{pq} \cdot \delta(f_p, f_q) \tag{2.13}$$

where

$$\delta(f_p, f_q) = \begin{cases} 1 & \text{if } f_p \neq f_q \\ 0 & \text{otherwise} \end{cases} \tag{2.14}$$

and

$$\omega_{pq} \propto exp(-\frac{(I_p - I_q)^2}{2\sigma^2}) \cdot \frac{1}{dist(p,q)} \tag{2.15}$$

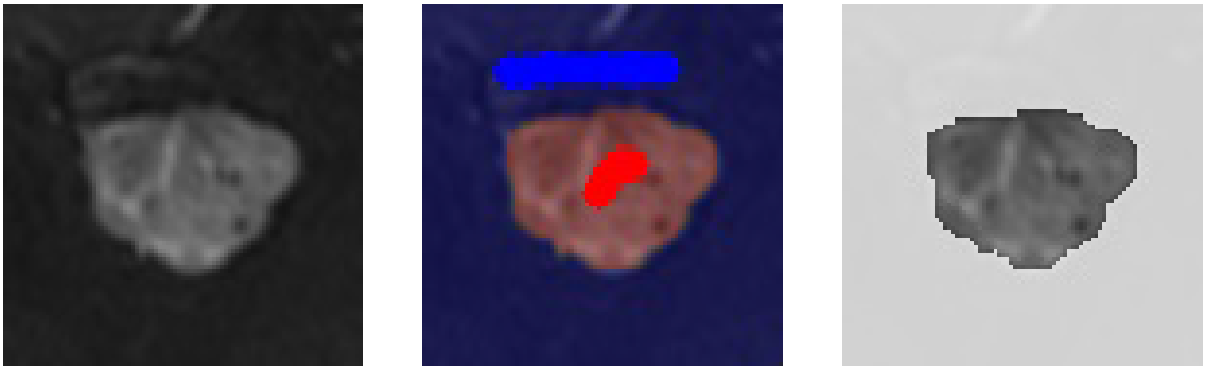
The function for smooth term measures the discontinuity cost between two pixels next to each other. Two similar pixels, i.e. $|I_p - I_q| < \sigma$, will have a large penalty between them while there is only a small cost when two pixels have large intensity difference like the pixels on the image intensity edges. Intuitively, this function corresponds to the distribution of noise among neighboring pixels of an image and σ can be estimated as "camera noise" [6].

Another highlight of this approach is that it can resegment with new user inputs. A segmentation can be obtained based on some initial input. If the result is not satisfying, additional object and background seeds can be added. To incorporate these new seeds the algorithm can efficiently adjust the current segmentation without recomputing the whole solution from scratch. This feature makes this method very intuitive and fast for users to obtain the segmentation they desire.

Last but not the least, the interactive graph cut segmentation can apply to N-dimension images as well, which is quite suitable for our 3D brain tumor segmentation problem. Figure 2.3 shows an example segmented using Boykov's approach.

2.2.2 Grab Cut

Grab cut [32] proposed by Rother et.al. extends considerably the interactive graph cut segmentation approach introduced by Boykov and Jolly [6]. There are three developments within grab cut. First, the appearance model is described by a Gaussian Mixture Model (GMM) rather than histograms. GMM is more suitable for images in RGB color space since it avoid constructing too many color bins. Second, Rother et.al. alternate iteratively between estimation and parameter learning. It means grab cut energy optimizes not only on the segmentation but also on the appearance. This improvement gives grab cut the advantage to allow the automatic refinement for appearance model. And it guarantees proper convergence properties at least to a local minimum [32]. Third, the user only need to provide a rectangle for initialization which means less interaction compared to interactive graph cut method. Grab cut takes all pixels outside the initial rectangle as hard constraint background. Pixels within the rectangle provide the initial appearance for foreground. There is no hard constraints within the box.



(a) Brain tumor image (b) User seeds (c) Segmentation

Figure 2.3: Brain tumor segmentation using Boykov's approach with user seeds. (a) is a cropped slice from brain MRI volume with a tumor in the center; (b) is the user input seeds, the blue stroke stands for background while the red stroke stands for object; (c) is the segmentation result with only object showing. [25]

The grab cut algorithm has three steps. The segmentation is initialized by the rectangle provided by the user containing the whole object that needs to be segmented from the background. With the initial segmentation produced with the rectangle, the algorithm will iteratively reestimate the appearance model and then resegment with a fixed appearance model until converge. Finally, further stoke refinement is still allowed for a desirable result. An example using grab cut is shown in Figure 2.4.

Figure 2.4: An example image segmented by grab cut. (a) The original image with a user input rectangle containing the whole object needs to be segmented; (b) The segmentation result. [32]

2.3 Star Shape Prior

Star shape prior [35] proposed by Veksler, is an additional shape constraint that can be easily added to graph gut algorithm. Performing segmentation with shape prior will make the result more robust if the object is known to have the star shape property. Rather than restricting the shape to a specific type, star shape prior is a generic shape prior which can be applied to a wide class of objects. The definition of a star shape is with respect to a center point. An object has a star shape if for any point p inside the object, as in Figure 2.6, all points on the straight line between the center c and p also lie inside the object [35]. There are some examples of star shape given in [35], as shown in Figure 2.5.

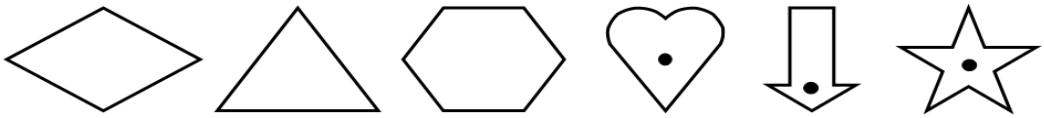


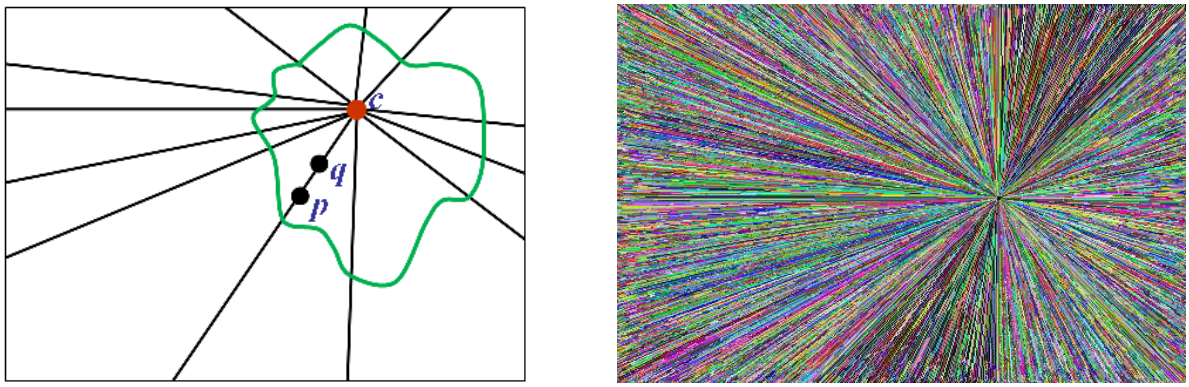
Figure 2.5: Star shape examples. First three shapes are convex and therefore are stars with respect to any inside point as the center. Last three shapes are stars with respect to the specified center, however there are multiple other valid centers [35].

Star shape center is usually provided by the user when performing segmentation. For most star shapes, there is more than one valid choice for a center. For most cases, star shape centers should be within some certain area or along some lines, such as the ”heart” and ”star” image in Figure 2.5. It also should be noted that for convex objects, any point within the object can be the star shape center.

2.3.1 Star Shape Constraint Term

Recall that we can still add other terms to the standard graph cut energy function as long as they satisfy the submodularity condition. Star shape prior can be incorporated into the energy optimization with very little additional complexity. Next, we will discuss implementation details.

Consider the shape with green boundary in Figure 2.6. The red pixel c is the star shape center. There many lines passing through c and pixel p and q are on one line among them. To satisfy the star shape prior, when pixel q is assigned label 1, pixel p cannot have the label 0. Other combinations of labeling for pixel p and q are all allowed. Thus, the additional star shape constraint term can be defined like this



(a) (b)

Figure 2.6: (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.

$$S_{pq}(f_p, f_q) = \begin{cases} 0 & \text{if} \quad f_p = f_q, \\ \infty & \text{if} \quad f_p = 1 \quad \text{and} \quad f_q = 0, \\ \beta & \text{if} \quad f_p = 0 \quad \text{and} \quad f_q = 1 \end{cases} \tag{2.16}$$

It has been proved in [35] that the putting S_{pq} only between neighbor pixels is enough to guarantee the star shape for the entire segmentation, and that the neighborhood system for incorporating the star constraints is the same as for the boundary constraints. Hence, adding star shape constraints to graph cut framework requires almost no extra computation. The value β in Equation 2.16 will be discussed in next section. With the additional star shape constraints, the energy function becomes

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \lambda \sum_{\{p,q\} \in \mathcal{N}} V_{pq}(f_p, f_q) + \sum_{\{p,q\} \in \mathcal{N}} S_{pq}(f_p, f_q) \tag{2.17}$$

Since images consist of discrete pixels, we need to discrete the lines passing through center c. An example of discrete lines passing one center is shown in Figure 2.6b with random color.

2.3.2 Bias Towards Longer Boundaries

Graph cut is well-known for its bias towards shorter boundaries. To reduce this bias, one way is to decrease the weight for pairwise boundary term by changing the relative weight parameter λ in the energy function. Having a large λ in energy function makes boundary term more costly, which will result in a segmentation with a shorter boundary. By reducing

weight for the boundary term, the shrinking bias can be lowered, which is also equivalent to weighing the data term more heavily. Increasing the weight of the data term is only helpful if the object/background appearance models are relatively accurate. If they are not accurate, putting more weight into them may be unhelpful. If the object is distinctive enough from the background, then reliable appearance models can be obtained with sufficient amount of seeds from the user. A few seeds are usually not enough to have a reliable foreground/background model. However, there are cases where the foreground and background model do not have much difference. In these cases, tuning relative weight parameter λ is not an ideal method to solve the shrinkage problem.

Another way to prevent the shrinking bias with the absence of a reliable appearance model is to have a "ballooning" force for encouraging a larger foreground segment. The "ballooning" bias can be easily incorporated by setting the β in Equation 2.16 to a negative number [35]. The sum of the last term S_{pq} in Equation 2.17 is the number of boundary pixels pairs which is in proportion to the length of the segmentation. A negative β encourages the segmentation to have longer boundary for it can help to reduce more energy.

To choose an appropriate β value, a ratio energy as follow is considered

$$E_{ratio}(f) = f_{weight} + \beta \cdot f_{length} \tag{2.18}$$

f_{weight} is related to the cost of the object boundary, and f_{length} is related to the object area or boundary length. Equation 2.18 is similar to the energy in Equation 2.17 just without the first data term. f_{weight} is the sum of ω_{pq} over all boundary pixel pairs while f_{length} is the sum of all S_{pq} terms with respect to the boundary length. In [35], the optimal β is obtained by searching for a minimum cost labeling (without ballooning) that gives the segmentation with some minimum size. Some segmentation results with star shape prior is listed in Figure 2.7.

2.4 3D Slicer

3D slicer is an open source and multi-platform software package for medical image visualization and analysis. Building with the aim to provide a platform for a variety of applications through a community-development model, slicer possesses initial implementations of many of the core medical image computing requirements like the data model, the GUI application infrastructure, visualization component and basic medical image analysis algorithms [29]. For researchers, Slicer is an end-user application for image analysis. And for software developers, it is an open-source environment which is easy to extend. The interface of 3D Slicer software looks like Figure 2.8. It integrated a lot of common medical image processing algorithms and basic user interaction tools.

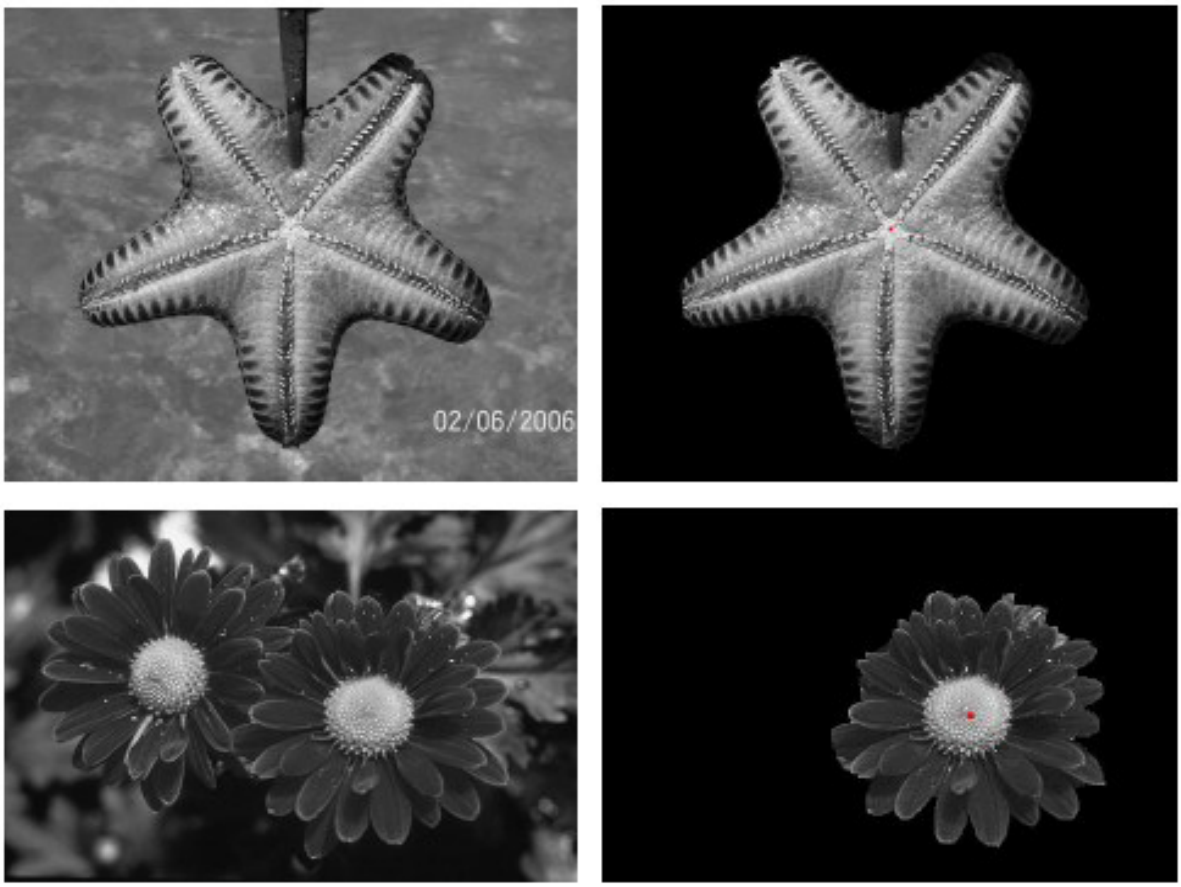


Figure 2.7: Star shape segmentation results. Left: Original images; Right: Segmentation with star shape prior, the star center is marked with the red dot [35].

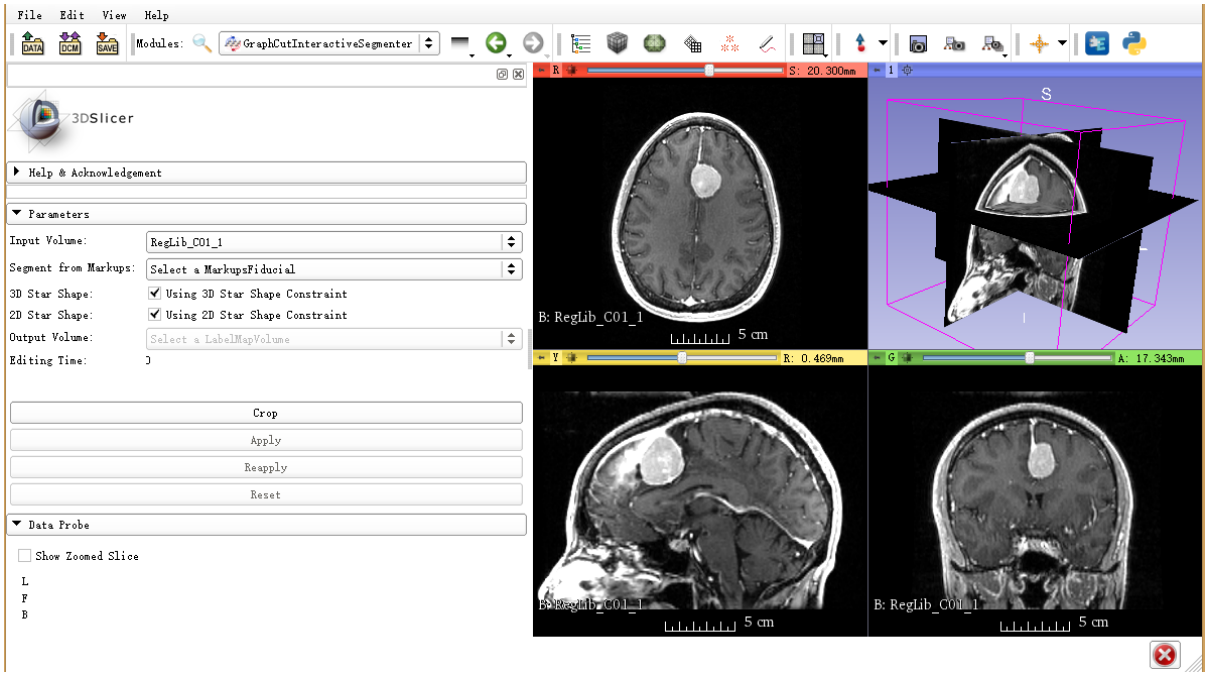


Figure 2.8: 3D Slicer Interface

2.4.1 3D Slicer Architecture

To suit the characteristic of its distributed developer and user environment, one of the most important principles that Slicer follows is modularity. The modular system is achieved by layers of abstractions and componentized functional units, shown in the figure below. This guarantees a mechanism to take full advantage of contributors creativity and expertise as well as Slicers stability[28].

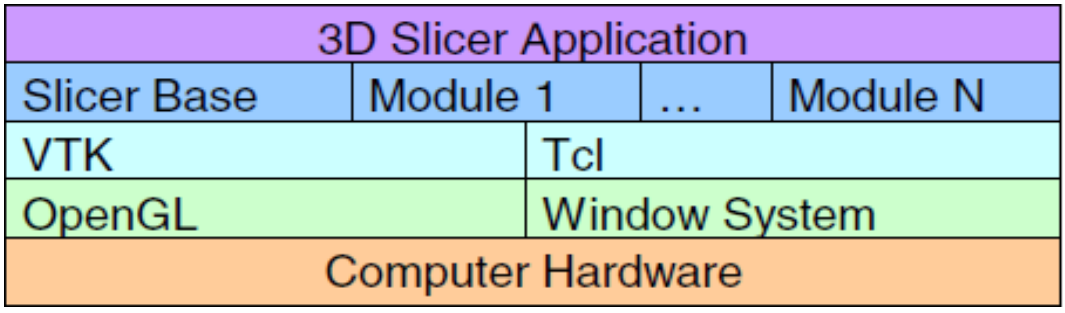


Figure 2.9: 3D Slicer Architecture

Figure 2.9 shows the architecture of 3D Slicer. Slicer is layered mainly on the binding of VTK (Visualization Toolkit) and TCL, the interpreted languages. VTK gives Slicer the

extension of its base functionality with C++ modules for medical imaging. The binding with TCL offers a common library of TCL script code and implements Slicers cross-platform user interface written in the TCL/TK windowing toolkit.

The upper module layer is the place that the component system built on. Modules can be developed individually and do not require access to all other modules to compile or function. Slicer supports three types of modules, command line interface (CLI), loadable modules and scripted modules. CLIs are standalone executables with a limited input/output arguments complexity. Loadable modules are C++ plug-ins that are built against Slicer. They can have custom GUIs since they have full control over the application. It is usually recommended for heaving computation. The scripted modules are written in Python, which are often used for fast prototyping.

2.4.2 3D Slicer as a Tool for Interactive Brain Tumor Segmentation

Because of its easy and interactive user interfaces and powerful plug-in architecture, 3D Slicer also becomes an ideal platform for interactive brain tumor segmentation. The main factors lie in Editor Module and its integrated segmentation algorithms as well as Slicers flexibility to add new segmentation methods.

Editor is an editing tool containing a variety of interactive segmentation effects for volumetric scalar data. Effects cover a wide range of capabilities, from painting, polygonal outlining and thresholding through a variety of connectivity and morphology tools, all the way to novel segmentation algorithms [23]. These comprehensive interactive segmentation effects provide all kinds of basic operations that users need for tumor segment. To show the result of the segment, editor module will generate a Label Map file for each corresponding volumetric scalar data file, which indicates the label of each point. The Label Map file can be edited interactively and also saved and reopened for later use. The tools in Editor module is shown in Figure 2.10. When editing the Label Map, users can even choose the label according to the requirements.

Slicers modular system also enables developers to integrate innovative algorithms into it as extensions. Loadable modules which can both have their own custom GUIs and do heavy computation are quite suitable for integrating new brain tumor segmentation algorithm into this convenient interactive tool. A part of various extensions available in 3D Slicer are listed in Figure 2.11.

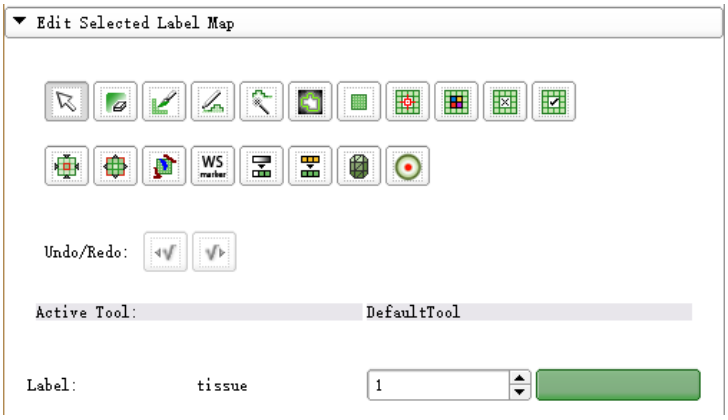


Figure 2.10: Editor Module

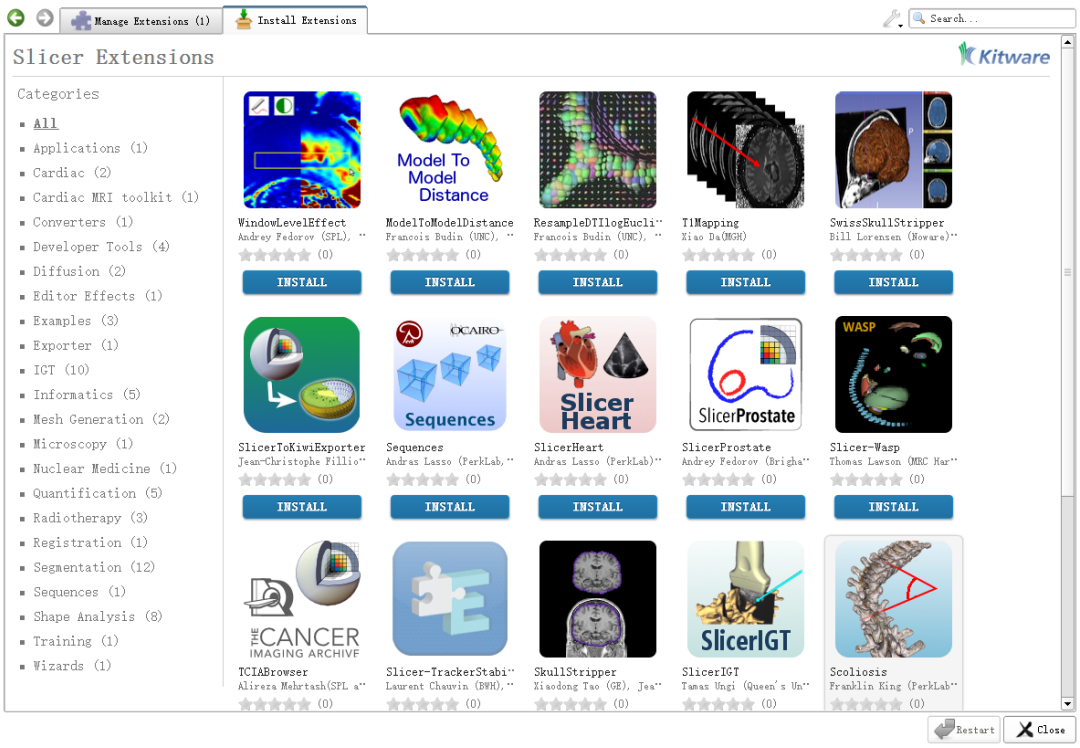


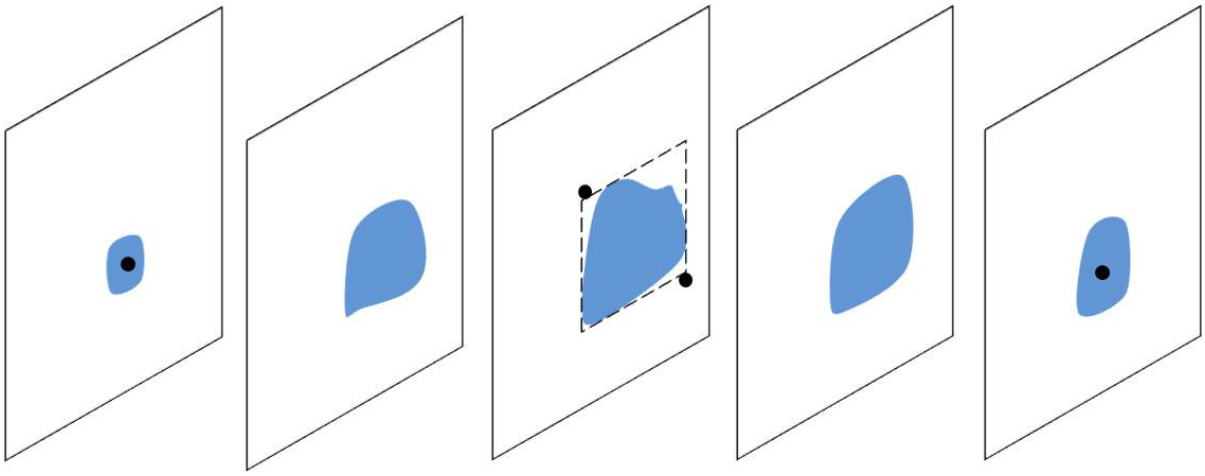
Figure 2.11: Extensions available in 3D Slicer

2.5 Prior Work on Interactive Brain Tumor Segmentation

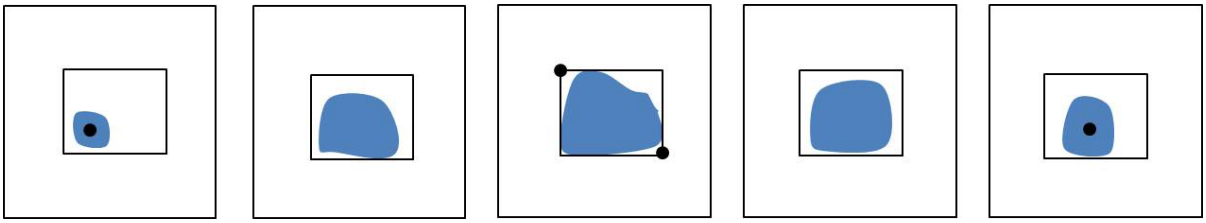
Liqun Liu [25] develops an interactive brain tumor segmentation algorithm that achieves high accuracy. Our proposed system is based on the work in [25]. In this section, we describe the components of [25] that we build on.

2.5.1 Minimal User Input

In [25], the user input for initializing the segmentation is only four clicks. Two clicks on the end slices roughly marking the tumor centers of corresponding slices. And another two clicks on the slice having the largest tumor at the diagonal corners of a rectangle containing the whole tumor area. The four clicks are shown in Figure 2.12.



(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 2.12: An example for user interaction and 3D bounding box obtained from user clicks [25].

The four clicks form a 3D bounding box with nearly the entire tumor in it. The projection of a box to each slice also gives a rectangle. These simple clicks provide a lot of information for

the segmentation algorithm. Apart from the size and location of the tumor, it can also provide key information for initial appearance model and star shape center estimation. The details of making the most of user input in appearance model and star shape will be discussed later.

2.5.2 Data Term

As discussed in previous part, data term measures the extent the labeling fitting into the observed data, usually described by the appearance model. It stands for the individual penalty for assigning a pixel p to be object and background, in the binary segmentation case. The negative log-likelihood of the intensity histogram is often used as the data term energy, shown in Equation 2.19

$$\begin{aligned} E_{data}(f) &= \sum D_p(f_p) \\ D_p(\text{BKG}) &= -log\,\Pr(I_p\,|\,\text{BKG}) \\ D_p(\text{OBJ}) &= -log\,\Pr(I_p\,|\,\text{OBJ}) \end{aligned} \tag{2.19}$$

where P corresponds to pixels $p \in P$ of the image, f is a labeling for all pixels within the image, f_p is the label for a single pixel p, and I_p stands for the image intensity of pixel p. OBJ represents the object label while BKG is the background label in our binary label set $L = \{OBJ, BKG\}$. The log-likelihood function, shown in term of logPr, indicates the probability the pixel p belong to the corresponding label. High probability means the pixels fit quite well for the label, and should have a low energy cost. The minus in front of logPr just allows small penalty for high logPr and large penalty for low probability. Figure 2.13 shows how the data term corresponds to the $t-links$ in a graph. The known appearance model comes from the initial user input. Inadequate inputs may lead to unreliable appearance model. In Liu's [25] approach, a bounding box containing the whole tumor and tumor center for every slices can guarantee a good initial estimation of the appearance model.

Intensity contains the key information of an image. Hence, the intensity histogram is the most common and simple methods to model the region appearance. A normalized intensity histogram reflects the intensity probability distribution of a region within an image. For wide range intensities, a binned histogram is needed for the sake of both computational efficiency and reliability. After binning, the intensity histogram will become smoother. However, attentions should be paid when choosing the number of bins. Too few bins may lead to oversmoothing of the distribution while too many bins can result in a distribution that is too noisy. The two extreme cases are shown in Figure 2.14 With the user input clicks, pixels within the bounding box are taken for foreground histogram and the rest pixels outside the bounding box form the background histogram.

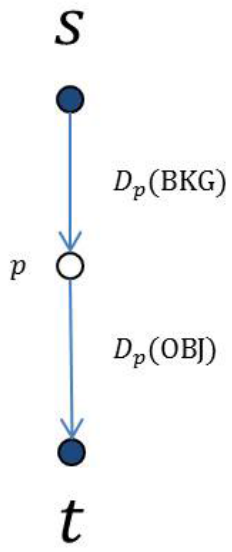
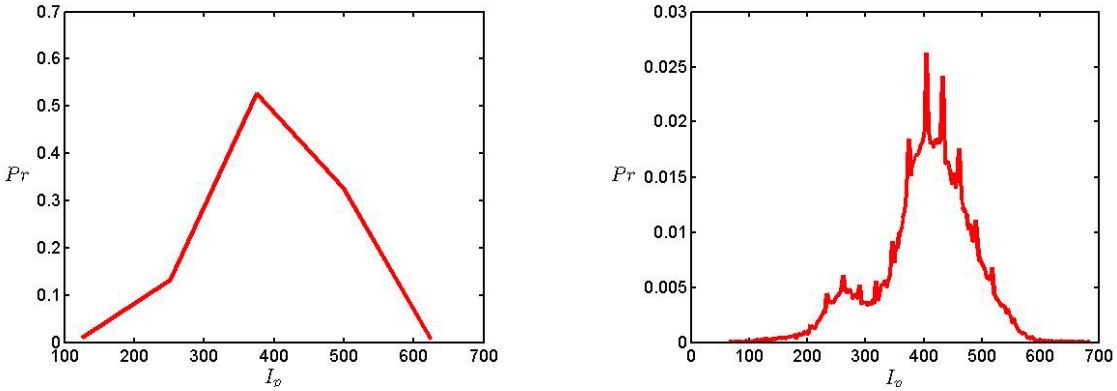


Figure 2.13: $T-links$ weights for pixel p in a graph [25].



(a) Histogram with 5 bins

(b) Histogram with 200 bins

Figure 2.14: Histograms with different number of bins. Pr represents the probability, Ip 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. [25]

The appearance of tumor varies from slice to slice even within one volume. End slices contain only a small tumor region while the middle slices contain a large tumor region. Therefore, a local intensity histogram in each 2D slice is used instead of the global one. The basic idea is still the same. The projection of the bounding box onto each slice is used as the initial rectangle for appearance construction. It is named "local histogram" because each slice uses its local intensities and has its own individual appearance model rather than sharing the global model among all the slices.

2.5.3 Smooth Term

Smooth term in the energy function is a piarwise term encouraging the coherence between neighboring pixels. It can be interpreted as a penalty for the discontinuity i.e. different labels between pixels pairs. Hence, the penalties all lie on the boundary of the segmentation. Since the energy is minimized, this term encourages the segmentation to have a shorter boundary. The smooth term in our energy function follows the term in Equation 2.20.

$$E_{smooth}(f) = \lambda \cdot \sum_{\{p,q\}\in\mathcal{N}} V_{pq}(f_p, f_q) \tag{2.20}$$

where

$$V_{pq}(f_p, f_q) = w_{pq} \cdot \delta(f_p, f_q) \tag{2.21}$$

and

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

The penalty cost can be based on local intensity gradient, Laplacian zero-crossing, gradient direction and other criteria, and can also decrease as a function of distance between pixel p and q [6]. In common cases, the boundary often locates at the pixel pairs where there is great intensity contrast. To align the segmentation boundary with intensity edges, w_{pq} is chosen to be a non-increasing function of the intensity difference $|I_p - I_q|$, where I_p and I_q are intensity of pixels p and q respectively. The definition for w_{pq} is shown in Equation 2.22.

$$w_{pq} = \exp(-\frac{(I_p - I_q)^2}{2\sigma^2}) \cdot \frac{1}{dist(p,q)} \tag{2.22}$$

σ here is set to be the average absolute intensity difference between neighboring pixels in the image, so that the smooth penalty is based on the global standard. When the intensity difference $|I_p - I_q|$ is greater than the average difference, w_{pq} is a small value which allows an edge between them. And in the case where $|I_p - I_q|$ is smaller than the average difference, the cost is expensive to label them differently.

Note that the pixel distance $dist(p,q)$ is also included here. In different neighborhood system, the distances between neighbor pixels are not always the same. The penalty is in inversely proportional with the distance because the closer pixel pairs are more likely to have the same label. In the 3D MRI data, each volume may have different sampling rate along x, y direction and z direction. Instead of using measurement unit in pixel like in normal 2D images, the distances between 3D voxels in MRI volume are measure by the physical one. The incorporation of $dist(p,q)$ can give us more reliable penalty.

Recall that there is a parameter in front of the smooth term in the standard energy function to control the relative weight between data term and smooth term. To take both local information of each slice in the 3D volume and global coherence between slice, different weights for smoothness within 2D slice and smoothness between slices are adopted. As a result, the smooth term becomes

$$E_{smooth}(f) = \lambda_{xy} \cdot \sum_{\{p,q\} \in N_{xy}} V_{pq}(f_p, f_q) + \lambda_z \cdot \sum_{\{p,q\} \in N_z} V_{pq}(f_p, f_q) \tag{2.23}$$

where N_{xy} is the 4-neighborhood system within a slice and N_z is the 2-neighborhood system for pixels on different slices. They are subsets of the 6-neighborhood system N, as shown in Figure 2.15. As mentioned in data term section, the tumor region varies from slice to slice.

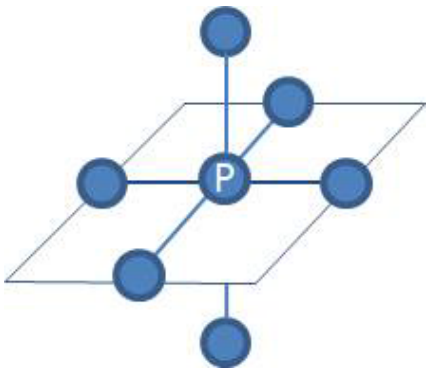


Figure 2.15: 6-neighborhood system. Each voxel has 4 connections to its immediate left, right, top, bottom neighbors within the slice, and two more connections to its closest neighbors in the previous and next slice. [25]

Therefore, the λ_{xy} weight is not the same for different slices. The local intensities of each slice are taken as the clue for computing different λ_{xy} of their own.

2.5.4 Star Shape Constraint

The brain MR images have blurred boundaries between tumor and surrounding tissue, and not always have unique appearance. If there is no constraint of connectivity, it is prone to have seg-

mentation with isolated parts, holes inside or strange shapes. Examples of these unsatisfying segmentation can be found in Figure 2.16. Star shape prior discussed in **Section 2.3** can help to rule out these segmentations and obtain a integral and convex segmentation.

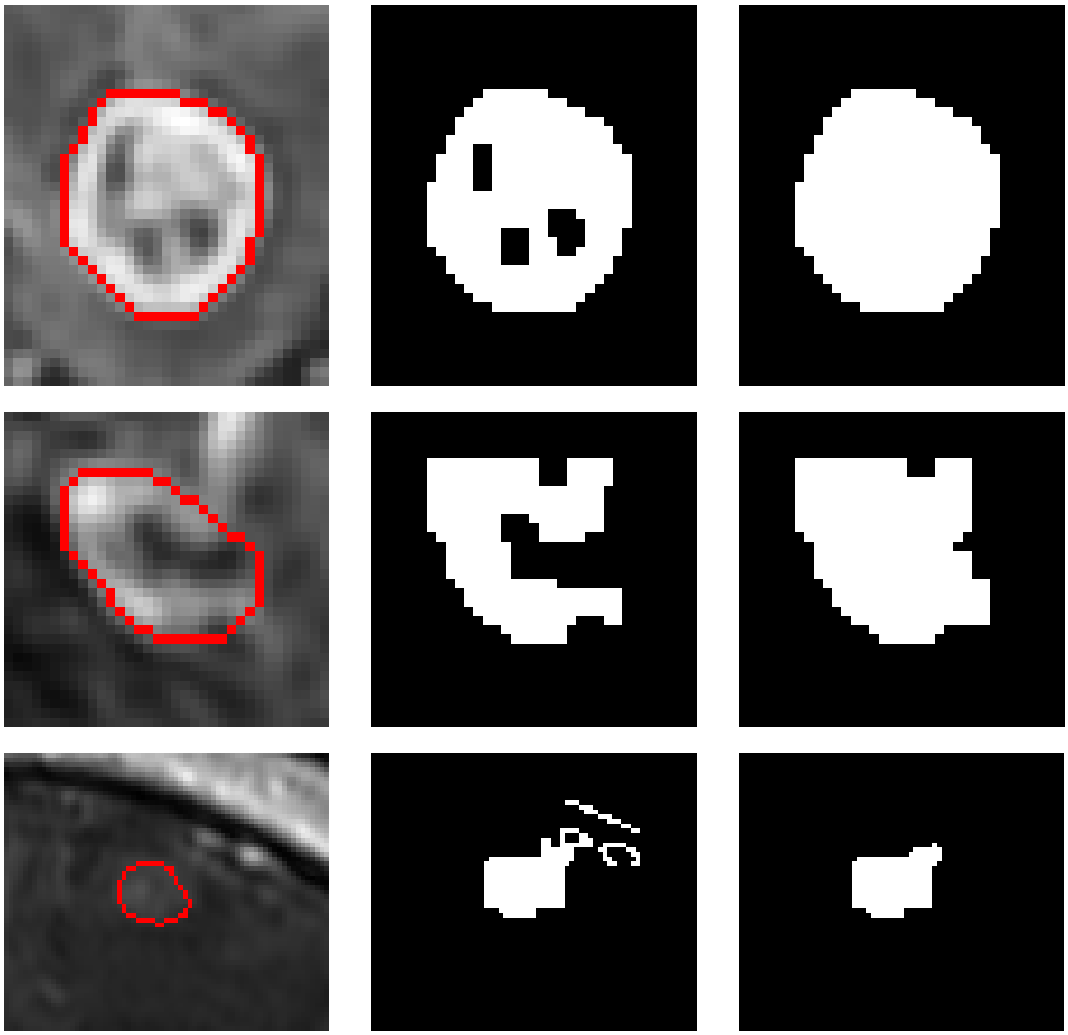


Figure 2.16: Left: a tumor slice. Middle column: (top) segmentation with holes inside. (middle) segmentation with unlikely shape. (bottom): segmentation with isolating regions. Right: segmentation with star shape constraint. Red is the ground truth boundary. White pixel stands for tumor and black stands for background. [25]

In star shape prior, there is a parameter β that can help to get longer segmentation boundary with the absence of a reliable appearance model when not enough user input is provided. In Liu's [25] approach β is set to 0 since the segmentation is initialized with a bounding box and tumor centers on end slices, which are adequate to build a reliable appearance model. Thus,

the Star shape constraint is in the form in Equation 2.24.

$$S_{pq}(f_p, f_q) = \begin{cases} 0 & \text{if} \quad f_p = f_q, \\ \infty & \text{if} \quad f_p = 1 \quad \text{and} \quad f_q = 0, \\ 0 & \text{if} \quad f_p = 0 \quad \text{and} \quad f_q = 1 \end{cases} \tag{2.24}$$

Both 2D and 3D star shape are included in Liu's [25] approach because using only one of them is not enough to guarantee the desired segmentation. Figure 2.17 is an example where the segmentation is not violating 3D star shape but the ring-like segmentation on previous and next slice are not following the 2D star shape constraints. It can be easily imagined that if there is

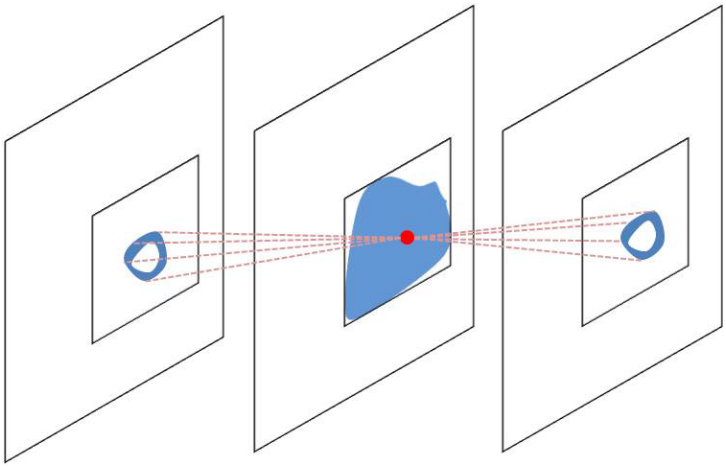


Figure 2.17: An example which is star-shape in 3D but not a star-shape in 2D projections. The red dot is the 3D star shape seed and dashed lines are discretized lines passing through star seed to pixels on previous and next slice.[25]

only 2D star shape and not 3D, the segmentation may be against the convexity with the respect to tumor center in 3D space.

3D star shape center is the center of the rectangle on the slice where user put two clicks. As for 2D star shape, the centers on each slice need to be estimated beforehand. The 3D star center and the two tumor centers on end slices provided also by the user are connected by two lines. And the intersection of the two lines with each slice is the corresponding 2D star center. The process is illustrated in Figure 2.18.

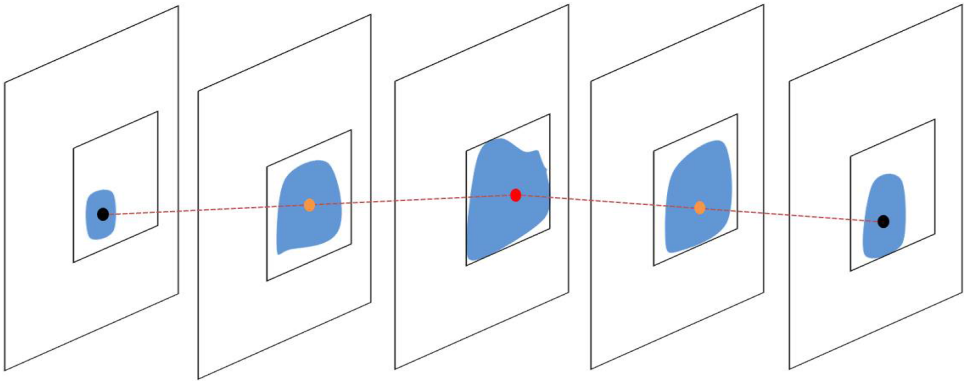


Figure 2.18: 2D star shape centers for the whole volume. Red dot: 3D star shape center. Black dots: user clicks in first and last slices, and also the 2D star shape centers for first and last slice. Yellow dots: 2D star shape centers from the intersection of the lines with the 2D slices. [25]

Chapter 3

Improving Interactive Brain Tumor Segmentation

Our approach builds on the interactive brain segmentation system of Liu [25]. We introduce several features that improve the accuracy of their system, using F-measure for evaluation. The main components of the algorithm in [25] is minimal user interaction, using optimization with graph cuts, and the star shape prior.

We introduce an adaptive search in each slice for the volume ballooning parameter. We also develop better compactness measures that are essential for parameter search algorithm to work properly. Finally, we incorporate the inclusion constraint between the slices that adds more robustness to the segmentation results.

In this section, the improvements in our approach will be broken into parts to have further detailed discussion.

3.1 Overview of Our Approach

Firgure 3.1 illustrates the algorithm flow of our improved interactive brain tumor segmentation. Before applying our algorithm, we need to pre-process the MRI brain data. The raw data contains the information of a whole brain. Dealing with the whole brain data is time-consuming and unnecessary. As discussed in previous section, the bounding box provided by the user restricted the tumor within the box. Thus, based on the bounding box, a larger box can be generated to mark the ROI (region of interest). And all other information outside the ROI box and be set aside. In general, the input data for our approach is the cropped ROI from the whole MRI brain data.

We begin our algorithm with the slice by slice 2D segmentation to utilize the local infor-

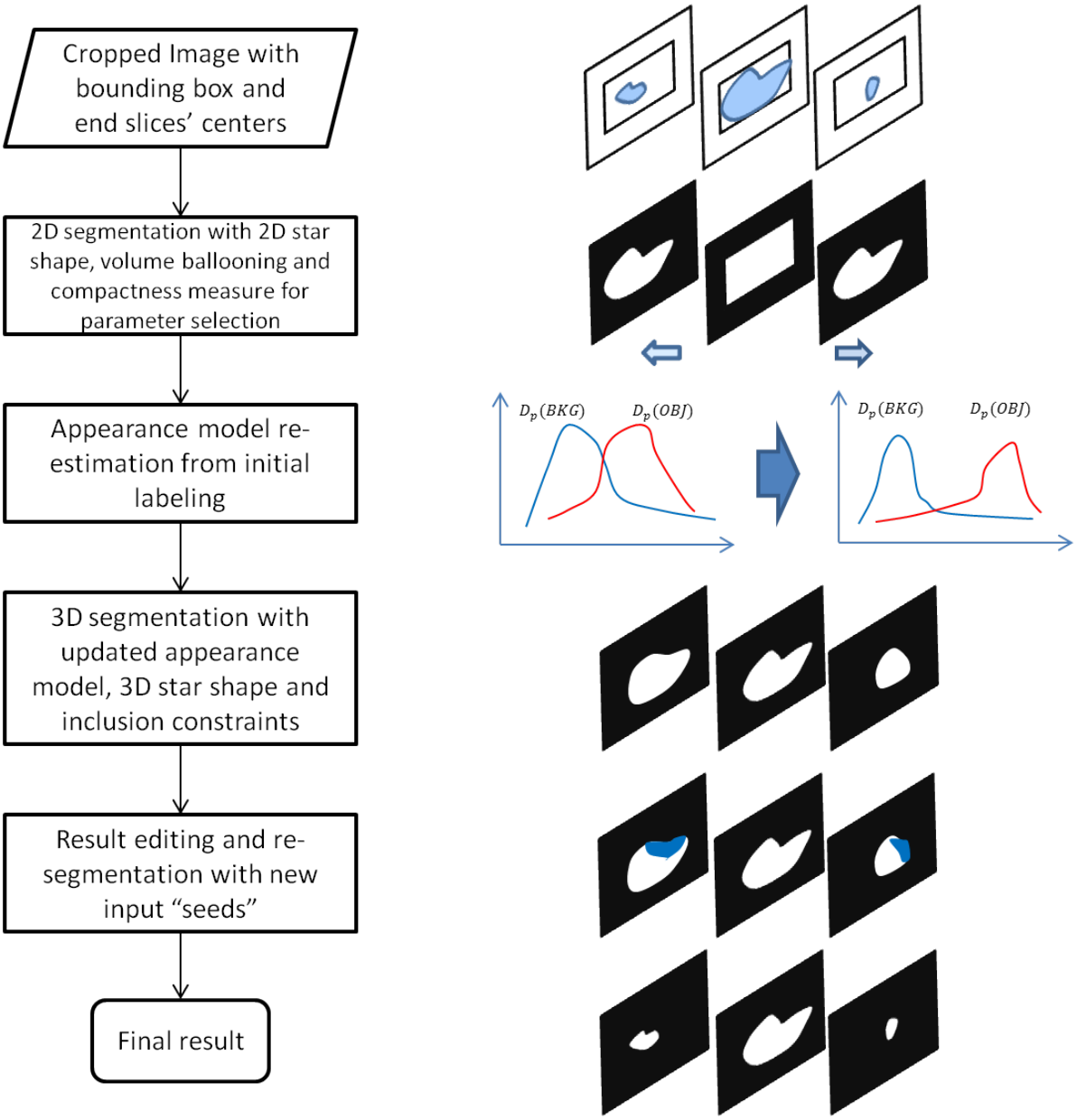


Figure 3.1: Algorithm flow of interactive brain tumor segmentation.

mation of each slice first because the variance of the brain tumor image usually makes the global information usually not quite enough for an accurate segmentation. The middle slice here refers to the slice where user put two clicks that form the rectangle. This slice is expected to have the largest tumor area and has the user markings, we expect the segmentation for this slice to be highly reliable. The rectangle plays the same role as the initialization rectangle in grab cut [32], only we take pixels outside the box in a band of size 1/4 of the user provided box. After segmenting the middle slice, the result is used as the foreground mask for the following slice, i.e. we build the foreground histogram for next slice based on the pixels within the mask. As for background histogram, it still consists of the pixels outside the rectangle. The 2D segmentation propagates in the direction from the middle to the sides. The 2D star shape prior and volume ballooning are integrated into the 2D segmentation. During the 2D segmentation, there is a volume ballooning parameter searching process where the parameter selection is based on the compactness measure of the 2D segmentation.

Using the same idea in grab cut [32], we do not assume a fixed foreground and background model. After slice by slice segmentation, we can re-estimate the appearance model utilizing the result labeling for each slice. With the updated appearance model, 3D segmentation can be done within the global scale. In 3D segmentation, the 3D star shape prior and the inclusion constraints between slices are added. For energy minimization, we use the graph cut [5] framework. After the 3D segmentation, the user can evaluate the result can edit the labeling with additional foreground and background seeds to correct another segmentation. Both 2D and 3D segmentation result get reused. Therefore, the graph does not need to be rebuilt from scratch in our algorithm, which greatly improves the efficiency.

3.2 Volume Ballooning

Even though the bounding box gives us a lot information about the foreground and background to construct an appearance model, in some cases, the foreground and background models constructed from the initial rectangle are very similar. In such case, we may want to change parameters of the energy function until we get a segmentation that is large enough, or that satisfies some other criterion. Changing the weight of the appearance model vs. the weight of the smoothness term may be insufficient to get the desired, say larger, results. An example can be found in Figure 3.2. In this image, suppose the goal is to segment the larger square that is white inside and black on the border from the noisy background. Suppose the initial box provided by the user is as the red square in Figure 3.2b. The foreground and background appearance model built with this initialization have an approximately the same mixture of black and white pixels, so they are virtually indistinguishable and highly unreliable. Even with a very small

(a) Orginal image (b) Result without ballooning (c) Result with ballooning

Figure 3.2: Comparison of segmentation with and without volume ballooning. (a) Original image; (b) Segmentation without volume ballooning. The red rectangle is the initial user input and the green line marks the segmentation boundary; (c) Segmentation with volume ballooning. It has the same user input rectangle but with volumic bias, it gets larger segmentation.

cost for the boundary we get segmentation outlined in green in Figure 3.2b. Furthermore, there is no setting of the smoothness parameter λ that gives us the desired segmentation. The method in [25] performs a search over parameter λ until a "good" segmentation is found. One of the criteria for good segmentation is that the foreground object is large enough. However, as the example in Figure 3.2 shows, in cases of large foreground/background overlap, searching over parameter λ may never return a segmentation that is large enough. In fact, we did observe this in several faiure cases as shown in Figure 3.3.

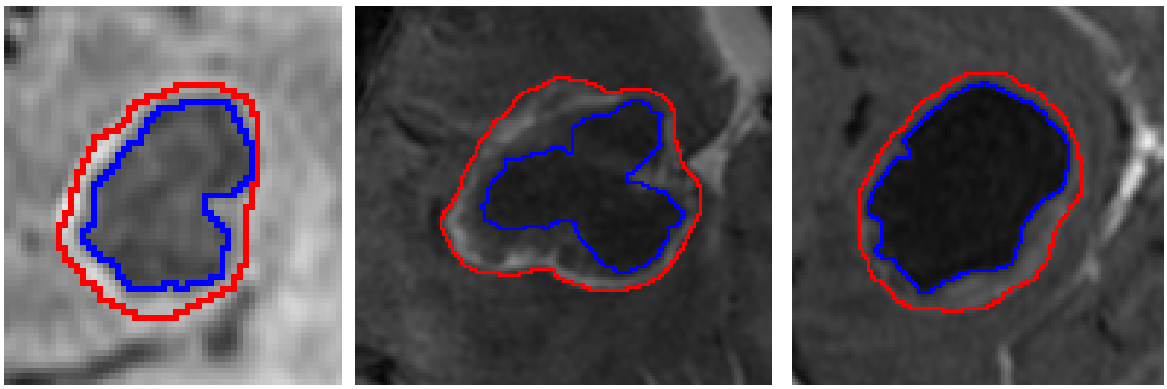


Figure 3.3: Failure cases of getting large segmentation with parameter λ search. The red countour is the ground truth and the blue contour is our segmentation.

Computing foreground/background overlap statistics from the ground truth in our dataset, we find that the significant overlapping of foreground and background intensity histogram is not rare case and it may contribute to segmentation errors. This motivates us to add volume

ballooning into our approach to get bigger object labeling.

There is non-uniform ballooning and uniform ballooning. In non-uniform ballooning, for each pixel p we add its own bias to be labeled as the foreground. For example, in [1], non-uniform ballooning force decreases at the rate of $1/r$ where r is the distance from pixel to the circle center c. Uniform ballooning, on the other hand, gives all pixels an equal ballooning force. In Liu's work, the non-uniform ballooning has been used. The radius r for the middle slice is determined by the length of the bounding box. And the radius decrease linearly from the center to the end slices.

Uniform ballooning [8] is adopted in our method. It is implemented by adding a bonus to each pixels in the image if it is assigned the foreground label. For each pixel p, we add an additional penalty for it to be labeled as a background, namely for each pixel p, we add a term $B_p(0) = \beta$. That is there is an additional penalty for pixel to be assigned to the background label 0. Then our energy function based on Liu's approach [25] with volume ballooning becomes

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

Since B_p is a unary term, it is easy to optimize with graph cuts. The larger is the parameter β, the larger is the ballooning force.

The ballooning force for each slice may vary. For example, slices far away from the middle may need a larger ballooning because of weaker edges and appearance while middle slices may need only a little ballooning. To suit the different needs for different slices, we choose the ballooning parameter according to the local information of each slice. This means that we may have a different setting of ballooning parameter β for different slices. The searching process for ballooning parameter β is as follow. First, an appropriate parameter search range is set according to the area criteria. The initial search range is from $\beta_{min} = 0$, which means no ballooning, to β_{max}, which is a large enough value to balloon the segmentation to the whole bounding box. β_{min} and β_{max} are adjusted within the initial range according to the segmentation area. If the segmentation area with the ballooning β_{min} is smaller than the minimum area, we increase β_{min}. Similarly, if the segmentation area with the ballooning β_{max} is larger than the maximum area, we decrease β_{max}. β_{min} and β_{max} are set until the area criteria is satisfied. Then, the best ballooning force is searched within the range based on the compactness measure of the segmentation. We sample ballooning parameter β between β_{min} and β_{max}. Each β corresponds to a segmentation. All segmentations are evaluated by compactness and we choose the best β with the best compactness measure segmentation.

Figure 3.2c shows the result we obtain with volume ballooning on the example where smoothness parameter search fails to produce a desirable segmentation. For a range of β, we

can obtain the desired segmentation as illustrated in Figure 3.2c. Figure 3.4 shows the comparison of a slice labeling with and without the volume ballooning for some example segmentations on our dataset.

3.3 Compactness Measure

When searching over a parameter range, we need a "goodness of segmentation measure" to choose the parameter giving most desirable segmentation. One of the criteria is size. In [25], $size_{max} = 1.3 \cdot size_{mask}$ and $size_{min} = size_{mask}/3$ where for middle slice $size_{mask}$ is the size of bounding rectangle and for other slices $size_{mask}$ is the size of the previous slice's segmentation. Most tumors tend to be quite regular or compact in shape. Thus another criterion is how regular or "compact" the foreground segment is. Here we use the word "compact" informally, to mean that the shape is not too irregular, i.e. the ratio of its perimeter to area is small. Therefore we include a measure of compactness to evaluate how good a segment is, in addition to a segment size. Here, we use it as the standard to measure the shape of our segmentation.

Even though Star Shape constraint has already been included in our approach, it is only a shape constraint with respect to the center point and not necessarily give us a regular or compact shape. In fact, the result can be quite far from convex. When choosing the appropriate parameters for energy function, like volume ballooning parameter, we evaluate the segmentation and select the one with the best compactness measure. Because the compactness measure is only used for assisting us to choose the appropriate parameter, it is only used in 2D slice by slice segmentation process. 3D segmentation compactness measure is not in our concern.

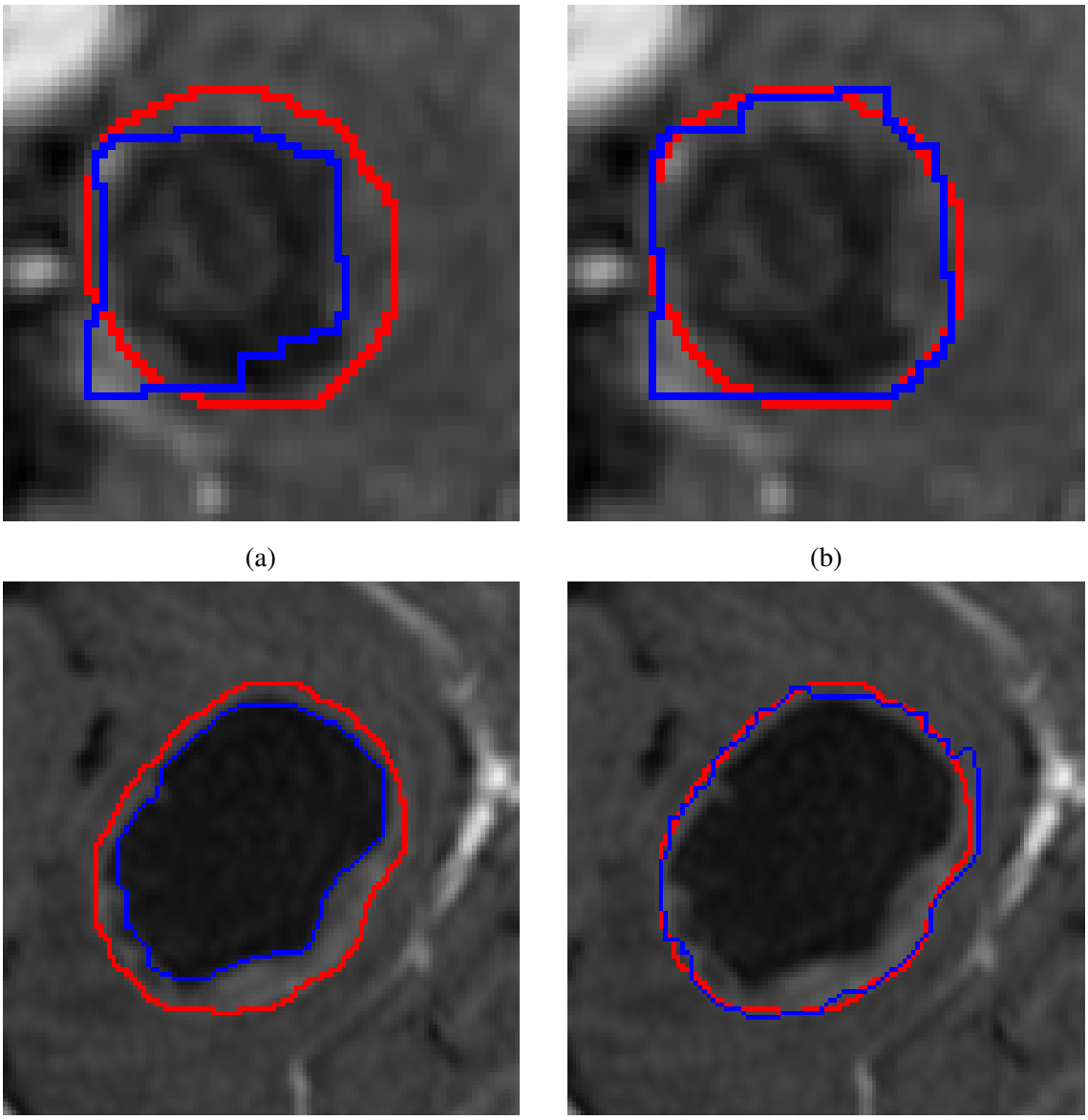
Compactness is one of the widely-used region-based shape descriptors. The classical shape compactness is usually associated with the ratio, often called $P2A$

$$C = P^2/A \tag{3.2}$$

where C is the shape compactness, A is the shape area and P is the shape perimeter. However, when applying the $P2A$ to discrete digital images, problem arises because of the evaluation of the length of region border [26].

Haralick [17] proposed a inner distance approach in 1974 for digital compactness based on circularity. In his method, a shape centroid is calculated and also all the Euclidean distances from the centroid to each boundary pixel. And finally, the ratio of the media and the standard deviation of the distance set is used as the compactness.

In [25], they used a certain compactness measure similar to [17]. We name it circularity compactness in this thesis. Instead of calculating the shape centroid, the star shape centers of each slices are regarded as the approximation of tumor centers. Then the distance from



(a) (b)

(c) (d)

Figure 3.4: Comparison of segmentation with and without volume ballooning. The red contour is the ground truth and the blue contour is the segmentation (a) Segmentation of a slice without ballooning bias; (b) Segmentation with volume ballooning. This is the same slice with Figure (a) but only segmented with ballooning bias and it gets bigger segmentation; (c) Another slice segmentation without ballooning. The segmentation prefers the inner circle where have high intensity contrast; (d) Segmentation with volume ballooning. With ballooning bias, the segmentation is more accurate.

each boundary pixel to the tumor center is calculated. Finally, the standard deviation of all these distances is taken as the compactness measure of the segmentation. Figure 3.8a shows one example distance from boundary pixel P to tumor center C. We found experimentally that this measure does not always produce the desired results. Some failure examples using compactness measure in [25] are shown in Figure 3.5. Therefore we experimented with several

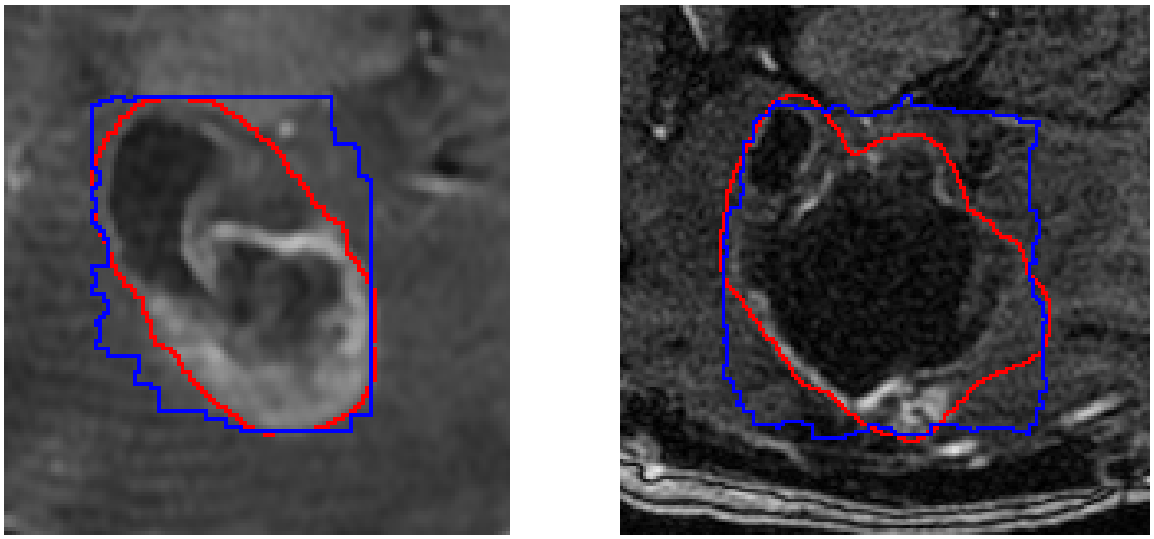


Figure 3.5: Failure examples using circularity compactness. The red contour is the ground truth and the blue contour is our segmentation.

other measures of compactness, described next.

3.3.1 Ellipse fitting

Since the compactness measurement in [25] is based on circularity, it will prefer round segmentations. From the ground truth, we observe that most tumors are much close to an ellipse, rather than circle, in their shape. Thus we designed a measure of compactness based on the deviation from an ellipse of best fit.

According to the property of an ellipse, the sum of distances of a boundary point to the two focal points of the ellipse is a constant, as shown in Figure 3.6. Following the same idea in Haralick's [17] method, the shift from circularity to ellipsity can be easily achieved by changing the distance set. Unlike a circle with only one center, an ellipse has two focal points which cannot be reliably estimated from the user input. So there should be another step for calculating the focal points.

Our approach for ellipsity compactness is simple. First, we use ellipse fitting to find a best-fit ellipse for the segmentation we need to evaluate. With the ellipse shape, the two focal

points can be easily gained through geometric calculation. Then, as shown in Figure 3.8b, we calculate the distance sum, $PF_1 + PF_2$, of each segmentation boundary point P to the two foci, F_1 and F_2, of the approximated ellipse. All the sums consist of a distance set. And the standard derivation of the distance set is the compactness we use for shape evaluation. The standard derivation reflects the likelihood the segmentation to the ellipse shape. The smaller the value is, the more similar the segmentation to the ellipse. Hence, we will choose the segmentation with the smallest ellipsity compactness measure.

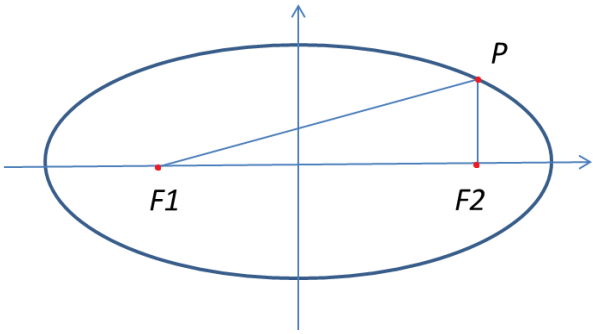
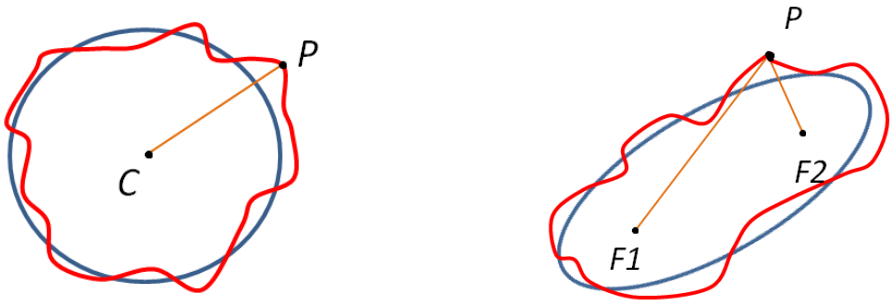


Figure 3.6: An ellipse shape. The point P is a point on the border; points F_1 and F_2 are two foci of the ellipse. The sum of PF_1 and PF_2 is a constant.

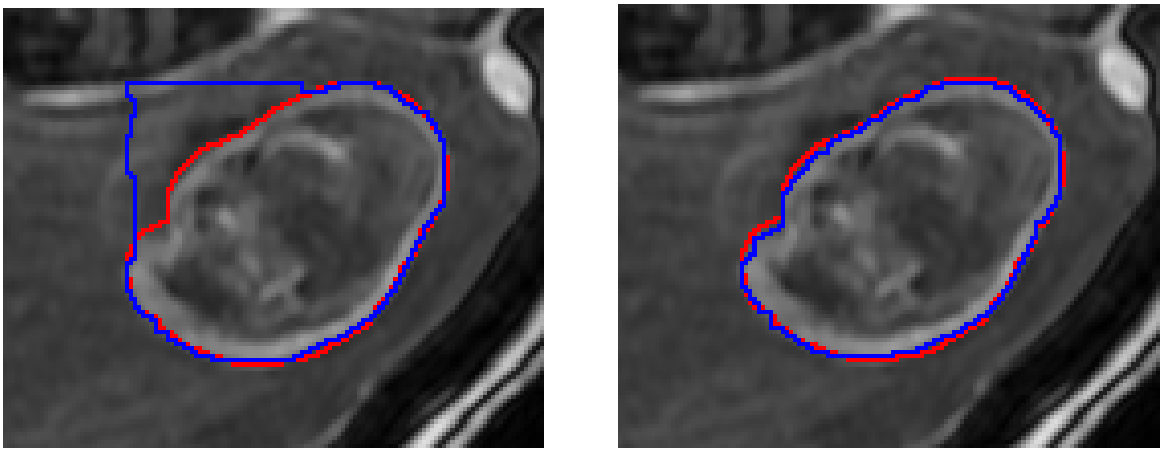


(a) Circularity compactness

(b) Ellipsity compactness

Figure 3.7: Comparison of circularity compactness and ellipse compactness. (a) Distance in circularity compactness. The red contour is the segmentation, the black point is a boundary point on segmentation, the blue point is the centroid. The distance for circularity compactness represented by the orange line connecting the black and blue point; (b) Distance in ellipsity compactness. The red contour and the black point are the same with circularity compactness, two blue points are the foci of the best fitting ellipse. The distance for ellipsity compactness is the sum of the two orange lines connecting the black and blue points.

We tried both circularity and ellipsity compactness in our experiment. Figure 3.8 shows the segmentation of a tumor slice using circularity and ellipsity compactness respectively. Circularity compactness chooses a more round shape while ellipsity compactness chooses the shape that is more like an ellipse.



(a) Segmentation with circularity compactness (b) Segmentation with ellipsity compactness

Figure 3.8: Segmentation comparison of a slice using circularity compactness and ellipse compactness. The red contour is the ground truth and the blue contour is the segmentation of our method. (a) Segmentation using circularity compactness. The result contour is more like a circle shape; (b) Segmentation using ellipsity compactness. The result contour is more like an ellipse shape.

3.3.2 Convexity Deviation

Convexity is another method to evaluate how regular the segmented shape is. Although we already have star shape prior, it only imposes convexity constraints along the lines passing through the star center and there is no constraint with the respect to other points. Therefore, star shape actually allows for non-convex segmentations, such as the shape of a real star object. Our desired segmentation for the tumor actually should be convex. Ellipse fitting can help us to select convex segmentation to some extent because it favors smooth segmentation closer to the ellipse. But it can still pick out non-convex segmentation in many cases, like Figure 3.11a and Figure 3.11c.

A more straight-forward way to avoid choosing non-convex segmentation is to integrate the convexity directly into the compactness measure. Gorelick et. al. proposed a convexity shape prior in [16]. In continuous case, segmentation S is convex if and only if for any points $p, r \in S$ there is no point q on the line between them s.t. $q \notin S$. It indicates that there should

be no 1-0-1 triplet configurations along any straight lines of a convex labeling, where 1 stands for the foreground label and 0 stands for the background label in the binary segmentation. In discrete case, the convexity constraints are approximated as follows. Let $i \in \{1, ..., m\}$ enumerate discrete line orientations, as shown in Figure 3.9a, and let L_i be the set of all discrete lines $l \subset \Omega$, where Ω is the set of all image pixels, of given orientation d_i such that

$$l = \{p_0, p_1, ..., p_n | p_t = p_0 + t \cdot d_i, t \in Z^+, p_t \subset \Omega\} \tag{3.3}$$

Figure3.9b illustrates the discrete line set L_i for one particular orientation d_i. The convexity

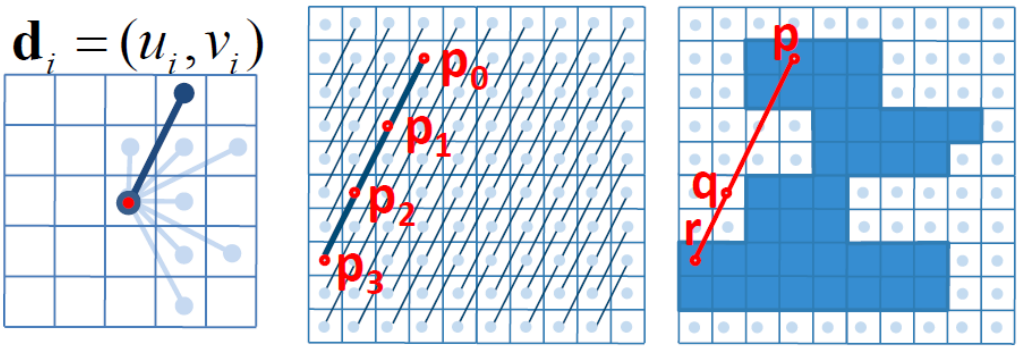


Figure 3.9: Left: Example of discretized orientations given by a 5 by 5 stencil. d_i is one of the orientations. Middle: Set L_i of all discrete lines on image grid that are parallel to d_i. Right: Example of a triple clique (p, q, r) that violates convexity constraint. [16]

evaluation of the segmentation should go through all triplets over all orientations and all lines. The total number of triplet violating the convexity can be written like the Equation 3.4.

$$N_{convexity}(f) = \sum_{l \in L_i} \sum_{(p,q,r) \in l} \Phi(f_p, f_q, f_r) \tag{3.4}$$

where

$$\Phi(f_p, f_q, f_r) = \begin{cases} 1 & \text{if} \quad (f_p, f_q, f_r) = (1, 0, 1), \\ 0 & \text{if} \qquad \text{otherwise} \end{cases} \tag{3.5}$$

f_p, f_q, f_r here represent the corresponding label of pixel p, q, r respectively.

It seems quite expensive to count the number, seeing from the definition. In [16], an efficient evaluation method using dynamic programming is introduced. Consider pixels $p_s, p_t, p_v \in l$. If $s < t$, pixel p_s precedes pixel p_t; if $v > t$, pixel p_v succeeds pixel p_t. Denote $C^-(t)$ the

number of pixels p_s preceding pixel p_t such that $f_s = 1$, and $C^+(t)$ the number of pixels p_v preceding pixel p_t such that $f_v = 1$

$$\begin{aligned} C^-(t) &= \sum_{s<t} f_s \\ C^+(t) &= \sum_{v>t} f_v \end{aligned} \tag{3.6}$$

Let us first consider one pixel $p_t \in l$ with $f_t = 0$. The total number of combination of triplet (p_s, p_t, p_v), $s < t < v$, with configuration $(1, 0, 1)$ can be given by $C^-(t) \cdot C^+(t)$. Then, the equation for the total convexity violation triplet number can be written like

$$N_{convexity}(f) = \sum_{l \in L_i} \sum_{t \in l} C^-(t) \cdot C^+(t) \tag{3.7}$$

where L_i is the set of lines of all orientations, l is one line among the line set and t is one pixel on line l.

The calculation of $C^-(t)$ and $C^+(t)$ for all pixels on one line can be done in one passing through dynamic programming. An example of how $C^-(t)$ and $C^+(t)$ is computed on one line is shown in Figure 3.9. For the first pixel p_0, we count the number of pixels having label 1

Index	0	1	2	3	4	5	6	7
f_l	1	1	0	1	0	1	1	0
C^-	0	1	2	2	3	3	4	5
C^+	4	3	3	2	2	1	0	0
			6		6			0

Figure 3.10: Example of $C^-(t)$ and $C^+(t)$ computation on one line. The first row is the labels of each pixel on line l. The second row and the third row are the number of pixels with label 1 precedes and succeeds pixel p_t. The last row is the number of triplet violating convexity with the respect to pixel p_t with $f_p = 0$.

before and after it, i.e. $C^-(0)$ and $C^+(0)$. For the next pixel p_t, $C^-(t)$ and $C^+(t)$ can be get from the result of previous pixel, $C^-(t) = C^-(t-1) + f_{t-1}$ and $C^+(t) = C^+(t-1) - f_{t-1}$. For lines of a particular orientation d_i, they can cover almost all pixels in the image, as shown in Figure 3.9b. Therefore, the total computational time of $N_{convexity}$ is $O(mN)$, where $N = |\Omega|$ is the total number of pixels in the image and m is the number of orientations.

Since the number of triplet violating the convexity in a segmentation indicates the overall convexity of the result. We can use the number as the approximation of the compactness measure. For each segmentation, there is a corresponding convexity violation number. The smaller the number we have, the more convex the segmentation we get. In parameter searching process, the parameter giving the segmentation with the smallest convexity evaluation number is then chosen.

Figure 3.11 shows the comparison of the segmentation result using ellipse fitting and convexity. Ellipse fitting is more likely to let the segmentation have an ellipse shape while convexity prior can restrict the segmentation to have more convex result. In the examples of Figure 3.11, convexity fixes the cases where ellipse fitting fails to have an accurate segmentation.

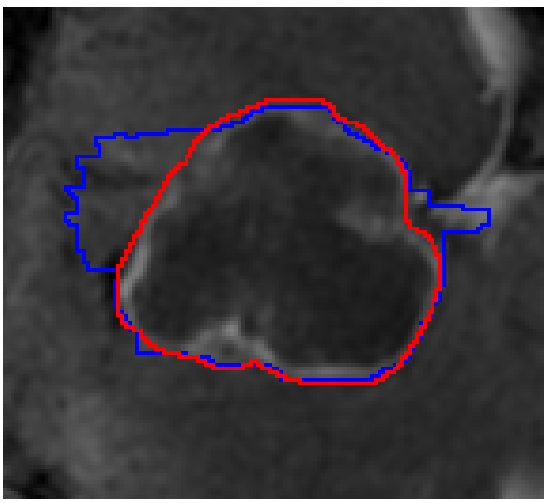
3.4 Inclusion Constraint

So far, we discussed several improvements for the slice by slice 2D segmentation. Although there are 3D star shape constraint and 3D graph cut, the constraint between slices are not strong enough. We notice that the brain tumor area on 2D projection images decreases from middle slice to side slices. This motivates us to add an inclusion constraint between slices so that the segmentation for the side slices would not be much larger than its previous middle slice. More precisely, a segmentation in a slice, when projected on the ”previous” slice, should be mostly included in the previous slice. Here the previous slice is computed as follows. The middle slice has no previous slice. Any slice to the left of the middle, has as its previous slice the slice immediately to the right. Any slice to the right of the middle, has as its previous slice the slice immediately to the left. We verified experimentally on our ground truth that inclusion constraint is satisfied for the majority of pixels. If we now allow inclusion with the margin, that is, if the previous slice is enlarged, then the inclusion holds, then all of the pixels in our data satisfy the inclusion constraint with the margin of 1.7, on average. The inclusion constraints consist of two parts mask inclusion and pairwise constraint inclusion.

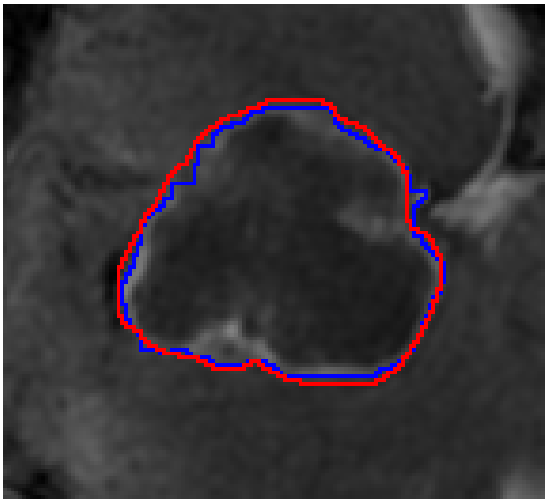
3.4.1 Inclusion using Mask

This case applies to the first stage of our algorithm where optimal smoothness and ballooning parameters are determined. In this case, we perform segmentation on each 2D slice separately, starting from the middle slice, and going towards the side slices. A straightforward method of adding inclusion between slices is using the result of previous slice.

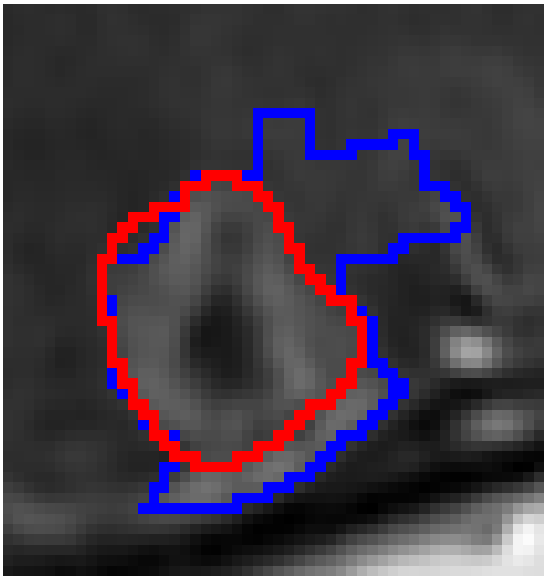
The labeling of previous slice can be used as a constraint mask for current slice. Pixels outside the mask can be encouraged ”softly” to be background while there is no change for



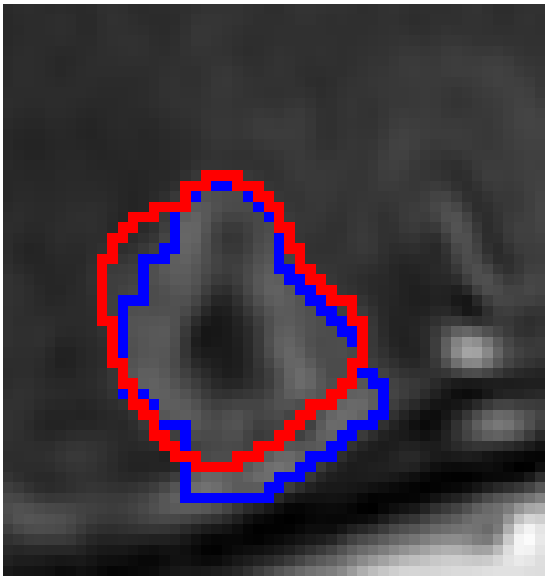
(a) Segmentation with ellipse fitting



(b) Segmentation with convexity



(c) Segmentation with ellipse fitting



(d) Segmentation with convexity

Figure 3.11: Segmentation comparison of two slices using ellipse fitting and convexity. The red contour is the ground truth and the blue contour is the segmenation of our method. The left images are result with ellipse fitting and the right images are result using convexity.

pixels within the mask. The mask constraint is illustrated in Figure 3.12. This soft constraint

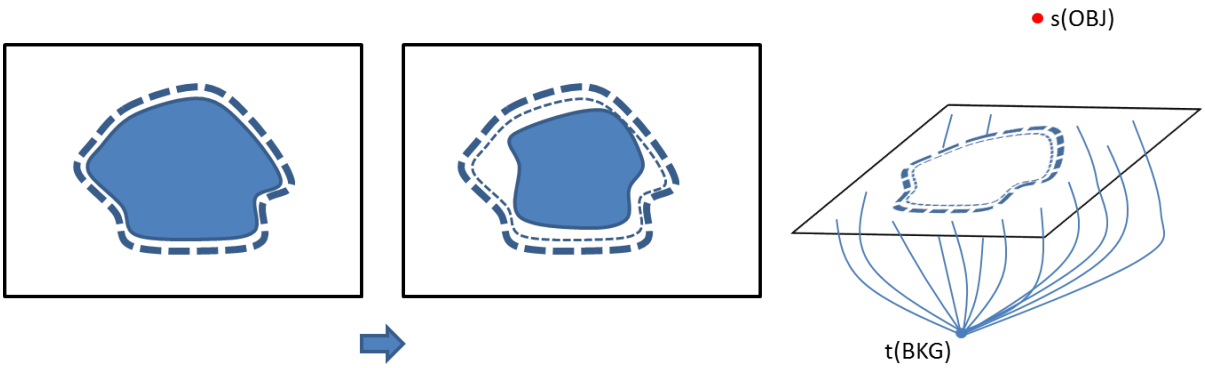


Figure 3.12: Process of inclusion using mask. Left: the previous slice. The blue region is the segmentation. The thick blue dash boundary has the same shape with the segmentation boundary but with a margin. Middle: the current slice for which to add the inclusion constraint. The blue region is the tumor area of the current slice, the thin dash line is the segmentation boundary of the previous slice and the thick dash line is the segmentation boundary with a margin. Right: Mask inclusion in graph. The mask for inclusion can be either the thin dash boundary or the thick one. The pixels outside the mask are imposed a soft constraint to the background.

can be implemented with a unary term in the energy in graph cut algorithm. The unary term with mask inclusion can be written in the form below in Table 3.1 where i refers to the previous

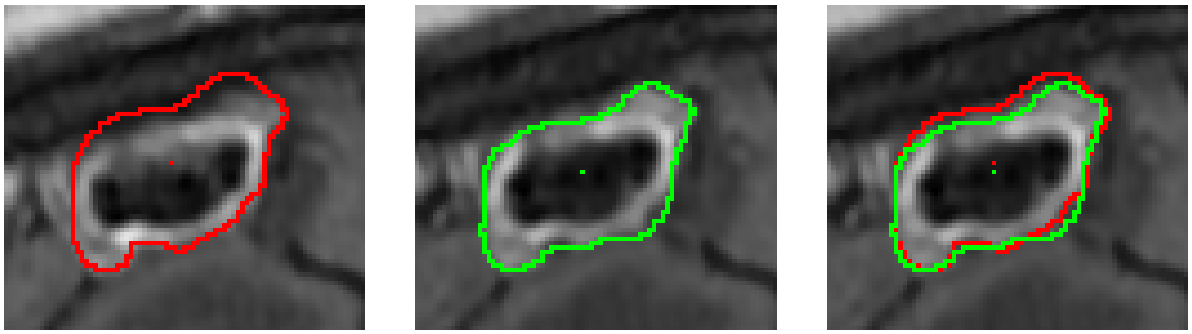
$\mathbf{f}_p^i$	$\mathbf{f}_p^j$	I_p
0	0	K_1
0	1	0
1	0	0
1	1	0

Table 3.1: Weights for data term with mask inclusion.

slice where the mask comes from, j refers to the current slice that needs to be segmented, p is one pixel in the 2D image, f_p^i corresponds to the label on the mask and f_p^j corresponds to the label of current slice. K here is the soft constraint so that the pixels outside the mask are more likely to be background.

Quite frequently, inclusion is not strict. This is partially due to the fact that the tumor can have an arbitrary position inside the 3D brain volume and may have wrong alignment for strict inclusion to hold. In many cases, the region of the current slice may be slightly outside the

previous one. An example is shown in Figure 3.13. Therefore, we can allow a margin along



(a) (b) (c)

Figure 3.13: Example of non-strict inclusion. (a): Boundary and the tumor center of a 2D slice; (b): Boundary and the tumor center of the following slice of (a); (c): Overlap of two boundaries. Putting the boundary of (a) onto slice (b), (b) is not strictly included in (a).

the boundary of the mask so that the enlarged mask can fully include the tumor region on the current slice, as shown in Figure 3.12. The enlargement of the mask can be easily implemented through dilation. The mask inclusion with margin is implemented in exactly the same way as without mask enlargement, the only difference is that the mask gets dilated. We use the enlarged mask for the inclusion so that all pixels outside the large mask have bonus to be labeled as background.

3.4.2 Inclusion using Pairwise Constraints

This case is used during the second stage of the algorithm. Now that the smoothness and ballooning parameters have been estimated, all 2D slices are segmented simultaneously, so the inclusion has to be handled differently. Here we can use pairwise constraint so that the pixel pair will be given a penalty if assigned labels against the inclusion. Delong and Boykov [10] talked about three geometric interactions for regions in their multi-region segmentation, containment, exclusion and attraction. The containment interaction between regions in [10] is similar to our inclusion constraint. Borrowing the idea from the containment geometric interaction, the simple pairwise inclusion for our approach can be defined in Table 3.2. Denote where i is the previous slice, j is the current slice which should be added inclusion to, p and q correspond to a pixel on the 2D image, ω_{pq}^{ij} is the weights between the pixel p and q in slice i and slice j. K_2 gives penalty for the labeling where pixel p is assigned background in slice i and pixel q is assigned foreground in slice j. In the simple case without margin, p and q can be equal so that it refers to the constraint between the pixels with the same coordinates in the

$\mathbf{f}_p^i$	$\mathbf{f}_q^j$	ω_{pq}^{ij}
0	0	0
0	1	K_2
1	0	0
1	1	0

Table 3.2: Weights for pairwise inclusion term.

two slices and the weights here indicate that foreground labeling of slice i should include the one in slice j.

In the more complicated case, the margin should be taken into consideration. Unlike the mask inclusion case where we already have the segmentation in the previous slice, here, we do not know the segmentation boundary yet since segmentation is performed in all 2D slices simultaneously, i.e. it is a 3D segmentation. Using the idea for having margin in region containment interaction in [10], we impose the constraints between one pixel and its nearby pixel with some margin instead of adding constraints between the same pixel on two slices in the no margin case. The margin is set to grow along the star shape direction so that it is easy to

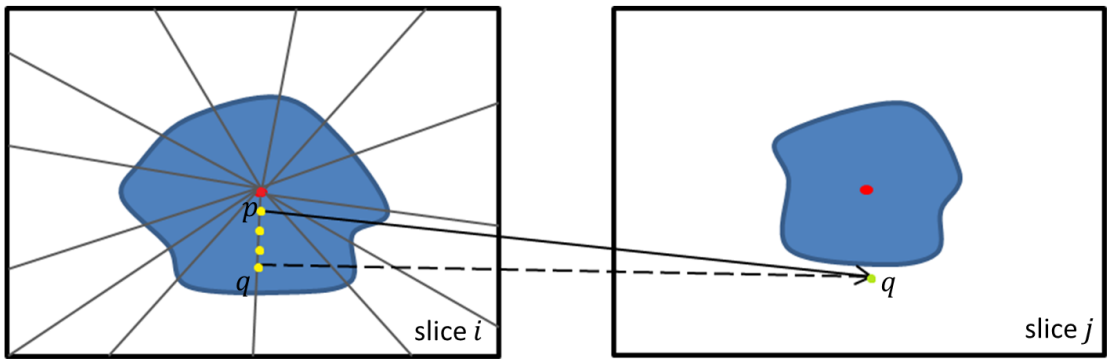


Figure 3.14: Pairwise inclusion in star shape direction. Left: Slice i. The gray lines are the star shape directions, red dot is the star center and the yellow dots are discrete pixels along a star shape line. Right: Slice j. The red dot is the star center and the green dot is pixel q which has the same index with pixel q on slice i. When adding inclusion constraint with margin, the constraint is imposed between pixel p on slice i and pixel q on slice j.

approximate a margin with a particular size. If a margin of x pixels is needed, it can be approximated by adding constraints between pixel p and pixel $q = p + x$, where pixel $p + x$ is the xth

pixel after pixel p along the line connecting star center c and pixel p. As shown in Figure 3.14, $x = 3$ here which means it allows a margin of 3 pixels.

Incorporating pairwise inclusion into our approach means adding another term in the energy function. Recall that graph cut can only optimize the submodular functions. It is obvious to see from Table 3.2 that the pairwise inclusion term satisfies the submodular condition in Equation 2.7. Therefore, our energy function with inclusion becomes like

$$\begin{aligned} E(f) = &\sum_{p\in\mathcal{P}} D_p(f_p) + \lambda \sum_{\{p,q\}\in\mathcal{N}} V_{pq}(f_p, f_q) + \sum_{\{p,q\}\in\mathcal{N}} S_{pq}(f_p, f_q) \\ &+ \sum_{p\in\mathcal{P}} B_p(f_p) + \sum_{p\in\mathcal{P}} I_p(f_p) + \sum_{\{i,j\}\in\mathcal{L}} I^{ij}(f_i, f_j) \end{aligned} \tag{3.8}$$

where $I_p(f_p)$ is the new mask inclusion term, $I^{ij}(f_i, f_j)$ is the new pairwise inclusion term and $\mathcal{L}$ is the slice set.

Figure 3.15 shows the segmentation result with and without inclusion constraints. The slices marked with yellow boundary is the middle slice. The inclusion constraint applies in the direction from the middle to the side. The segmentation of the side slice is included in the segmentation of the previous slice with the inclusion constraint as shown in the second row of Figure 3.15

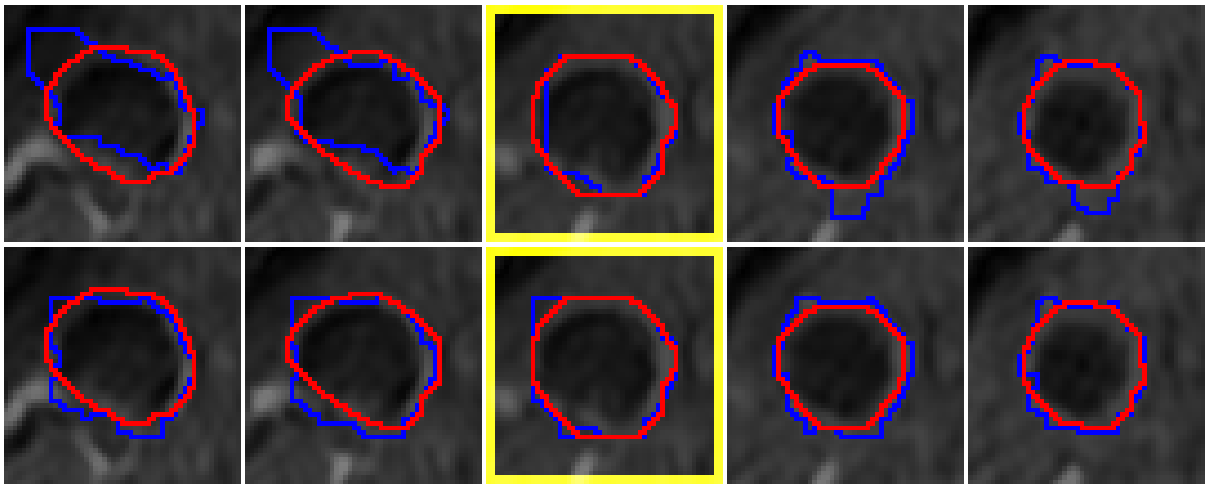


Figure 3.15: Segmentation comparison of slices in one volume. The slice with the yellow boundary is the middle slice. The first row is the segmentation without inclusion constraint. The second row is the segmentation with inclusion constraint.

3.5 Segmentation with 3D Slicer

New algorithms are usually incorporated into 3D slicer through extensions. A Slicer extension could be seen as a delivery package with new functionality. It can bundle together one or more modules. After installing an extension, the associated modules will be presented to the user as built-in ones. Extensions allow developers to build their new algorithms and its dependent modules into a package and then contribute to Slicer. After the publication of the extension through Slicer community, the new algorithms can be easily used for multiple operation system and different version of Slicer.

We integrate our approach into an extension for the practical medical use. Our brain tumor extension takes advantage of Slicers interactive tools such as Editor module and Annotation module to get user input. The user input for our algorithm are only two clicks at the center of two end slices and a bounding rectangle containing the tumor area in the middle slice where the user thinks having the biggest area. Since we use two corner points to define a bounding rectangle, the total user input we need is four clicks. The four points are marked using the fiducial in Annotation module. A bounding box can be generated based on the four clicks. 3D Slicer can take both the whole MRI brain volume and the cropped volume as the input. If it is a whole volume, the Corp Volume module needs to be called to crop the volume using another larger box outside the bounding box. Figure 3.16 shows the cropping box in a MRI brain volume. Then, a subvolume containing only the information we are interested in is taken out. And it is the input data of our approach. After the computation of our algorithm, a label map file will be generated indicating the label of each voxel. Slicer assigns different colors for different labels to show the final segmentation. Users can edit the label map file to refine the result with the Painter tool in Editor module. The modified label map can be used for another segmentation based on the edited parts. To make the user aware of the time they spend on refining the label map, we calculate and display the editing time for each edit. Due to Slicers comprehensive functions, all the relevant files from the segment scene to label map to fiducial marking points can be saved for later use.

The structure of our brain tumor segmentation can be found in Figure 3.17. The extension integrates the function of Annotation Module, Crop Volume Module, Editor Module and our Tumor Segmentation Module. Users mainly interact through GUI of Tumor Segmentation Module. And the inputs and operation are passed to our algorithm in logic layer. And 3D Slicer's MRML library provides API for accessing the image data and visualization.

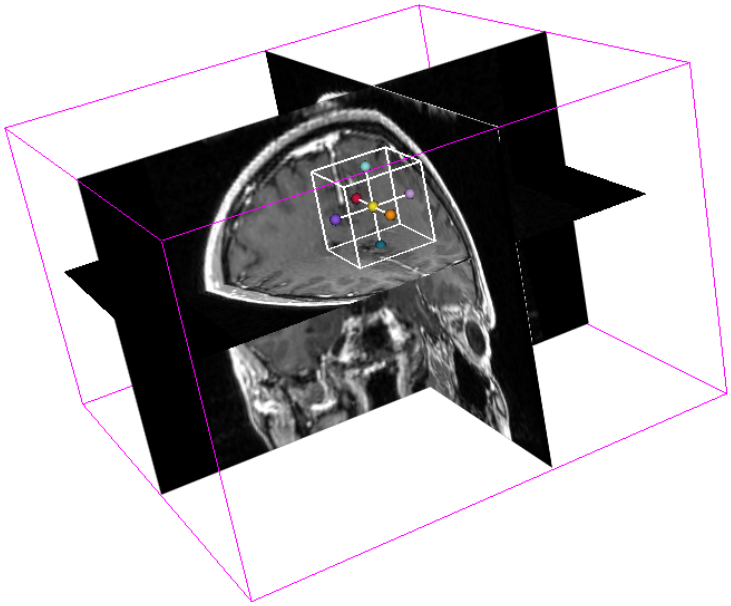


Figure 3.16: Cropping box in MRI brain volume.

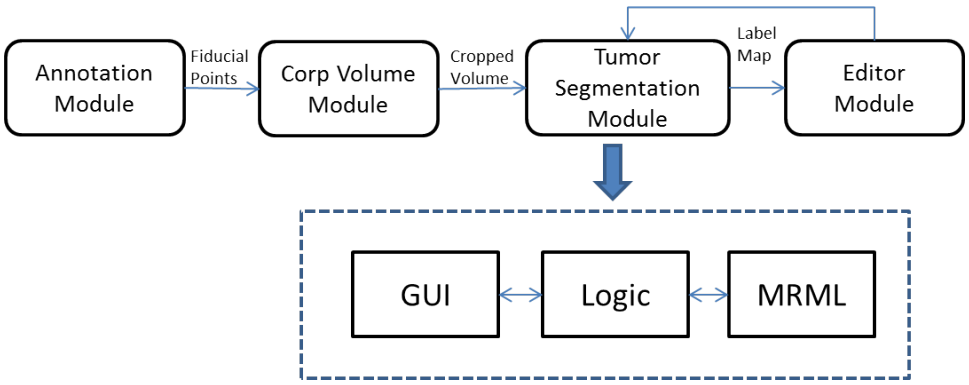


Figure 3.17: Structure of our extension.

Chapter 4

Experimental Results

In this section, the implementation details of our approach and the result evaluation and comparison will be discussed. There are many experiments have been tried, only the worthy ones are selected to be introduced here.

4.1 Implementation Details

4.1.1 Simulation of user interaction

The user input has great affects for the segmentation result. To ensure the coherence of the experiment input, we still use the user inputs simulated from the ground truth by Liu [25]. The initial input for our approach is the two clicks on the end slices marking the star centers and two clicks on one of the middle slice to specify the 2D bounding box containing the whole tumor within that slice. The two clicks on the middle slice are regarded as the diagonal corner points of a rectangle. They can be extracted from 4 edge point of the ground truth in the middle slice. And the other two clicks on the end slice are randomly picked pixels near the center of the end slices' center. The simulated bounding box is a tight one and real user input cannot be so precise. In [25], Liu performs sensitivity experiment for bounding box. It will not be given further discussion here.

4.1.2 Parameter Selection

There are two parameters needs to be provide beforehand when running the algorithm. They are the soft inclusion constraints K_1 and K_2 in Table 3.1 and Table 3.2. The value of these two parameters come from the training of the whole data set. To avoid overfitting, we use cross-validation. The candidate values of K_1 and K_2 are from 0, which indicates no inclusion

constraint, to 1000, which means hard constraint. Training the two parameters together consumes too much time due to the large candidate value range. Hence, they are trained separately with the other setting to 0. The training procedure is shown in the algorithm below. First, we compute a f-measure matrix for each file with corresponding candidate parameter value. Then, the f-measure row of each file is taken out at a time. And the column average of the rest matrix is calculated. The column with the highest average get one vote. Finally, the corresponding candidate value with the highest votes is chosen.

Algorithm 1 Cross Validation Training

1: **for** candidate value $v = v_{min}, ..., v_{max}$ **do**
2: **for** file $f = 1, \ldots, k$ **do**
3: Compute the f-measure for file f with the value v
4: **end for**
5: **end for**
6: **for** file $f = 1, \ldots, k$ **do**
7: Take out all the f-measures corresponding to file f.
8: Average the other f-measure matrixes and find the highest average F-measure, increase the vote for its candidate value v by 1.
9: **end for**
10: Take the candidate value v with the most votes as our choice.

The rest parameters λ, controlling the relative weight of data term and smooth term, and b, the volume ballooning force, require the local intensity information of each slice. Therefore, they are searched in the slice by slice 2D segmentation. Ideally, there should be a grid search for all the combinations of possible λ value and b value. But the efficiency of our algorithm will be greatly reduced. Hence, we do smoothness search first without ballooning and then search the best ballooning force with the selected smoothness.

When searching relative weight parameters λ, an appropriate range is set up first through binary search according to the segmentation area. The best parameter λ is then searched within the range based on the segmentation compactness measure. The best λ search algorithm is illustrated as following.

As for volume ballooning force, the searching process is the same with smoothness.

Algorithm 2 Parameter Searching

$\lambda_{middle} = \lambda_{left} + \lambda_{right}$
while $it < iteration\_time$ **do**
  if $area > max\_area$ **then**
    $\lambda_{right} = \lambda_{middle}$
  else
    break
  end if
end while
while $it < iteration\_time$ **do**
  if $area < min\_area$ **then**
    $\lambda_{left} = \lambda_{middle}$
  else
    break
  end if
end while
{Binary search for the range}
for $\lambda = \lambda_{left}, ..., \lambda_{right}$ **do**
  $compact$=SEGMENT(λ)
  if $compact < compact_{best}$ **then**
    $compact_{best} = compact$
    $\lambda_{best} = \lambda$
  end if
end for
{Search for best λ}

4.2 Experimental Results

4.2.1 Image Data

Our data is provided by Aaron Ward and Glenn Bauman from Robarts research. There is totally 64 MRI brain scans from 27 patients. Some scans from the same patient in different stage. Liu [25] extracted the parts containing tumors from the MRI data for the convenience of experiment. There are 88 cropped tumor volumes in total. The voxels are represented by 16-bit integers. Each volume has a ground truth provided by the expert. Some ground truth is imprecise because it was obtained by blob-manipulation tools.

4.2.2 Evaluation Methods

Error rate is a commonly used evaluation statistic for image segmentation. The specific equation is

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

where FP is the number of background pixels wrongly labeled as foreground, also known as false positive, and FN is the number of foreground pixels wrongly marked as background, i.e false negative. The ratio of all falsely labeled pixels and the total pixels need to be labeled evaluates the performance of the segmentation algorithm. However, this statistics is not as suitable for our approach because it can be greatly influenced by the size of bounding box. Assume that there are two identical segmentations, one is within a small bounding box while the other is within a big bounding box. The error rate for the identical segmentations is smaller for the larger bounding box case, since the number of TN (true negative) pixels is larger. TN refers to the background pixels that are marked correctly. The larger is the bounding box, the smaller will be the error for the same segmentation result. In other words, the error statistics is influenced by the ratio of the tumor/background pixels in the ground truth.

The more appropriate evaluation statistics for us is the F-measure, which is based on the precision and the recall of the result. Precision is the fraction of retrieved pixels that are object, and recall is the fraction of object pixels that are retrieved. In simple terms, high precision means that an algorithm returned substantially more relevant results than irrelevant, while high recall means that an algorithm returned most of the relevant results. The equation for F-measure is list below.

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

and

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

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

F_β score is the weighted average of precision and recall. β decides the relative importance between precision and recall. We set $\beta = 1$ here to let precision and recall be equally treated. Note that TN (true negatives) does not influence the F-measure.

4.2.3 Evaluation of the Results

Chapter 4 demonstrates all the methods for improving the interactive brain tumor segmentation. However, they have different influence on the performance of the final result. Thus, we tried various combinations of these methods and compare the F-measure averaged on all 88 volumes.

Comparing Compactness Measures

First, we will discuss the results using different compactness measures. The comparison of the average F-measure are listed in Table 4.1. When there is only star shape constraint, we can see

method	recall	precision	FM
$SCir$	0.8805	0.8667	0.8679
SE	0.8750	0.8697	0.8672
SC	0.8853	0.8668	0.8699
$SCirV$	0.9187	0.7924	0.8374
SEV	0.8827	0.8548	0.8626
SCV	0.9172	0.8132	0.8518

Table 4.1: Average F-measure of different methods. 'S', 'Cir', 'E', 'C' and 'V' stands for star shape constraint, circularity compactness, ellipsity compactness, convexity deviation compactness, and volume ballooning respectively. Each item of the first column is a combination of different options.

there is not much difference between circularity compactness and ellipsity compactness. And convexity deviation is slightly better than the other two. When the volume ballooning is added, ellipse fitting outperforms other two compactness measures.

Incorporating volume ballooning without inclusion will worsen the result because there is not strong enough constraint between slices. The end slices can get large segmentation even with small ballooning because of weak foreground/background appearance. We will show later that volume ballooning has a better performance with inclusion constraints.

Results with Inclusion Constraints

Next, the effect of inclusion constraints will be analyzed. To focus only on the inclusions constraints, we compare the results fixing the compactness measure to ellipse fitting here. Recall that inclusion constraint consists of two parts, mask inclusion and pairwise inclusion, and it has an option to have margin. The average F-measure results of all possible combinations are listed in Table 4.2. The soft constraints for inclusion are trained using the method in Section

method	recall	precision	FM
SEI_m	0.8479	0.8903	0.8626
SEI_p	**0.8582**	**0.9035**	**0.8762**
SEI_{mp}	0.8387	0.9033	0.8641
SEI_m^M	0.8554	0.8910	0.8682
SEI_p^M	0.8730	0.8744	0.8689
SEI_{mp}^M	0.8554	0.8914	0.8684

Table 4.2: Average F-measure of different combination of inclusion constraints with ellipsity compactness and star shape constraint. 'S', 'E', 'I', 'm','p' and 'M' stands for star shape constraint, ellipsity compactness, inclusion constraint, inclusion constraint using mask, inclusion constraint with pairwise term and inclusion constraint with margin respectively. Each item of the first column is a combination of different options.

4.1.2 with star shape constraint and ellipsity compactness measure.

Inclusion constraints with margin help to get better result compared to inclusion without margin. When there is no margin, the pairwise inclusion constraint yields the best result while the mask inclusion worsen the F-measure a little compared to the number of SE with no inclusion constraint in Table 4.1.

Result with Volume Ballooning

Table 4.3 shows the results of all combinations of inclusion constraints in Table 4.2 with volume ballooning. Comparing Table 4.2 and Table 4.3, volume ballooning only helps for no

method	recall	precision	FM
SEI_mV	0.9010	0.8311	0.8606
SEI_pV	**0.8890**	**0.8774**	**0.8793**
$SEI_{mp}V$	0.8926	0.8540	0.8690
SEI_m^MV	0.9224	0.7561	0.8176
SEI_p^MV	0.9038	0.8347	0.8622
SEI_{mp}^MV	0.9228	0.7567	0.8181

Table 4.3: Average F-measure of different combinations in Table 4.2 with volume ballooning. 'V' stands for volume ballooning.

margin pairwise inclusion constraint and using no margin mask and pairwise inclusion constraint together.

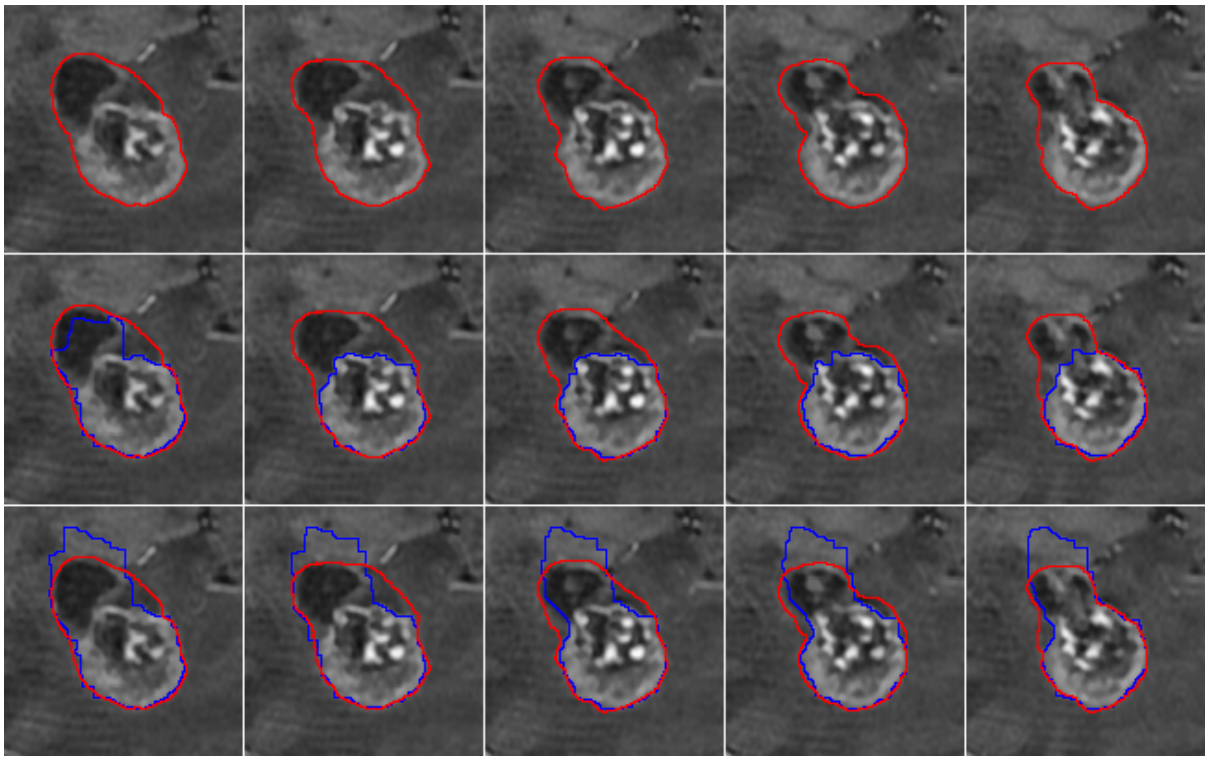
Comparing the F-measure result of volume ballooning without inclusion constraint in Table 4.1 and the results in Table 4.3, we can see inclusion constraint with no margin can help volume ballooning to get a better performance. Our best overall F-measure is 0.8793 segmenting with star shape constraint, ellipsity compactness measure, no margin pairwise inclusion constrain and volume ballooning, corresponding to the second line in Table 4.3.

4.2.4 Result Comparison

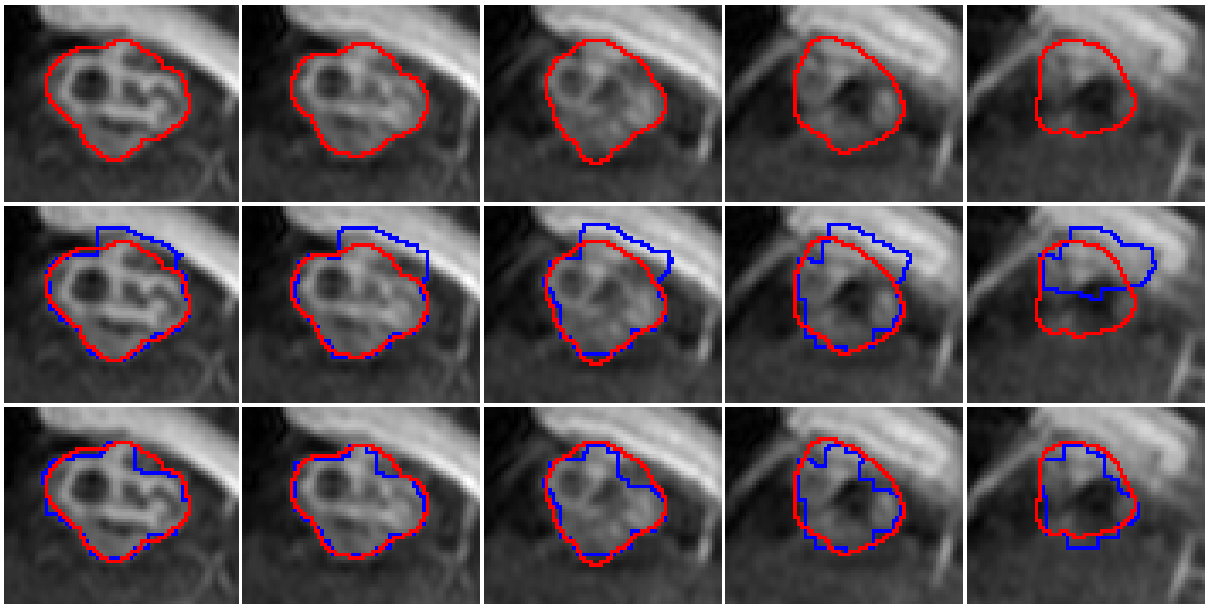
In this section, some representative examples are displayed to compare the final segmentation of different methods. First, three different kinds of compactness measure, circularity compactness, ellisity compactness and convexity deviation compactness will be compared. And the benefits of inclusion constraint will be illustrated next. Finally, the advantages of volume ballooning with inclusion constraint will be shown. Since we have too many combinations, there are too many comparisons that could be done. Therefore, we only choose the representative ones. A completed 3D brain tumor volume is too large to display. So only the 5 slices are sampled for each volume. Some failure cases are also shown in the comparison.

SCir **versus** *SE*

The average F-measure in Table 4.1 shows no big difference for circularity compactness and ellisity compactness when there is only star shape constraint. However, there are still cases where one works better than the other. Figure 4.1 shows two examples that ellipsity compactness measure gets better result. In these two cases, circularity compactness measure prefers segmentation more like a circle while ellipse fitting helps to choose segmentation more like an ellipse. Ellipse fitting is not always helpful. Figure 4.2 shows



(a)



(b)

Figure 4.1: Two examples where ellisity compactness is better than circularity compactness. For each example, the first row is the ground truth. The second row is the ground truth and segmentation with circularity compactness measure. Red contour is the boundary of ground truth and blue contour is the result of our segmentation. The third row is the ground truth and segmentation with ellipsity compactness measure.

two examples where ellipse fitting fails to get better segmentation.

SCirV **versus** *SEV*

When volume ballooning is incorporated, ellipsity compactness measure shows greater advantages than circularity compactness measure. Figure 4.3 lists two examples where ellipse fitting helps to get more accurate segmentation with volume ballooning force.

SE **versus** *SC*

Compared to ellipse fitting, convexity deviation can obtain more convex segmentation. Examples are shown in Figure 4.4.

SEV **versus** *SCV*

When volume ballooning added, convexity deviation compactness measure can still help to get more convex segmentation, as shown in Figure 4.5. But convexity deviation can also make the segmentation less accurate. In the examples in Figure 4.6, middle slice get more convex segmentation with larger region which also affects the local appearance model for the following slices. At the end, it results in inaccuracy for overall volume segmentation.

SE **versus** SEI_m

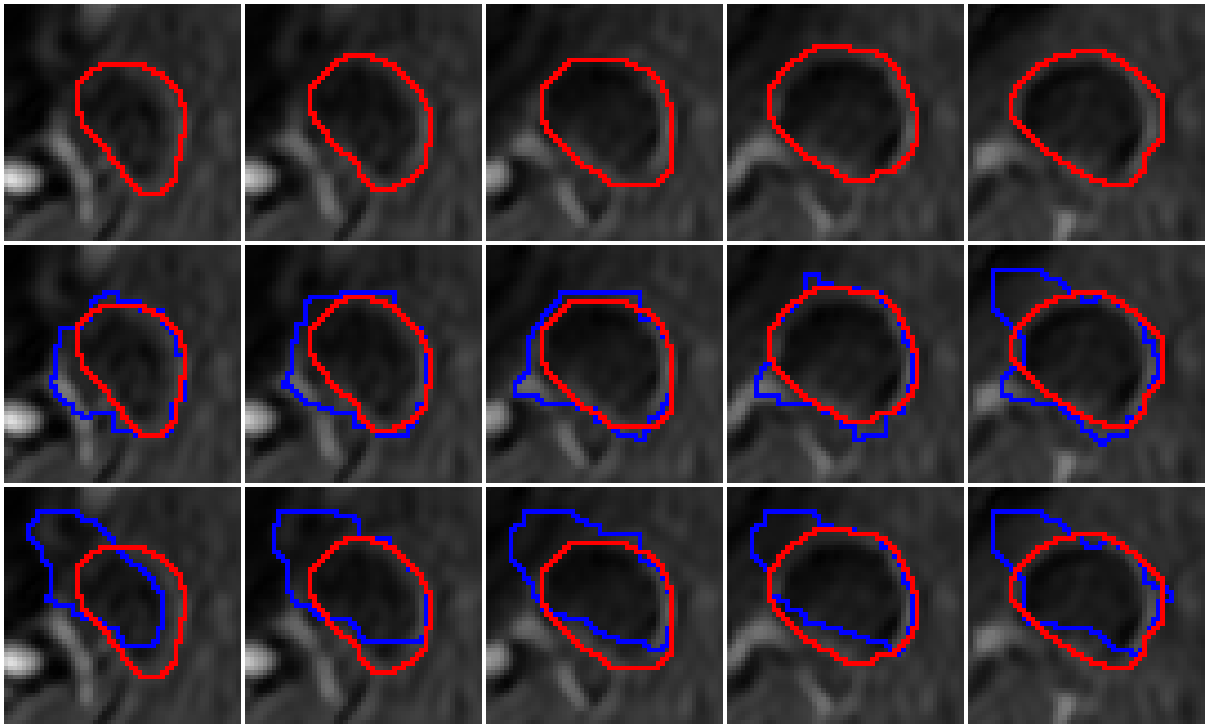
Inclusion constraint using mask can avoid the situation where the next slice segmentation outranges the previous one in the direction from the middle to the side. Representative examples are shown in Figure 4.7. However, inclusion using mask can also restrict next slices' segmentation. Cases like this are shown in Figure 4.8. In examples of Figure 4.8, the ground truth of each slice does not have too much difference. Thus their segmentation may not strictly follow the inclusion constraint. Adding inclusion with mask here somehow restricts the segmentation for next slice.

SEI_m **versus** SEI_m^M

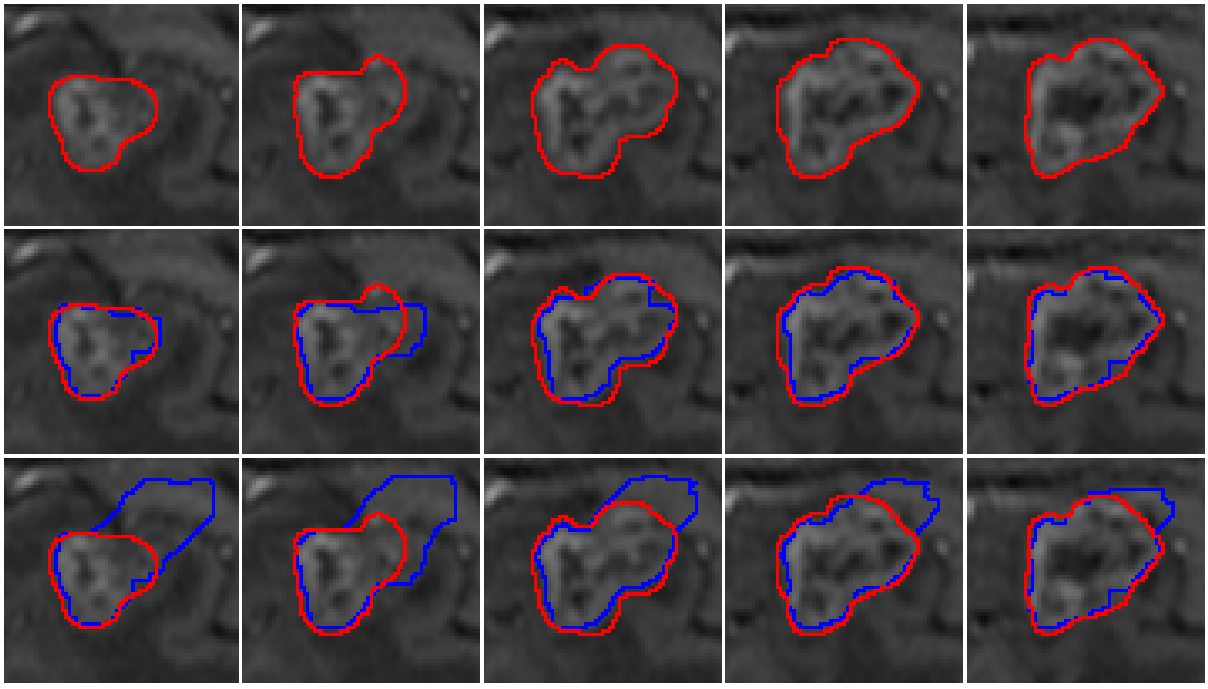
Allowing a margin for mask inclusion constraint can help with the next slice restriction problem. Two examples are shown in Figure 4.9. With a margin in mask inclusion, the next slice has more segmentation freedom.

SE **versus** SEI_p

Pairwise inclusion constraint can deal with the outrange situation as well. Figure 4.10 gives two examples where more accurate segmentation is obtained with pairwise inclusion constraint. In the first example of Figure 4.10, the tumor is well segmented even surrounded with similar appearance tissue. The second example also gets seperated from bright background with pairwise inclusion constraint.

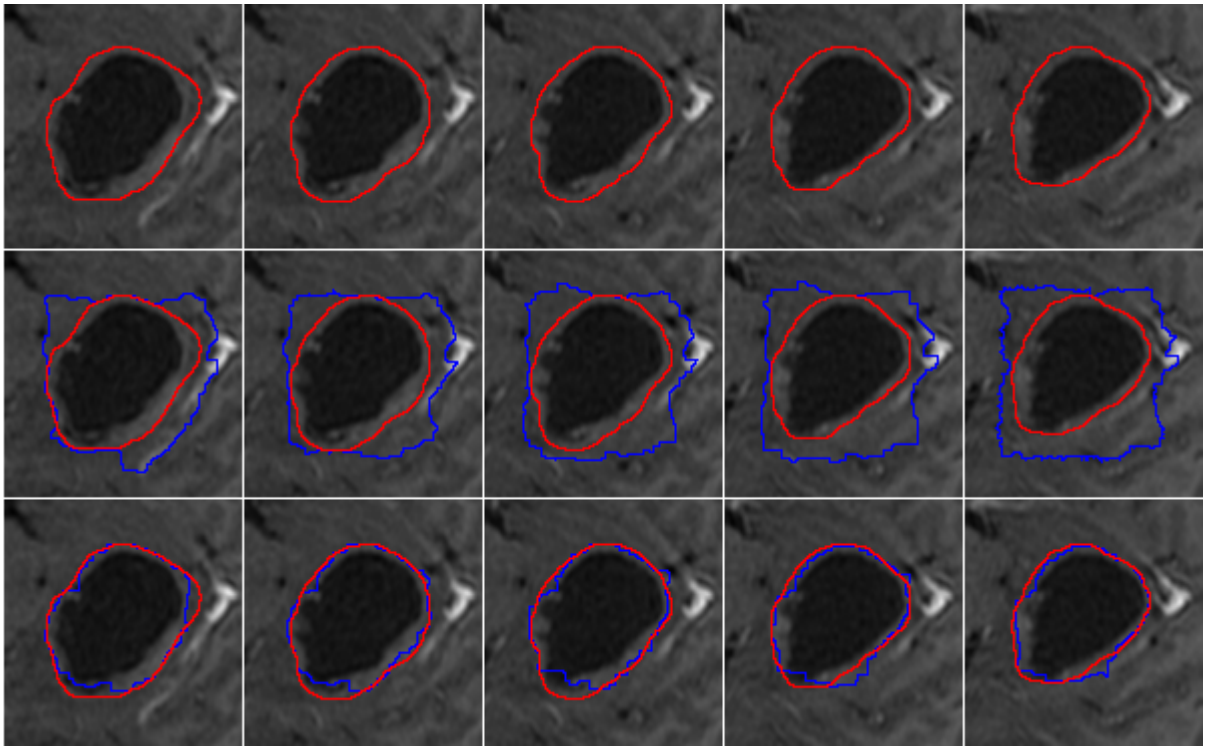


(a)

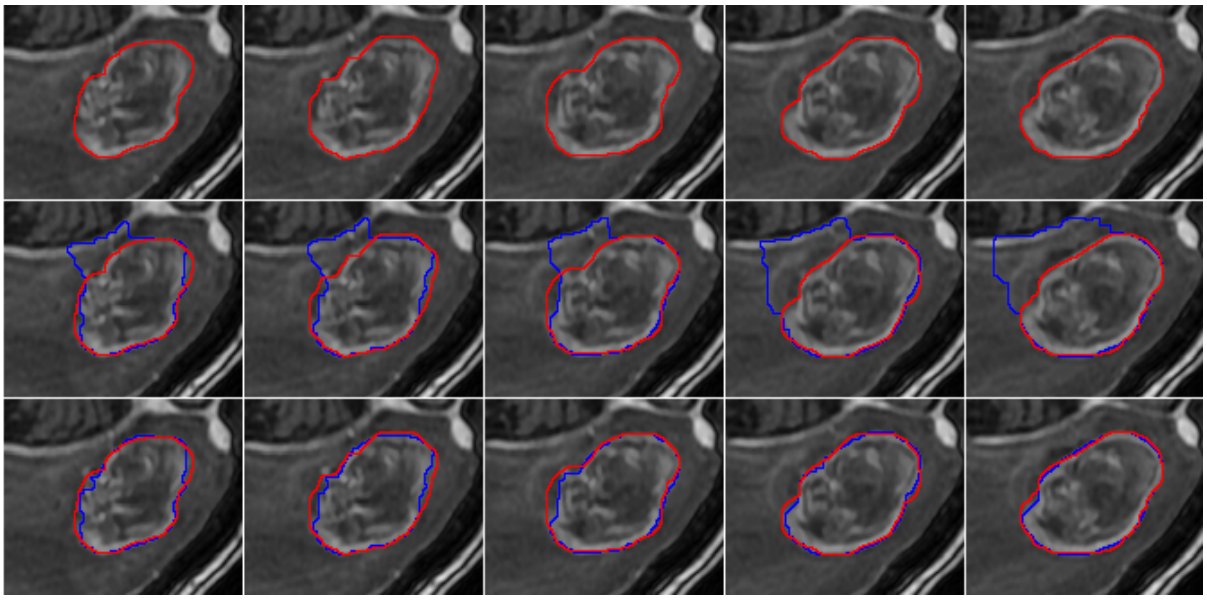


(b)

Figure 4.2: Two examples of circularity compactness better than ellipsity compactness. For each example, the first row is the ground truth. The second row is the ground truth and segmentation with circularity compactness measure. Red contour is the boundary of ground truth and blue contour is the result of our segmentation. The third row is the ground truth and segmentation with ellipsity compactness measure.

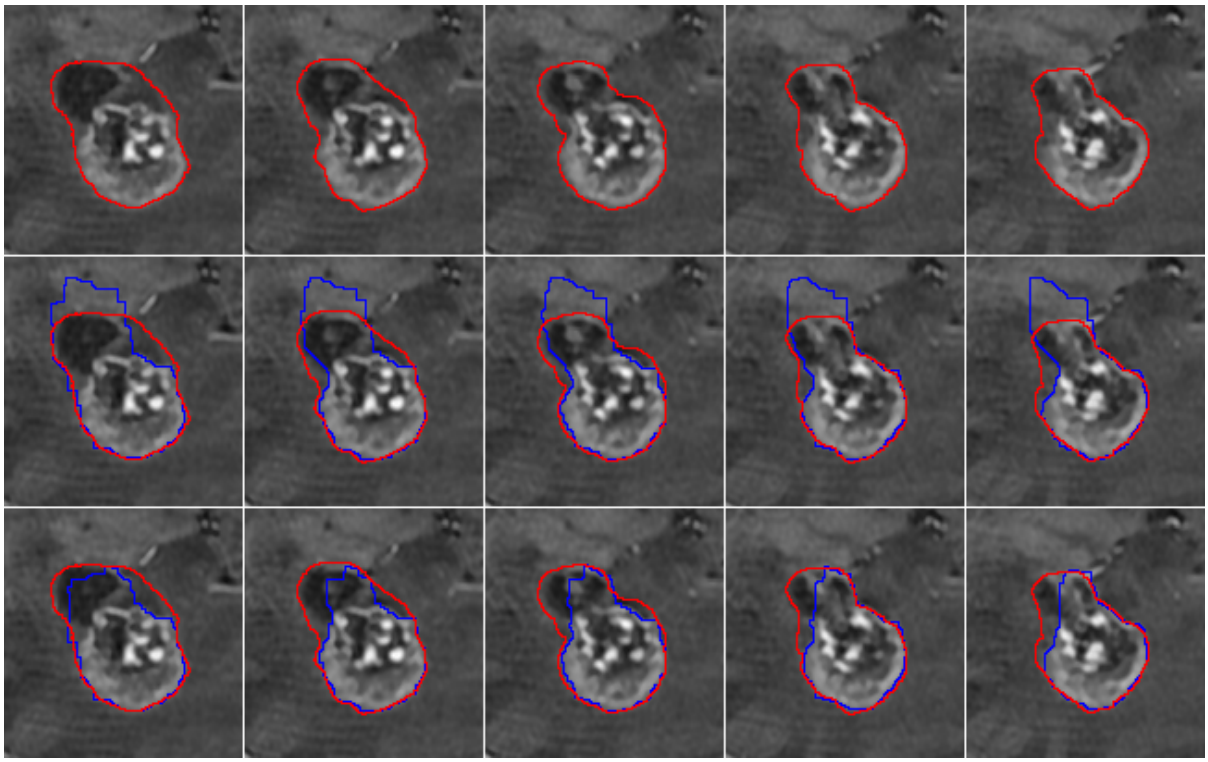


(a)

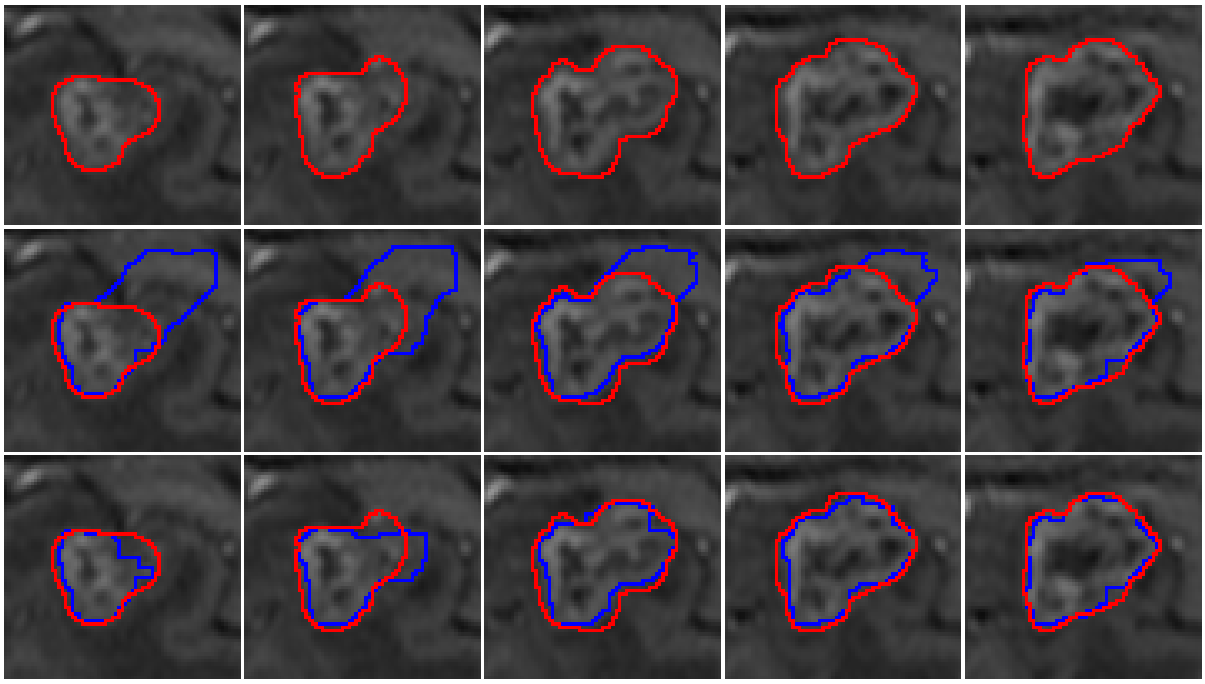


(b)

Figure 4.3: Two examples of ellipsity compactness measure better than circularity compactness measure when volume ballooning exists. For each example, the first row is the ground truth. The second row is the ground truth and segmentation with circularity compactness measure and volume ballooning. Red contour is the boundary of ground truth and blue contour is the result of our segmentation. The third row is the ground truth and segmentation with ellipsity compactness measure and volume ballooning.



(a)



(b)

Figure 4.4: Two examples of convexity deviation better than ellipse fitting. For each example, the first row is the ground truth. The second row is the ground truth and segmentation with ellipsity compactness measure. Red contour is the boundary of ground truth and blue contour is the result of our segmentation. The third row is the ground truth and segmentation with convexity deviation compactness measure.

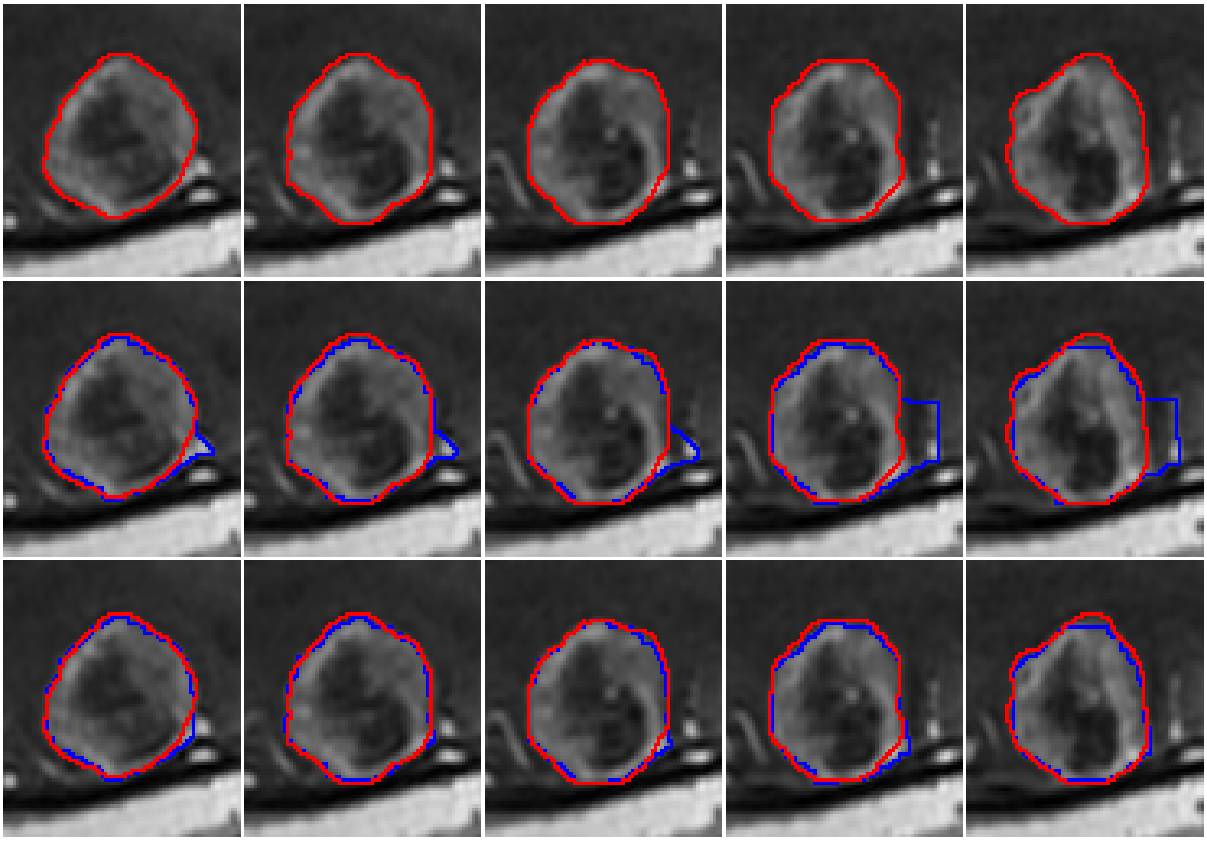
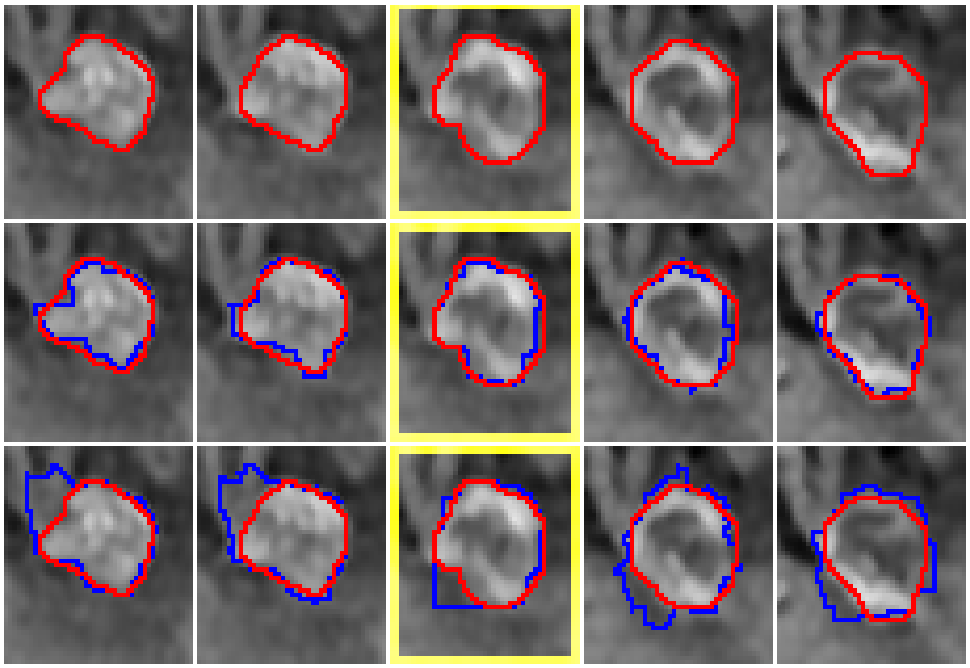
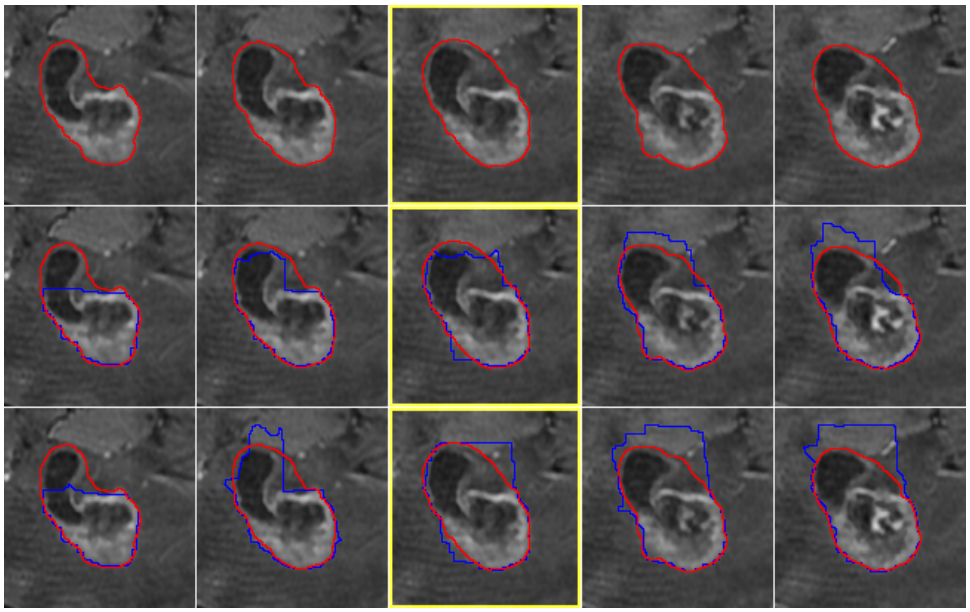


Figure 4.5: Example of convexity deviation better than ellipse fitting when there is volume ballooning. The first row is the ground truth. The second row is the ground truth and segmentation with ellipsity compactness measure and volume ballooning. Red contour is the boundary of ground truth and blue contour is the result of our segmentation. The third row is the ground truth and segmentation with convexity deviation compactness measure and volume ballooning.

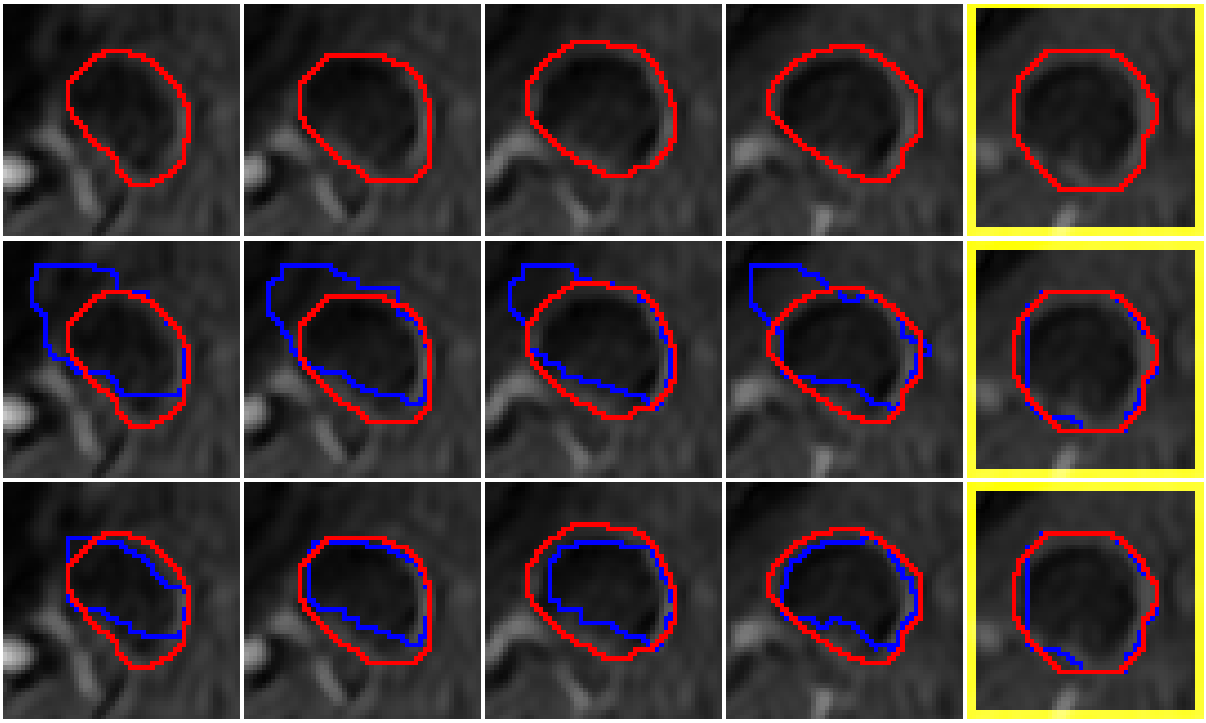


(a)

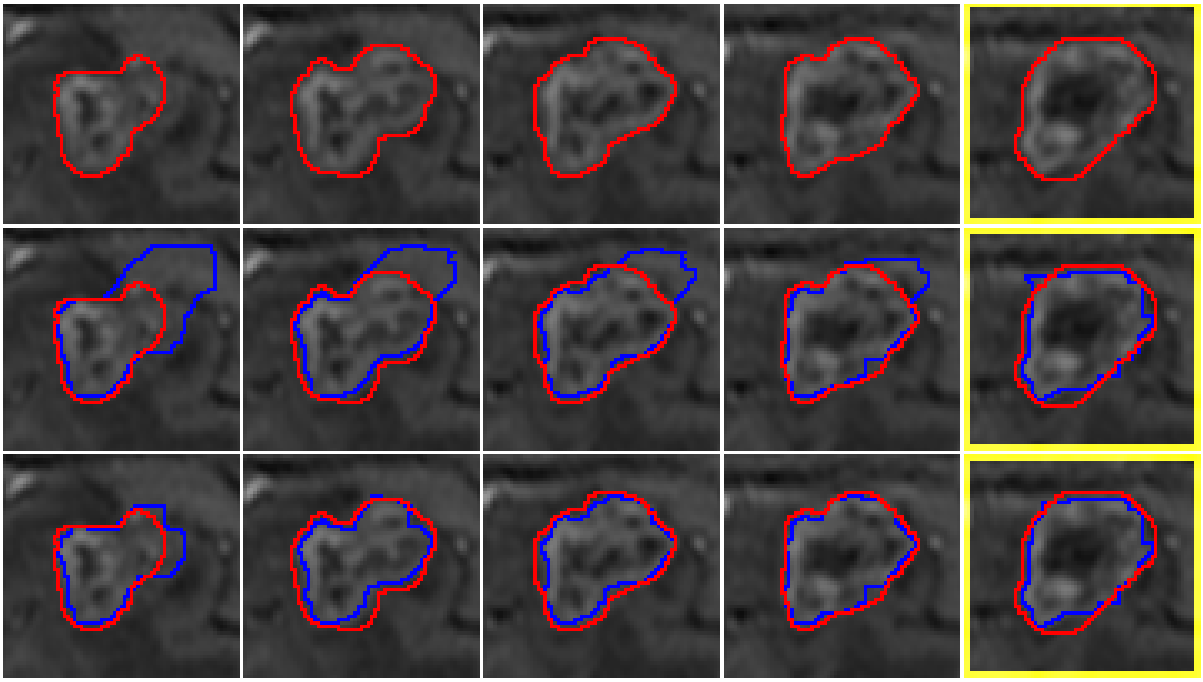


(b)

Figure 4.6: Two examples of convexity deviation gets less accurate segmentation. For each example, the first row is the ground truth. The second row is the ground truth and segmentation with ellipsity compactness measure and volume ballooning. Red contour is the boundary of ground truth and blue contour is the result of our segmentation. The third row is the ground truth and segmentation with convexity deviation compactness measure and volume ballooning. The slice with yellow rectangle is the middle slice.

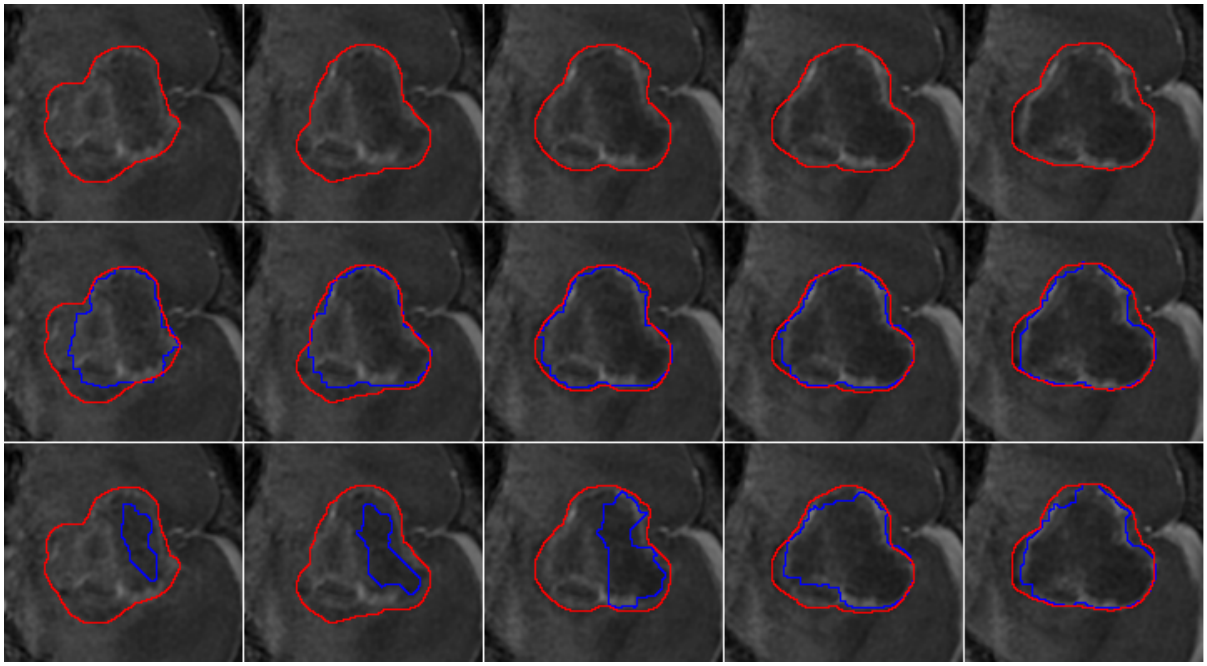


(a)

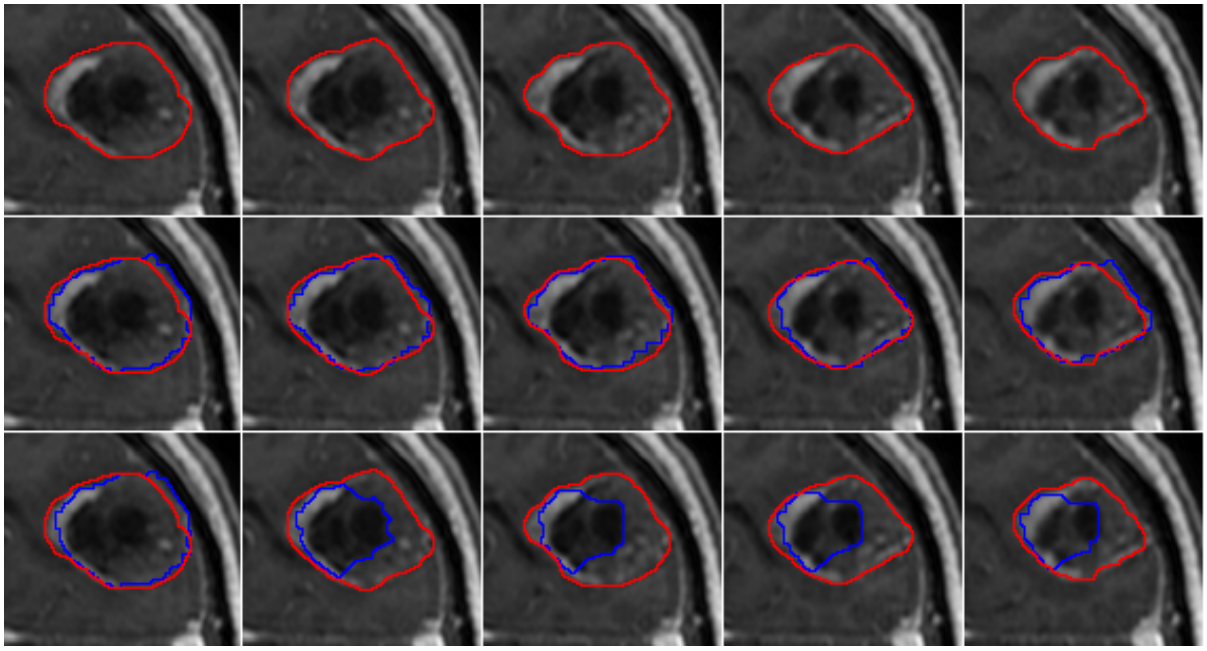


(b)

Figure 4.7: Two examples of mask inclusion gets better accurate segmentation. For each example, the first row is the ground truth. The second row is the ground truth and segmentation with ellipsity compactness measure. Red contour is the boundary of ground truth and blue contour is the result of our segmentation. The third row is the ground truth and segmentation with ellipsity compactness measure and mask inclusion. The slice with yellow rectangle is the middle slice.

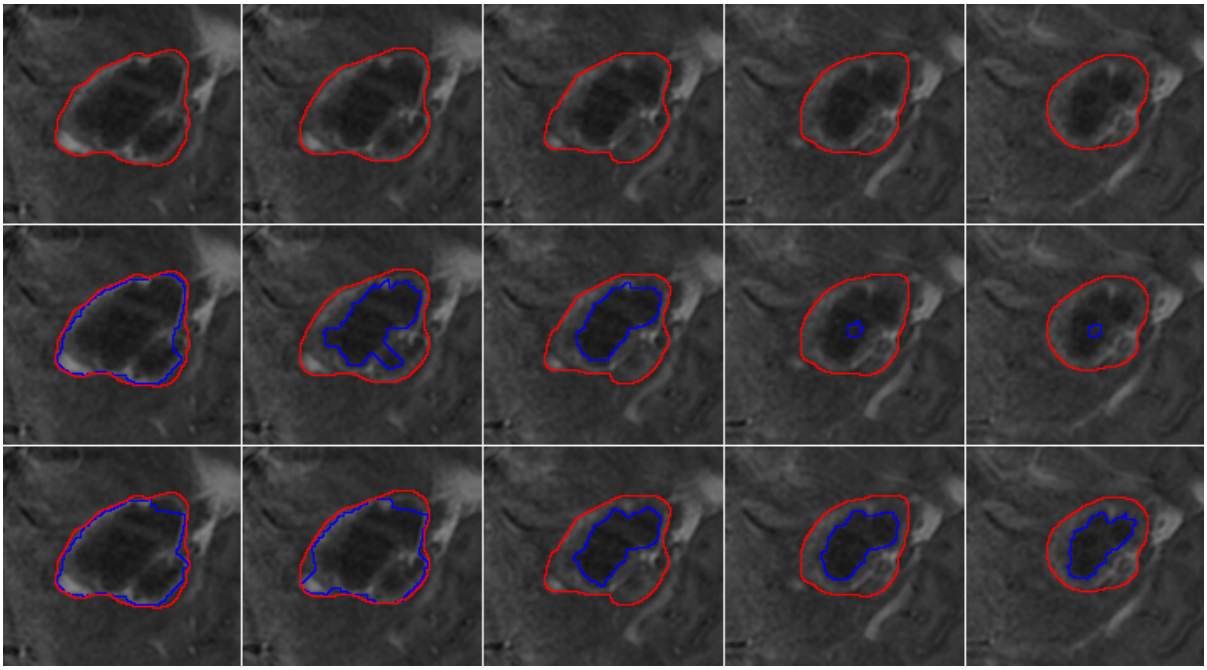


(a)

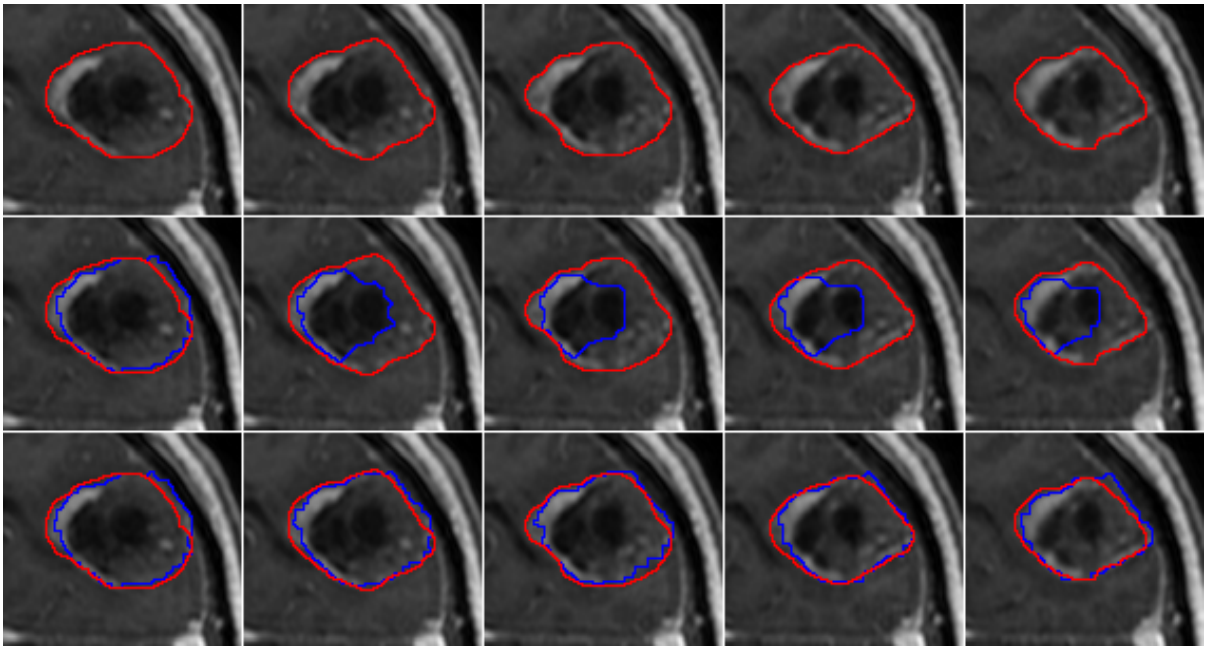


(b)

Figure 4.8: Two examples of mask inclusion restrict segmentation of next slice. For each example, the first row is the ground truth. The second row is the ground truth and segmentation with ellipsity compactness measure. Red contour is the boundary of ground truth and blue contour is the result of our segmentation. The third row is the ground truth and segmentation with ellipsity compactness measure and mask inclusion.

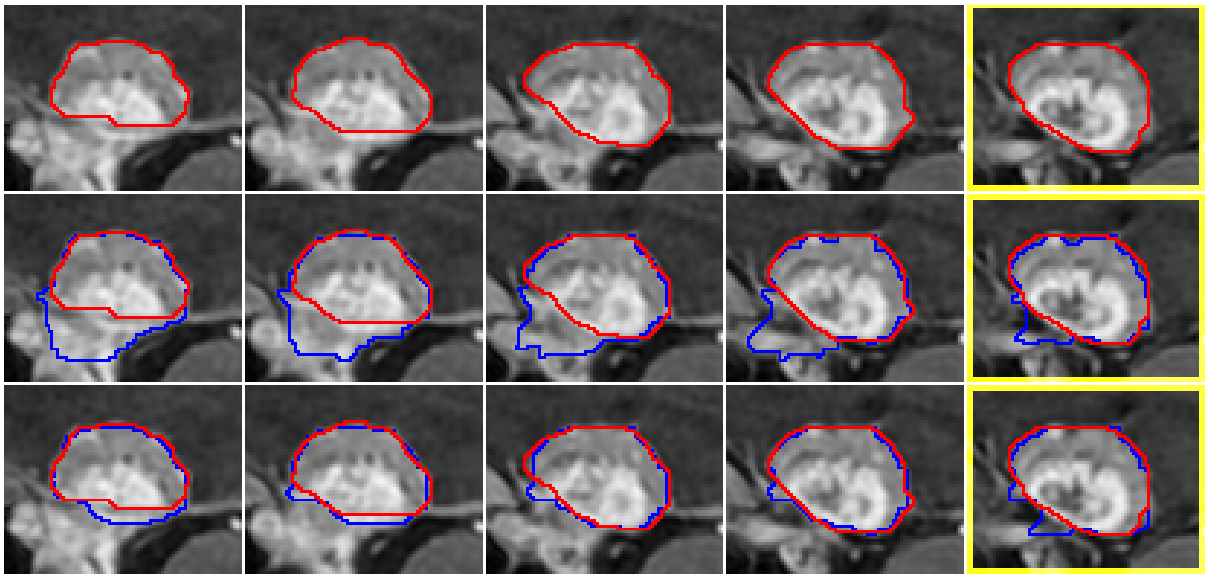


(a)

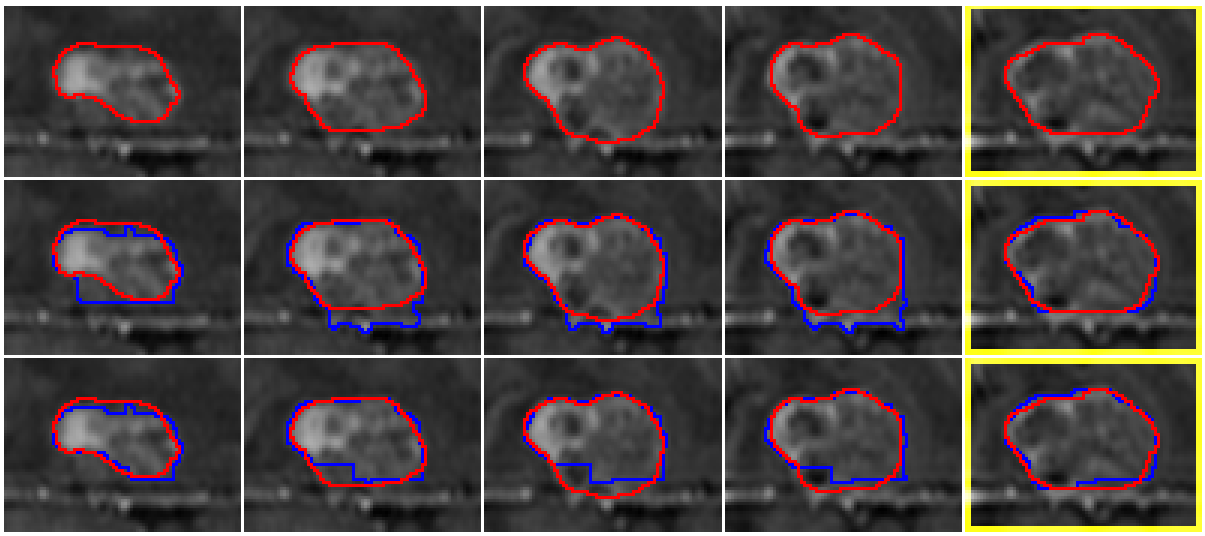


(b)

Figure 4.9: Two examples of allowing margin in mask inclusion. For each example, the first row is the ground truth. The second row is the ground truth and segmentation with ellipsity compactness measure and no margin mask inclusion. Red contour is the boundary of ground truth and blue contour is the result of our segmentation. The third row is the ground truth and segmentation with margin in mask inclusion.



(a)



(b)

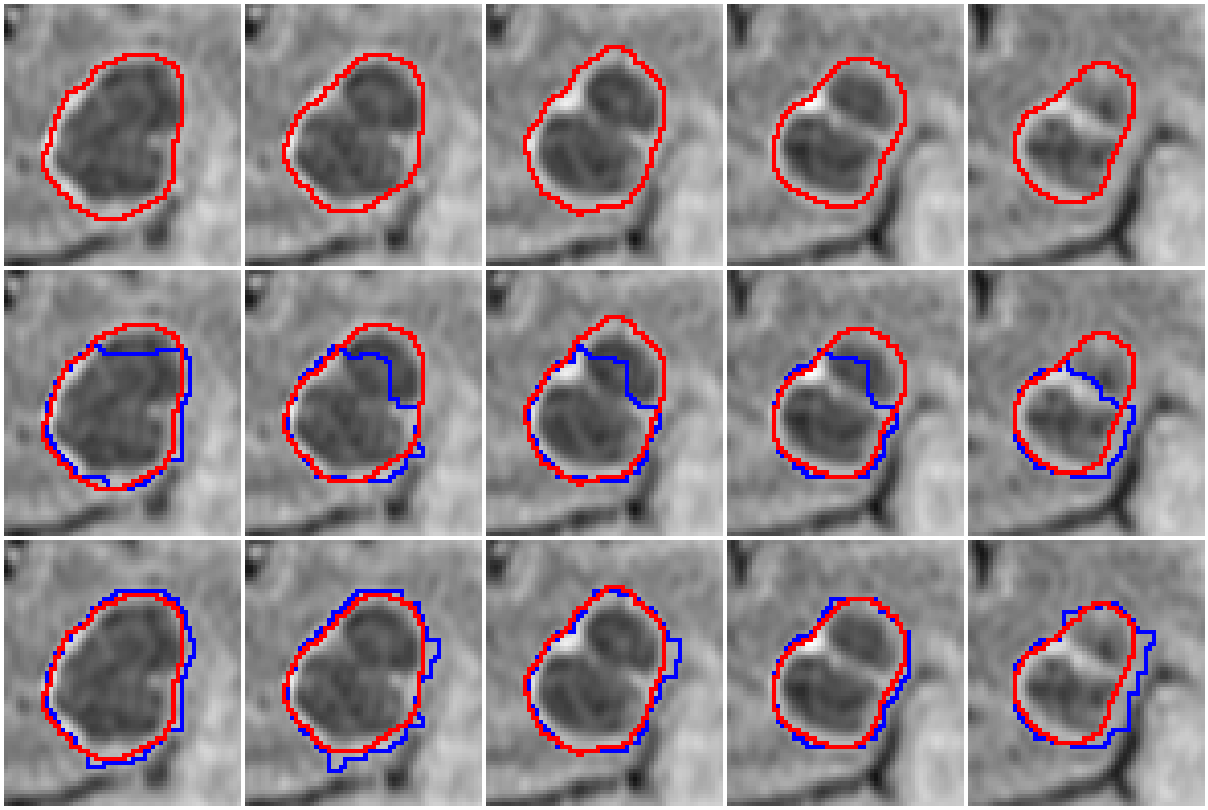
Figure 4.10: Two examples of pairwise inclusion constraint has better result. For each example, the first row is the ground truth. The second row is the ground truth and segmentation with ellipsity compactness measure. Red contour is the boundary of ground truth and blue contour is the result of our segmentation. The third row is the ground truth and segmentation with pairwise inclusion constraint.

SEI_p versus SEI_p^M

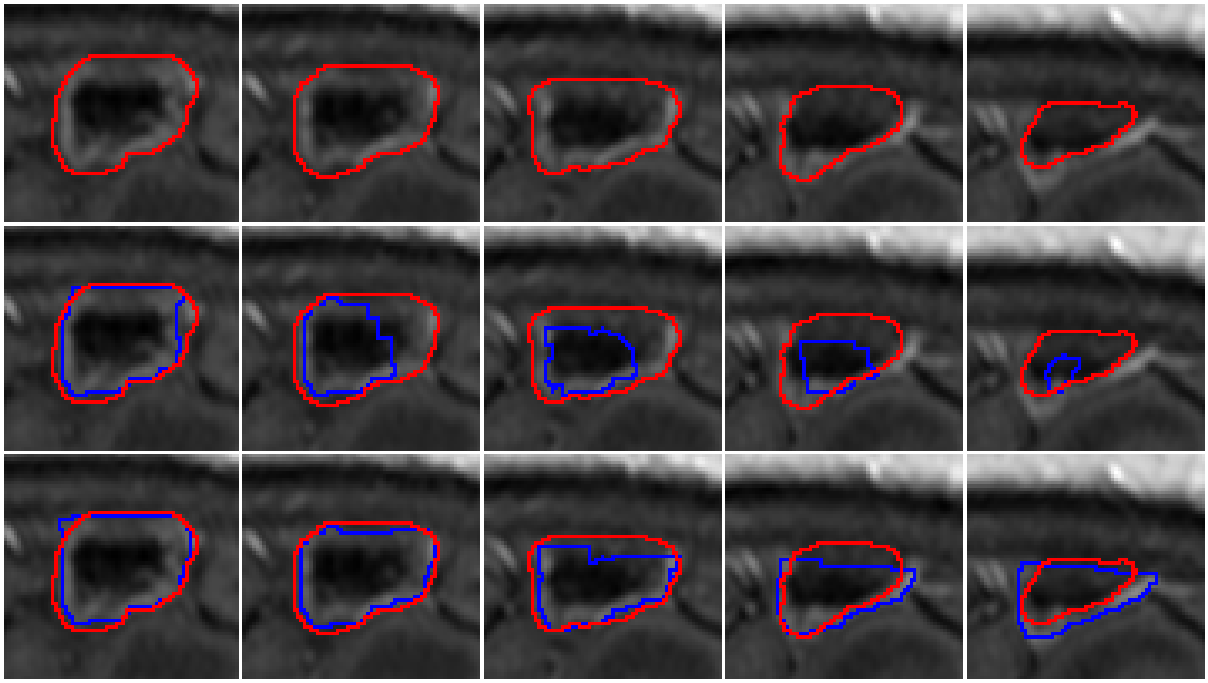
Similar to mask inclusion constraint, allowing a margin for pairwise constraint also help with the cases where the next slice's segmentation is restricted, as shown in Figure 4.11. But margin can worse some cases, such as the examples in Figure 4.12 where thereis no distinguishable foregorund and background appearance.

SEI_p versus SEI_pV

Recall that in Table 4.1 and Table 4.3 volume ballooning makes some F-measure results worse. But volume ballooning can help to obtain better result with pairwise inclusion constraint. Segmentation examples can be found in Figure 4.13. There are also few cases where adding volume ballooning makes the segmentation worse, as shown in Figure 4.14.

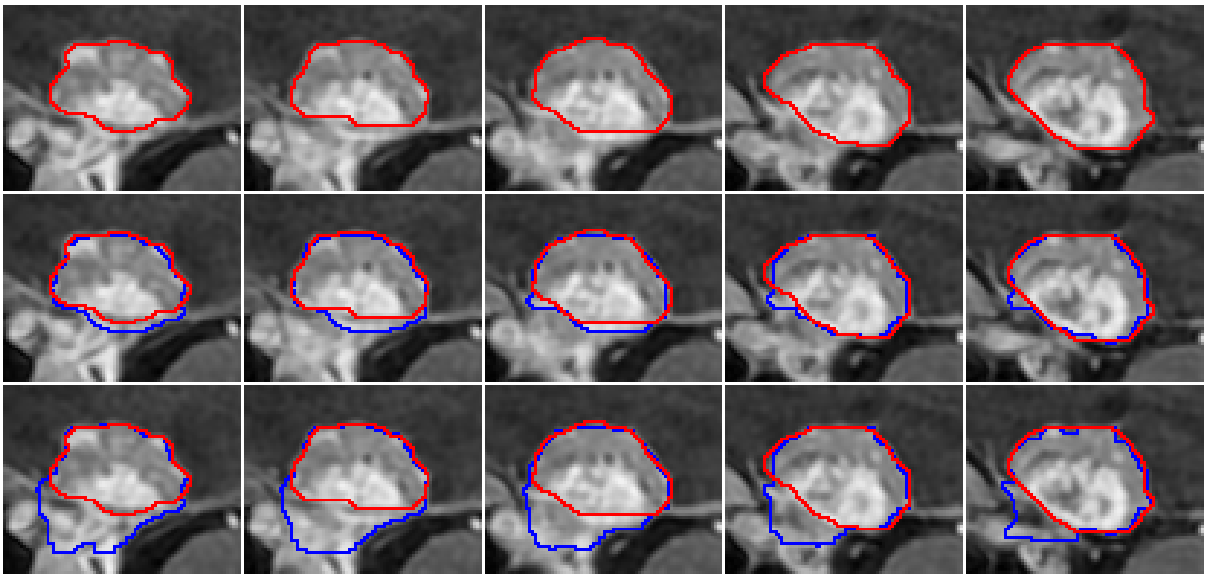


(a)

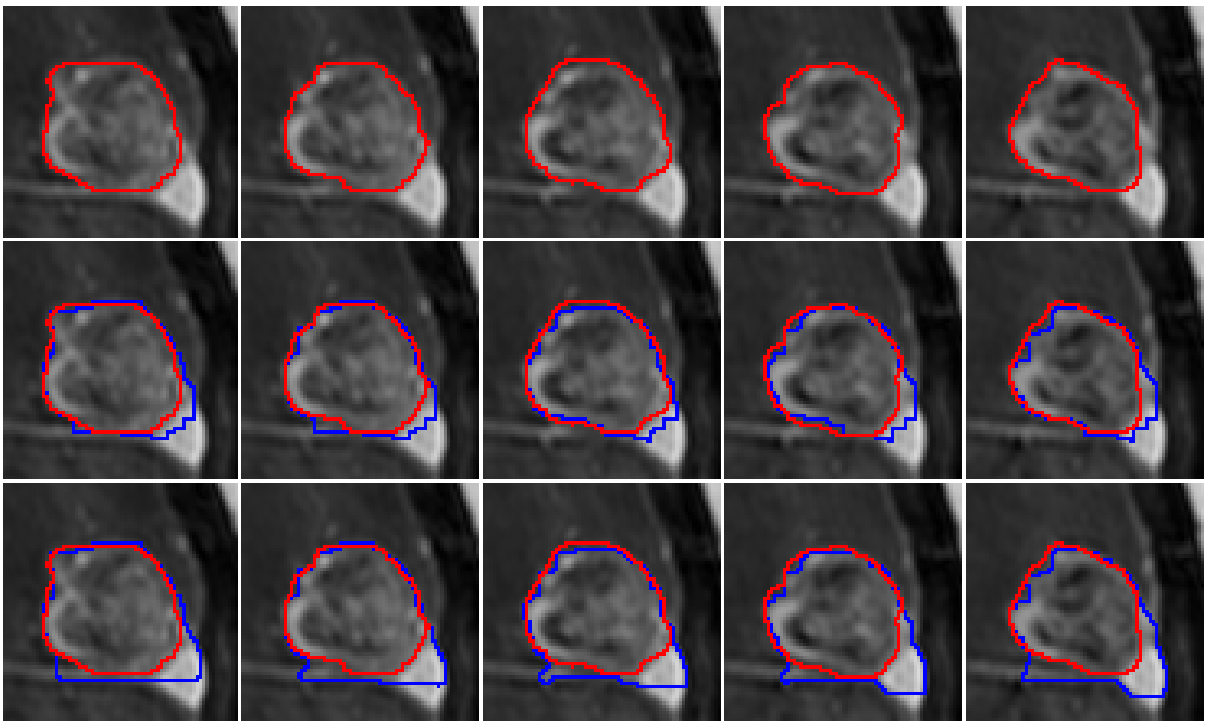


(b)

Figure 4.11: Two examples of allowing margin for pairwise inclusion constraint has better result. For each example, the first row is the ground truth. The second row is the ground truth and segmentation with ellipsity compactness measure and no margin pairwise constraint. Red contour is the boundary of ground truth and blue contour is the result of our segmentation. The third row is the ground truth and segmentation with margin in pairwise inclusion constraint.

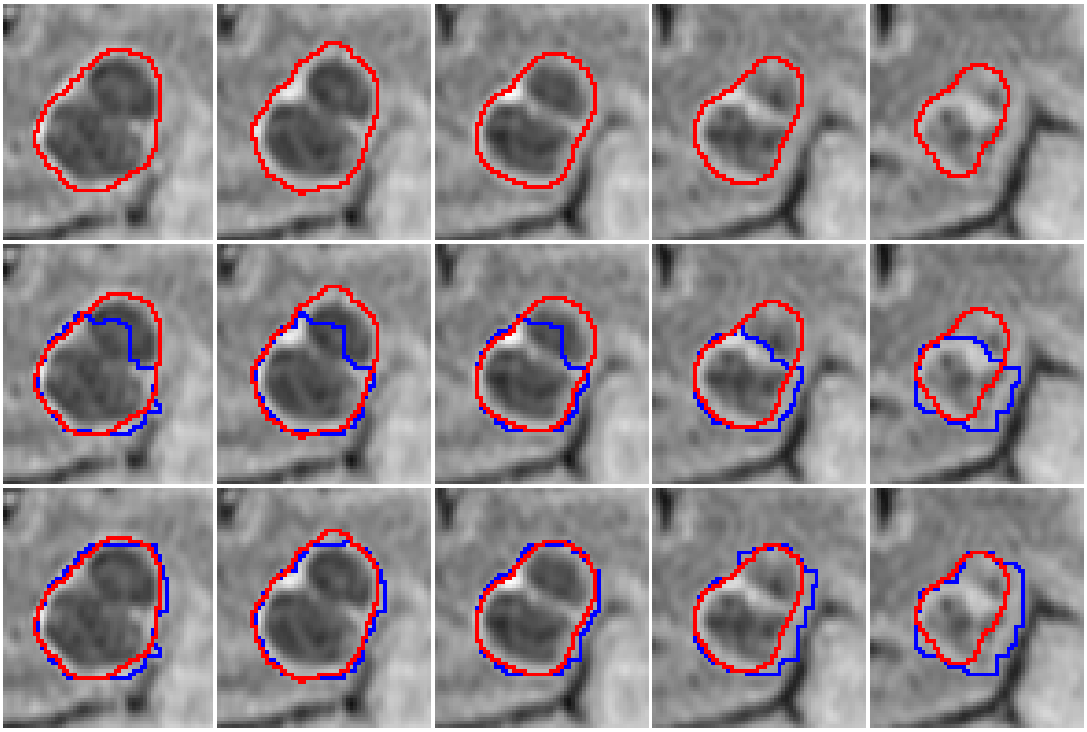


(a)

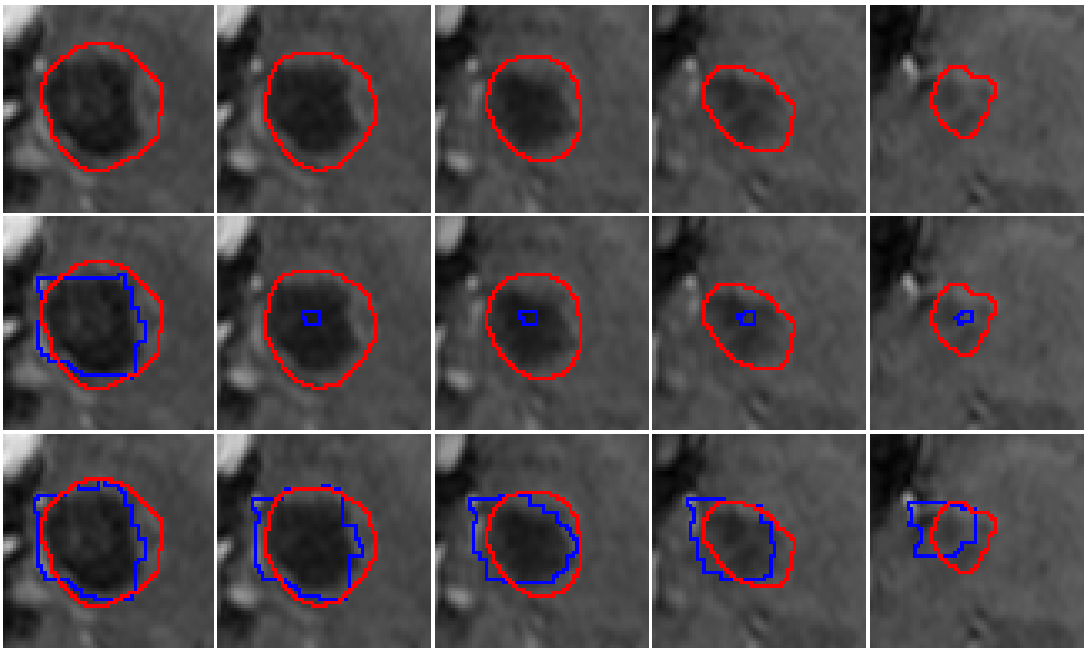


(b)

Figure 4.12: Two examples of allowing margin for pairwise inclusion constraint has worse result. For each example, the first row is the ground truth. The second row is the ground truth and segmentation with ellipsity compactness measure and no margin pairwise constraint. Red contour is the boundary of ground truth and blue contour is the result of our segmentation. The third row is the ground truth and segmentation with margin in pairwise inclusion constraint.



(a)



(b)

Figure 4.13: Two examples of volume ballooning with pairwise inclusion constraint has better result. For each example, the first row is the ground truth. The second row is the ground truth and segmentation with ellipsity compactness measure and no margin pairwise constraint. Red contour is the boundary of ground truth and blue contour is the result of our segmentation. The third row is the ground truth and segmentation with no margin pairwise inclusion constraint and volume ballooning.

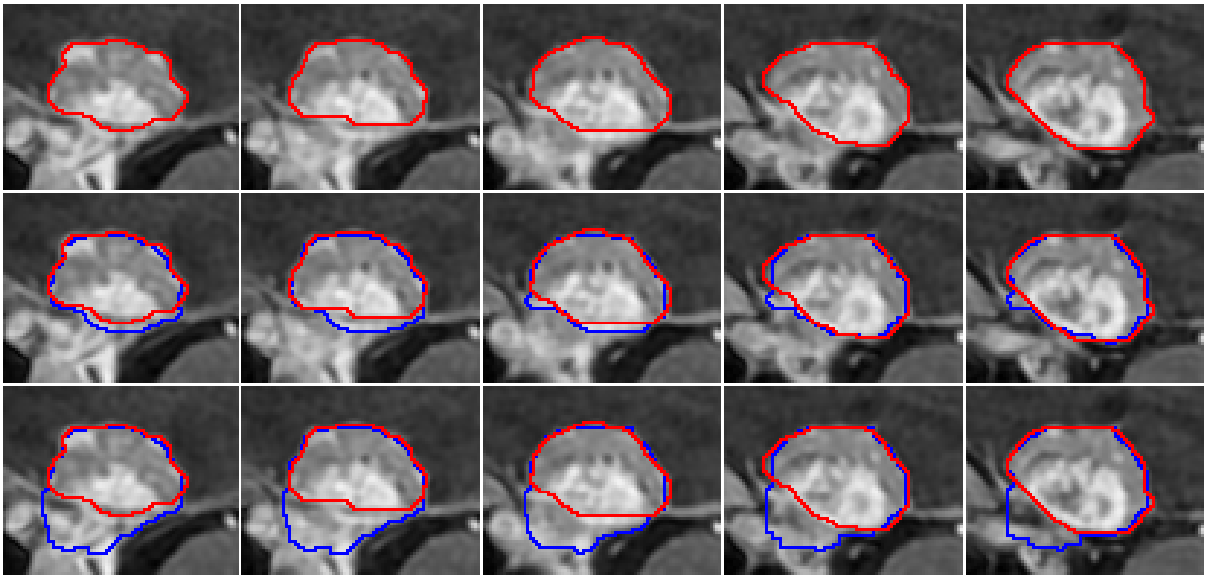


Figure 4.14: Example of volume ballooning has worse result with pairwise inclusion constraint. The first row is the ground truth. The second row is the ground truth and segmentation with ellipsity compactness measure and pairwise inclusion constraint. Red contour is the boundary of ground truth and blue contour is the result of our segmentation. The third row is the ground truth and segmentation adding volume ballooning.

Chapter 5

Conclusion and Future Work

In this thesis, we improved the interactive brain tumor binary segmentation approach based on Liqun Liu's [25] work. Inherited the minimal user assistance property, our system only needs 4 clicks of user input for segmentation initialization. Those are 2 clicks specifying a rectangle on the largest tumor region slice and another 2 clicks marking the centers of the first and last slice. Graph cut method is utilized for the energy optimization. Apart from the star shape constraint, we have also integrated volume ballooning and inclusion constraint into the framework, and tried two new compactness measure ellipse fitting and convexity deviation. In 2D segmentation, we make most of the local intensity of each slice and search different ballooning force adaptively for them. The compactness measure is served as the selection standard for parameter searching like volume ballooning parameters and relative weight in front of smooth term. In 3D segmentation, we add inclusion constraint between slices so that the side slices' labeling would not outrange the previous one. With all these improvements, the segmentation accuracy improves. Finally, the labeling of our approach can be further edited with foreground and background seeds according to users' requirement for resegmentation until a desired result is obtained.

In spite of that we have put forward some improvements for the interactive brain tumor segmentation, there is still space for improving our approach. Most errors are in the case when the foreground and background models have a significant overlap. Most tumors tend to be convex. We could incorporate a convex shape prior into our work, but current methods [16] are quite expensive. We could also investigate other shape priors, such as compact [9]. Other directions for improvement is incorporating the relative size constraints between 2D tumor slices.

So far we have had various constraints in graph cut segmentation, but not all of them are necessary to get a satisfying labeling when performing segmentation for different tumor cases. We can have these constraints optional for users so that they can decide the combination ac-

cording to the characteristics of the tumor data and choose the best with the most accurate segmentation. Moreover, we can let the user set some parameters such as the margin for inclusion constraint.

Bibliography

[1] Ben Appleton and Hugues Talbot. Globally optimal geodesic active contours. *Journal of Mathematical Imaging and Vision*, 23(1):67–86, 2005.

[2] Stefan Bauer, Roland Wiest, Lutz-P Nolte, and Mauricio Reyes. A survey of mri-based medical image analysis for brain tumor studies. *Physics in medicine and biology*, 58(13):R97, 2013.

[3] Yuri Boykov and Marie-Pierre Jolly. Interactive organ segmentation using graph cuts. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2000*, pages 276–286. Springer, 2000.

[4] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124–1137, 2004.

[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 Marie-Pierre 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] Mark A Brown and Richard C Semelka. *MRI: basic principles and applications*. John Wiley & Sons, 2011.

[8] Laurent D Cohen and Isaac Cohen. Finite-element methods for active contour models and balloons for 2-d and 3-d images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(11):1131–1147, 1993.

[9] Piali Das, Olga Veksler, Vyacheslav Zavadsky, and Yuri Boykov. Semiautomatic segmentation with compact shape prior. *Image and Vision Computing*, 27(1):206–219, 2009.

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

[11] Ivana Despotović, Bart Goossens, and Wilfried Philips. Mri segmentation of the human brain: challenges, methods, and applications. *Computational and mathematical methods in medicine*, 2015, 2015.

[12] Alexandre X Falcão, Jayaram K Udupa, Supun Samarasekera, Shoba Sharma, Bruce Elliot Hirsch, and Roberto de A Lotufo. User-steered image segmentation paradigms: Live wire and live lane. *Graphical models and image processing*, 60(4):233–260, 1998.

[13] Lester Randolph Ford Jr and Delbert Ray Fulkerson. *Flows in networks*. Princeton university press, 2015.

[14] Andrew V Goldberg and Robert E Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM (JACM)*, 35(4):921–940, 1988.

[15] Nelly Gordillo, Eduard Montseny, and Pilar Sobrevilla. State of the art survey on mri brain tumor segmentation. *Magnetic resonance imaging*, 31(8):1426–1438, 2013.

[16] Lena Gorelick, Olga Veksler, Yuri Boykov, and Claudia Nieuwenhuis. Convexity shape prior for segmentation. In *Computer Vision–ECCV 2014*, pages 675–690. Springer, 2014.

[17] R. M. Haralick. A measure for circularity of digital figures. In *IEEE Transactions on Systems, Man and Cybernetics*, volume SMC-4, page 334336. IEEE, 1974.

[18] Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1992.

[19] Vida Harati, Rasoul Khayati, and Abdolreza Farzan. Fully automated tumor segmentation based on improved fuzzy connectedness algorithm in brain mr images. *Computers in biology and medicine*, 41(7):483–492, 2011.

[20] Sean Ho, Lizabeth Bullitt, and Guido Gerig. Level-set evolution with region competition: automatic 3-d segmentation of brain tumors. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 1, pages 532–535. IEEE, 2002.

[21] Joseph P Hornak. The basics of mri, 2008. *URL http://www. cis. rit. edu/htbooks/mri/index. html*, 68, 2008.

[22] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

[23] Ron Kikinis and Steve Pieper. 3d slicer as a tool for interactive brain tumor segmentation. In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, pages 6982–6984. IEEE, 2011.

[24] 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.

[25] Liqun Liu. Brain tumor segmentation with minimal user assitance. Master's thesis, University of Western Ontario, December 2013.

[26] Raul S Montero and Ernesto Bribiesca. State of the art of compactness and circularity measures. In *International Mathematical Forum*, volume 4, pages 1305–1335, 2009.

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

[28] Steve Pieper, Michael Halle, and Ron Kikinis. 3d slicer. In *Biomedical Imaging: Nano to Macro, 2004. IEEE International Symposium on*, pages 632–635. IEEE, 2004.

[29] Steve Pieper, Bill Lorensen, Will Schroeder, and Ron Kikinis. The na-mic kit: Itk, vtk, pipelines, grids and 3d slicer as an open platform for the medical image computing community. In *Biomedical Imaging: Nano to Macro, 2006. 3rd IEEE International Symposium on*, pages 698–701. IEEE, 2006.

[30] Marcel Prastawa, Elizabeth Bullitt, Sean Ho, and Guido Gerig. A brain tumor segmentation framework based on outlier detection. *Medical image analysis*, 8(3):275–283, 2004.

[31] Jan Rexilius, Horst K Hahn, Jan Klein, Markus G Lentschig, and Heinz-Otto Peitgen. Multispectral brain tumor segmentation based on histogram model adaptation. In *Medical Imaging*, pages 65140V–65140V. International Society for Optics and Photonics, 2007.

[32] 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.

[33] Jainy Sachdeva, Vinod Kumar, Indra Gupta, Niranjan Khandelwal, and Chirag Kamal Ahuja. A novel content-based active contour model for brain tumor segmentation. *Magnetic resonance imaging*, 30(5):694–715, 2012.

[34] Olga Veksler. *Efficient graph-based energy minimization methods in computer vision*. PhD thesis, Cornell University, 1999.

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

[36] Shijun Wang and Ronald M Summers. Machine learning and radiology. *Medical image analysis*, 16(5):933–951, 2012.

[37] Tao Wang, Irene Cheng, and Anup Basu. Fluid vector flow and applications in brain tumor segmentation. *Biomedical Engineering, IEEE Transactions on*, 56(3):781–789, 2009.

[38] Alan Yuille and Peter Hallinan. Active vision. chapter Deformable Templates, pages 21–38. MIT Press, Cambridge, MA, USA, 1993.

Curriculum Vitae

Name:	Danfeng Chen
Post-Secondary Education and Degrees:	University of Western Ontario London, ON, CA M.Sc. in computer science, 2014 - Present Wuhan University of Technology Wuhan, Hubei, China B.E. in Software Engineering, 2010 - 2014
Honours and Awards:	WGRS, Western University, 2014 -2015 Outstanding Graduate, Wuhan University of Technology, 2014
Related Work Experience:	Teaching Assistant, Western University, 2012 -2013 Research Assistant, Computer Vision Group, University of Western Ontario, Sep. 2014 - Mar. 2016