A graph-based method for segmentation of tumors and lymph nodes in volumetric PET images

Markus Lane Van Tol
University of Iowa

Follow this and additional works at: https://ir.uiowa.edu/etd
Part of the Electrical and Computer Engineering Commons

A GRAPH-BASED METHOD FOR SEGMENTATION OF TUMORS AND LYMPH NODES IN VOLUMETRIC PET IMAGES

by

Markus Lane Van Tol

A thesis submitted in partial fulfillment of the
requirements for the Master of Science
degree in Electrical and Computer Engineering
in the Graduate College of
The University of Iowa

December, 2014

Thesis supervisor: Associate Professor Reinhard R. Beichel

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

MASTER'S THESIS

This is to certify that the M.S. thesis of

Markus Lane Van Tol

has been approved by the Examining Committee
for the thesis requirement for the Master of Science
degree in Electrical and Computer Engineering at
the December, 2014 graduation.

Thesis Committee: ______________________________
Reinhard R. Beichel, Thesis Supervisor

____
John Buatti

____
Milan Sonka

ACKNOWLEDGMENTS

Thank you to Dr. Reinhard R. Beichel for being my advisor for this whole process. Thank you to Dr. John Buatti and Dr. Milan Sonka for serving on my defense committee. Thank you to Kristin Plichta and Tangel Chang for assisting in the experiment. Thank you to Dr. Christian Bauer for helping during development.

ABSTRACT

For radiation treatment of cancer and image-based quantitative assessment of treatment response, target structures like tumors and lymph nodes need to be segmented. In current clinical practice, this is done manually, which is time consuming and error-prone. To address this issue, a semi-automated graph-based segmentation approach was developed.

It was validated with 60 real datasets, segmented by two users manually and with this new algorithm, and 44 scans of a phantom dataset. The results showed a statistically significant improvement in intra- and interoperator consistency of segmentations, a statistically significant improvement in speed of segmentation, and reasonable accuracy against consensus images and phantoms. As such, the algorithm can be applied in cases that otherwise would use manual segmentation.

TABLE OF CONTENTS

LIST OF TABLES

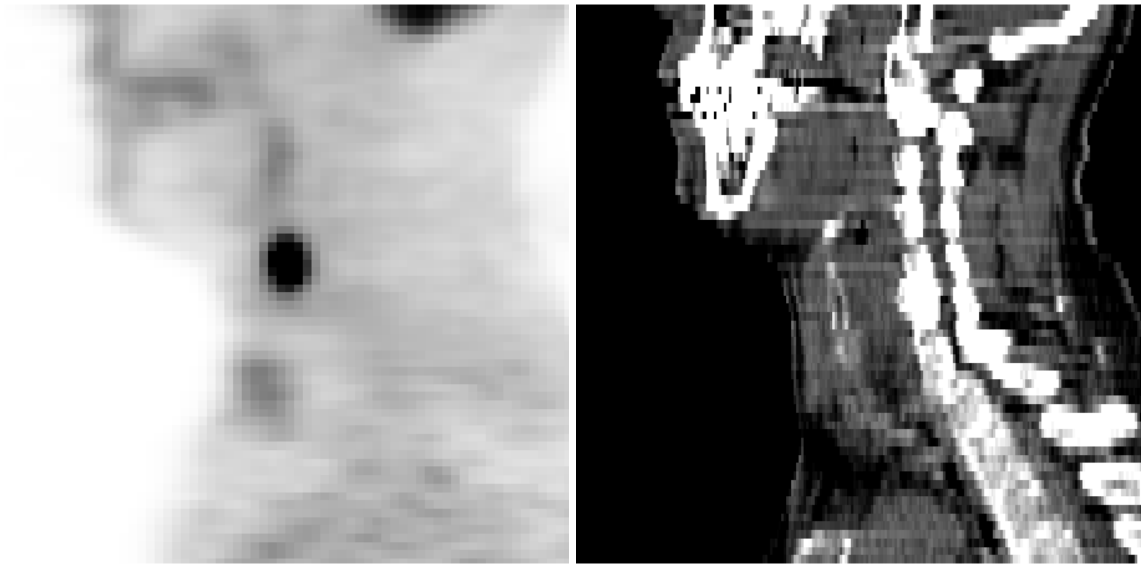
Table

LIST OF FIGURES

Figure

CHAPTER 1
INTRODUCTION

In ^{18}F-fluoro-deoxy-glucose-positron emission tomography (^{18}F-FDG PET or just FDG PET) functional imaging allows physicians to detect cancer and assess the activity of cancer cells, including in the head and neck. These cells show up much better in FDG PET images than in computed tomography (CT) images, as seen in Figure 1.1, making this imaging modality well suited for measurement of cancerous tumors and lymph nodes.



(a) (b)

Figure 1.1: A comparison of two imaging modalities for viewing a tumor. (a) The FDG PET image shows the tumor clearly. (b) The CT image fails to show the same tumor well.

With consistent measurement of the tumors and nodes, these FDG PET images could be used to better predict outcomes of treatments by measuring quantitative indices in pre- and post-treatment volumes of tumors and nodes. Consistent measurement of those tumors and nodes, however, requires consistent delineation of the boundaries of those tumors and nodes, which is not trivial. Unfortunately, FDG PET images are relatively low resolution, generally around $3 \times 3 \times 3$ mm per voxel, sometimes worse, compared to computed tomography (CT) images at around $0.5 \times 0.5 \times 0.6$ mm. They also tend to be noisy. Both of these make recognizing the actual boundaries of lesions difficult. At the present time, it is generally done manually, which introduces issues with reproducibility and time, particularly when considering large datasets as would be required for response assessment in oncology clinical trials. This thesis introduces a method to more consistently and rapidly delineate head and neck lesions in FDG PET volumetric images while still making segmentations amenable to manual adjustment based on clinically-based knowledge and hence comparable to those made manually.

1.1 Motivation

As mentioned, FDG PET images give useful information for locating and assessing the activity of cancerous lesions. These are important for quantitative analysis for research purposes as well as for treatment. Various methods have been used to measure different quantitative indices from these scans, such as the maximum uptake in an object and the volume of an object. While the maximum uptake often requires only a rough segmentation, the volume requires more specific boundaries. Measurements such as the average and accumulated uptake could also be of use, along with other, not-yet-devised measures, which would require segmenting the volume. For radiation treatment planning, tumor delineation is required to calculate a treatment plan, and is currently almost exclusively manual in practice.

While manually drawing an object is currently the state-of-the-art, it has trouble with consistency and speed. One user making two segmentation of the same object with the same information at different times will usually make two different segmentations, as seen in Figure 1.2. Even with more specific instructions, the exact boundaries can vary noticeably, adding noise to the derived quantitative indices and hindering analysis. Once more than one user is making a segmentation of an object, the differences tend to become even greater.

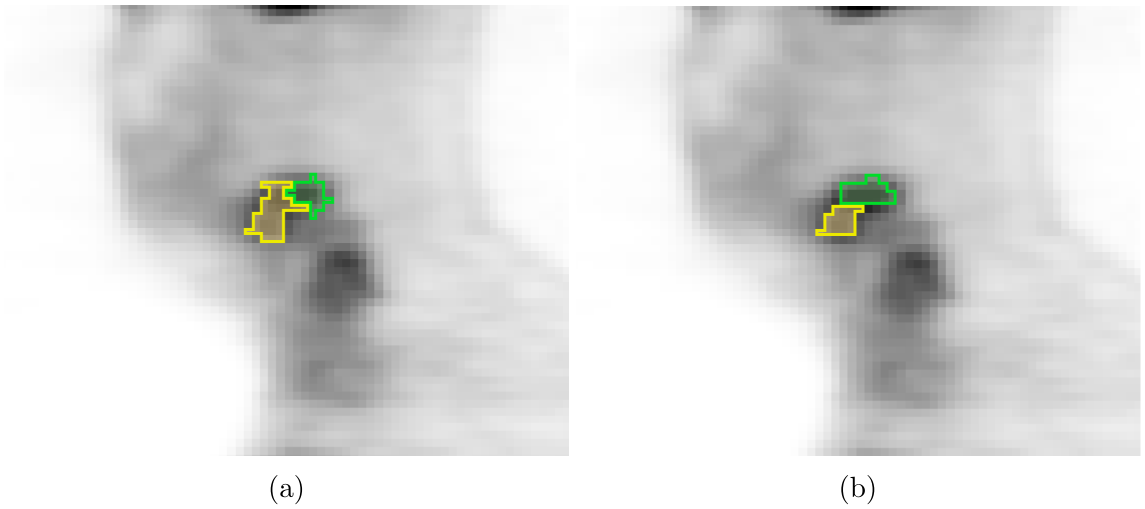
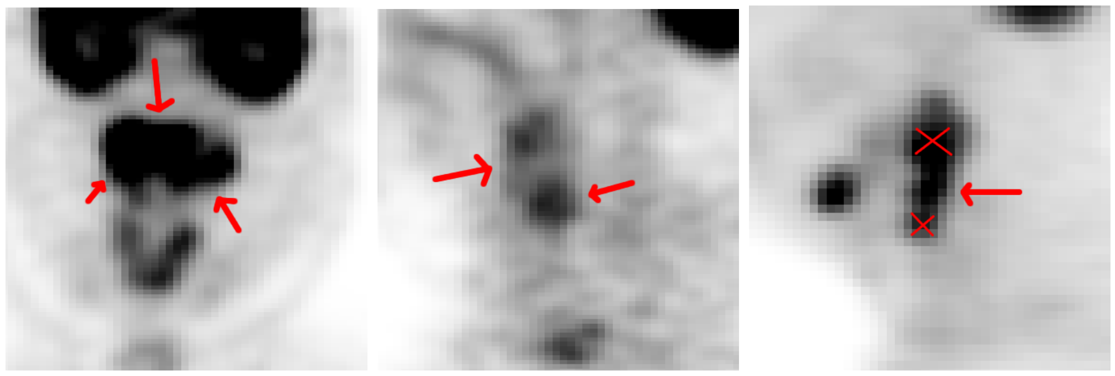


Figure 1.2: Two manual segmentations of a pair of lymph nodes by the same user. (a) The yellow-labeled object is segmented to stretch up vertically. (b) The same yellow-labeled object is segmented to cut off with the green-labeled object above it instead.

Though manual is generally the most common FDG PET segmentation approach, there are more methods available. Many can be fully automated, removing user interactions entirely and may even be perfectly consistent. Unfortunately, removing the user interaction and judgment entirely removes an important check on the results. These methods can have success in simple cases, but it's nearly impossible to make a completely automated segmentation tool that can handle all complex cases, some of which are depicted in Figure 1.3. In particular, Figure 1.3a and Figure 1.3b show cases with opposite solutions, which would look essentially the same to an algorithm. Semiautomated methods can allow for more interaction to accomplish more complex segmentations, at the expense of increased user variation, and still without necessarily being able to handle all the cases.



(a) (b) (c)

Figure 1.3: (a) A lesion with inhomogeneous tracer uptake. (b) A lesion has faint and seemingly disjoint parts. (c) A lesion is in close proximity to neighboring lesions.

So, there is a desire for a method that has simple interaction to get good consistency, but with built in functionality to handle tougher cases that should allow the user to efficiently control the boundary.

1.2 PET Segmentation Methods

As mentioned before, the most common method is simply manual segmentation, which is generally used in lieu of a real ground truth segmentation. Its strengths include ability to take full advantage of the medical knowledge and experience of the user, complete freedom to segment as needed without any algorithmic requirements getting in the way, and ease of separately segmenting individual lesions. Its weaknesses include issues with inconsistency, which limits experimental reproduceability, and time, which brings a financial expense. Furthermore, occasional jagged edges occur due to the one-slice-at-a-time approach and the limitations of a user's manual dexterity, which can be inherently at odds with the knowledge base.

Another common approach is thresholding, common in many comparative studies such as Drever's in 2007 [5], Sheperd's in 2012 [15], and Zaidi's in 2012 [19], wherein the voxels connected to the lesion target voxel above a certain uptake are all counted as the lesion. The lesions in a PET image show up as higher-uptake regions, so a threshold to separate the higher uptake lesion from the lower uptake background is a sensible way to choose the boundaries in most cases. Thresholding requires a good choice of base threshold value, and the ways to choose it are varied, from a constant standard uptake value threshold of 2.5, to percentages such as 40% or 50% of the maximum uptake, to more complex methods incorporating the signal-to-background ratio (seen in Daisne's work in 2003 [4] and Schinagl's work in 2007 [14]). One of the methods described in Sheperd's study is region growing in a sphere centered on a user point, in which the extend of the region growth is just those connected to the center above a user-defined threshold limited by a sphere, with similar overall results, though more variability in results. Thresholding gives good consistency, though less so with a selected threshold, but it leads to inability to separate out nearby uptake lesions without extension to the algorithm (shown in Figure 1.4a), inability to handle semi-necrotic regions effectively (shown in Figure 1.4b), and lack of freedom with regards to handling more complicated cases.

Classical region growing from Adams' work in 1994 [1] can be used, with specific growth criteria as in Li's in 2008 [10] or in the region growing algorithm included as part of Sheperd's 2012 study [15]. While the variant in Sheperd's study used a user-chosen threshold as its criterion for region growing, Li's variant used more adaptive criteria. This method used an automatically expanding threshold around the seed. The expansion was limited to the point when the total volume shot up rapidly, which indicated leakage into the background. This adaptive method still has many of the issues of thresholds, however, including the issue of ignoring high-uptake lesion boundaries.

Watershed segmentation, with the standard flood method (on an inverted image) and with minima markers as in Rivest's 1992 work [13], can also be used for

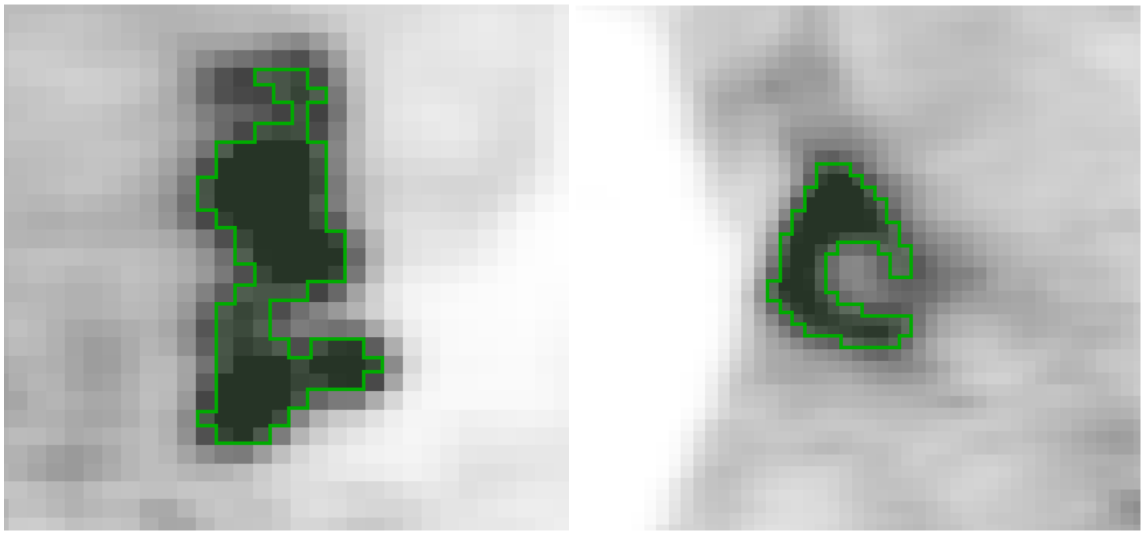


Figure 1.4: Complex cases that pure threshold-based segmentations fail to cover. (a) Thresholding fails to separate nearby lesions when the separation boundary is fairly high uptake. (b) Thresholding fails to include necrotic parts of a lesion.

segmenting lesions. The watershed with markers focuses on oversegmenting the lesions in the scan and then merging section using various user-defined markers as in Lefvre's 2007 work [9] for inclusion or exclusion, with some more complicated use of the spatial location of the markers. Watershed segmentation has the strengths of being able to easily separate lesions, even with high-uptake boundaries that threshold segmentations can miss. The built-in user interaction from markers gives users the ability to modify specific points, which can be good for complex segmentation but bad for consistency. Its weaknesses are mainly that watershed basins boundaries often fail to match up with the lesion boundaries, and that the basin boundaries themselves are also very susceptible to noise. The first weakness, the boundary mismatch, requires additional adaptation of the overall algorithm. The second weakness, noise vulnerability, requires significant de-noising to make watersheds effective, or making the "fluid" of the watershed "viscous", as in Vachier's 2005 version [17], to smooth out edges made rough by noise.

Methods based on fuzzy clustering or fuzzy C-means are also useable for segmentation, as in Belhassen's 2010 study [3] and as applied to head and neck cancers in Zaidi's in 2012 [19]. In this case, a voxel can be associated with multiple classes with probabilities based on uptake, though it can be expanded as demonstrated in Belhassen's work to include spatial information, reducing the sensitivity to noise. These algorithms are used to split up C different objects on a scan. Another variant in Belhassen's work uses the spatial wavelet transform to better deal with lesions with inhomogeneous internal uptake. While effective in segmentation, the segmentations these methods made were not generally of different lesions, but rather of different "layers" of uptake levels. So, it doesn't quite do what is desired here.

Stochastic modeling segmentation was used in Aristophanous's work in 2007 [2] and adapted for FDG PET head and neck tumors in Zaidi's 2012 study [19]. Here, a Gaussian mixture model is used to assign a probability distribution to voxels of belonging to individual classes, based on uptake. The approach's focus on reducing to binary classification and it's failure to include spatial information, however, makes it not ideal for dealing with segmentation of multiple different high-uptake lesions.

Edge detection methods, such as that described in Geets's 2007 study [6] and some methods used in general studies, such as the Sobel operator in Drever's 2007 study [5], can be used to segment FDG PET lesions. For edge detection in general, the algorithms are generally good at cutting out individual lesions from other nearby lesions with proper configuration. However, they don't tend to naturally have the adaptability for a combination of inhomogeneous lesions with faint edges within them and clusters of lesions with similarly faint edges between them, as can occur in the lesions sought here. The particular segmentation method in Geets' study utilizes watersheds for gradient crest detection and cluster analysis to fuse those watersheds.

It shows some of the drawbacks of edge detection, requiring denoising with an edge-preserving filter and with the same limitations regarding inhomogeneous lesions and clustered lesions.

One general idea for methods is to use a series of different algorithms to cover the weaknesses of individual parts, forming ”pipeline” methods, such as that of Kuhnigk’s work in 2006 [8], which was adapted for use in FDG PET images in Sheperd’s 2012 study [15]. Pipeline methods in general can use the various methods to cover each others’ weaknesses, but the more parts are included to improve the result, the slower the overall algorithm.

Further methods such as Han’s from 2011 [7] can use PET and CT data together. While useful, it can bring registration issues depending on the circumstances of the scans. Once that is handled, though, the higher resolution and contrast available in CT combined with the information unique to PET can allow for some more interesting methods, such as that of Han’s, which uses a Markov Random Field to perform a graphical segmentation, a method shown in in Li’s 1994 work [12].

1.3 Goal

Ultimately, the goal is a segmentation method that is faster and more consistent than manual segmentation, while still able to successfully segment all lesions. It must handle all the variations of lesion in a way that is as consistent as possible while still allowing for specific user interaction for changes. It has to be flexible enough for all circumstances of FDG PET lesion, but be consistent in boundaries for more normal cases.

As such, the goal is to make a minimally-interactive semi-automated tool, with well-defined options and modes in its application and an effective default procedure for use, allowing for the speed and consistency of algorithmic methods and the accuracy and flexibility of manual or largely manual methods.

CHAPTER 2
METHODS

A semi-automated tool has been developed to improve consistency and speed of segmentation with little loss of accuracy. This tool uses optimal surface finding, published in Li's 2006 paper [11], in order to convert the image segmentation problem into an efficiently-solvable graph optimization problem. This graph optimization finds a surface that minimizes a cost function for each node, with requirements for smoothness. The cost function is determined using a ratio of a localized background calculation to an internal peak calculation. Additional modes also account for minima and watersheds that might make up edge boundaries. After the base result, refinement options allow the user to move parts of the final surface out to specific features or to modify the base cost function. With this, the semi-automated tool can convert a few actions into a more consistent and predictable segmentation that can form a complete lesion across inhomogeneous parts or segment individual lesions near other lesions, with specific user control over the segmentation that still works in an algorithmic fashion, in only a few seconds. Additionally, a few adaptations can be applied to deal with very specific situations, such as highly noisy lesions or different reconstruction kernels.

2.1 Optimal Surface Finding

The optimal surface finding algorithm is a method described in Li's paper in 2006 [11] for converting image segmentation problems into graph optimization problems which can be solved efficiently. A brief description and explanation are provided here.

Organize a grid of non-intersecting columns through the region where the boundary should be placed (Figure 2.1a). Place graph nodes along these columns regularly. The columns and the nodes thereon should be as dense as the granularity of the boundary needs to be; each node is a possible location for the boundary. For each node, a cost is be assigned. Within constraints, the graph optimization selects a minimum cost set of nodes, one from each column for the surface.

In the graph, set these nodes based on the cost function in Section 2.2.2 and Section 2.2.3 (Figure 2.1b). Also, add an edge from each node to the next node downward with infinite capacity. Minimum cut will use this to get a minimum set of these nodes that includes, at its highest edge on each column, the minimum set of individual boundary nodes.

Now, enforce smoothness by adding infinite-capacity edges downward between columns, decreasing in node by at most the limit of difference (Figure 2.1c). Additionally, as described in Sun in 2013 [16], add low-capacity bidirectional edges laterally to the same node on each adjacent column. This gives a soft smoothness constraint, which works as a penalty for changing distance outward across columns.

Lastly, for the nodes with difference values above 0, add infinite capacity edges from the source. For nodes with difference values at or below 0, add infinite capacity edges to the sink. Now the graph can be solved to find the minimum cost boundary based on the original node costs. Consider the parts "below" the boundary to be the object and segment the corresponding region as such (Figure 2.1d).

The resulting segmentation is an optimal solution to the costs and constraints of the nodes that results in a fast and efficient segmentation of the region.

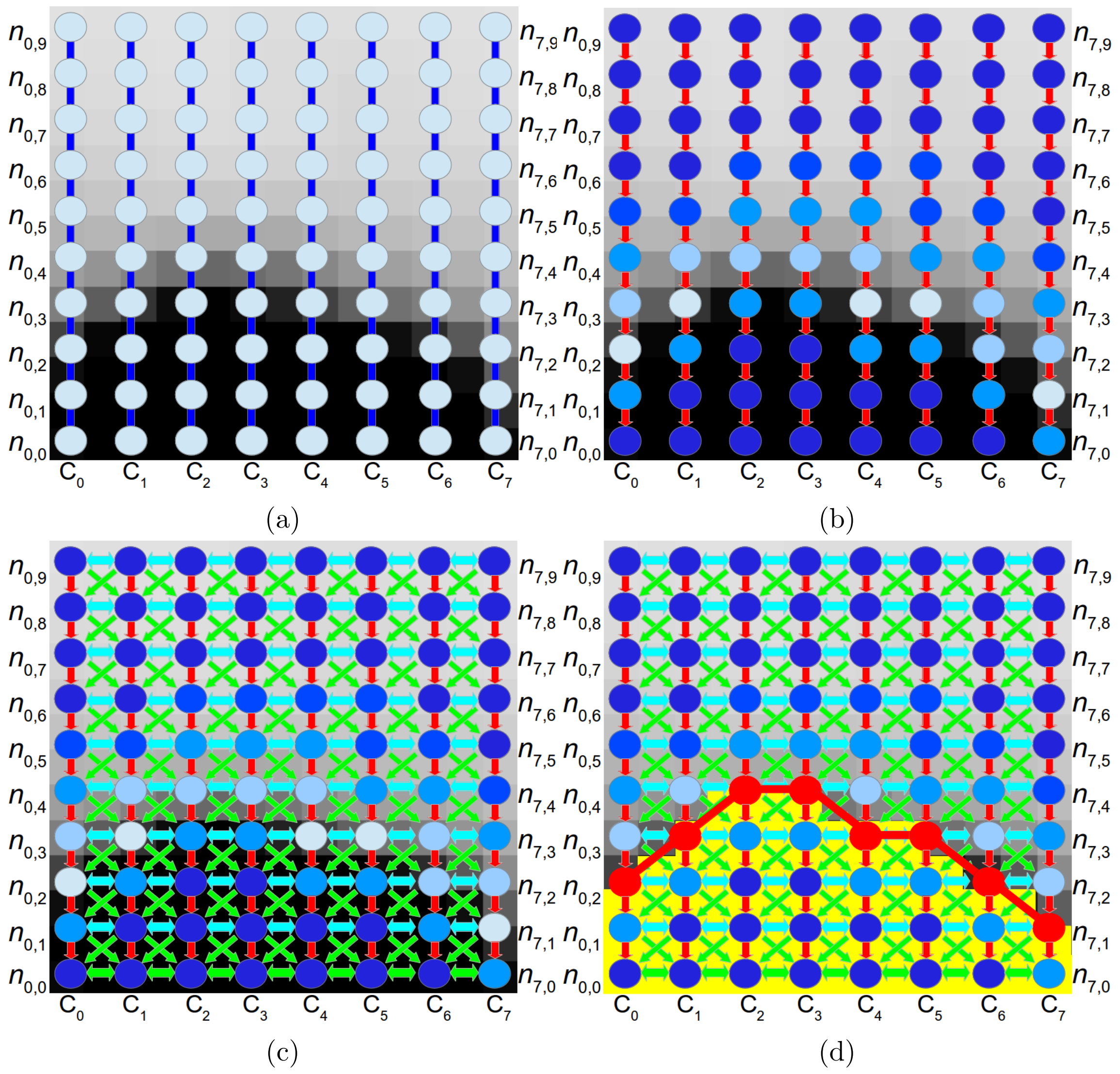

$n_{0,9}$ $n_{0,8}$ $n_{0,7}$ $n_{0,6}$ $n_{0,5}$ $n_{0,4}$ $n_{0,3}$ $n_{0,2}$ $n_{0,1}$ $n_{0,0}$
$n_{7,9}$ $n_{7,8}$ $n_{7,7}$ $n_{7,6}$ $n_{7,5}$ $n_{7,4}$ $n_{7,3}$ $n_{7,2}$ $n_{7,1}$ $n_{7,0}$
C_0 C_1 C_2 C_3 C_4 C_5 C_6 C_7
(a)
(b)
(c)
(d)

Figure 2.1: The steps for optimal surface finding, visualized. (a) Columns are placed across the boundary region, and nodes are added at regular intervals. (b) Add edges (red) along the columns with infinite capacity, going toward the interior. Determine the cost of the nodes (darker blue for greater cost). (c) Add edges between the columns in order to enforce smoothness, some downward with infinite capacity (green) for the hard constraint some laterally with minor capacity (cyan) for the soft constraint. (d) Convert the node costs into differences. Add edges with the source and sink and solve the maximum flow of the graph. The minimum cut of nodes is the set within the segmentation. The outer boundary of it (red) is the surface selected. Voxelize that to get an object label for every voxel (yellow).

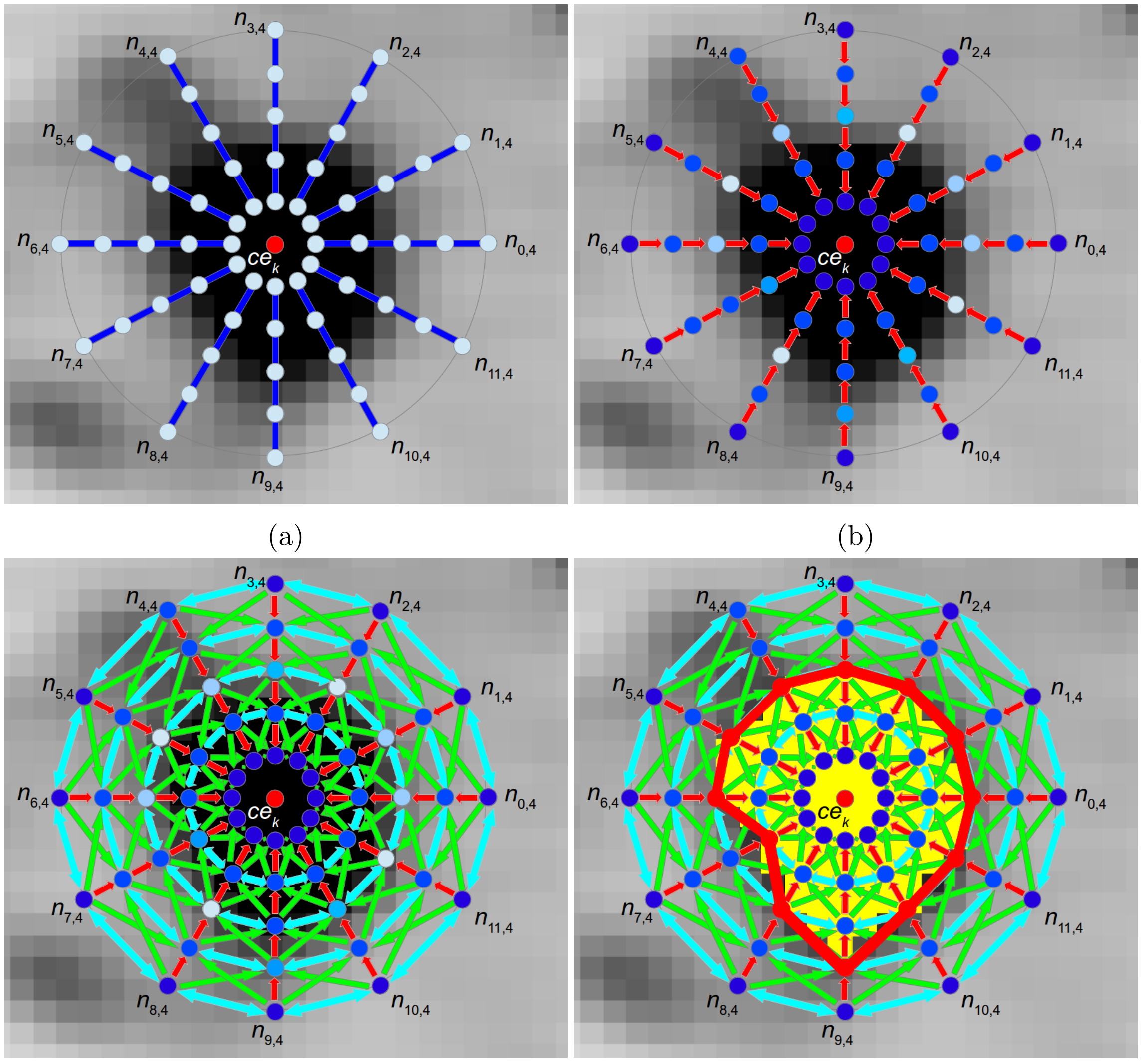
2.2 Main Method

As stated, the approach overall is to convert an image segmentation problem into a graph optimization problem that can be solved fast enough for an interactive tool, using the framework described above. A spherical mesh forms the basis for a graph, with the segments from the center of the sphere to each of the vertices making up the columns for the boundary locations. The nodes along the column are assigned a cost based on a threshold and some other factors related to the contents of the column. Figure 2.2 shows the basic construction adapated to a spherical shape (or circular, in the two-dimensional representation here).

The exact position of the boundaries of the final mesh are set to optimize a cost based on uptake at the point of the boundary, uptake in the entire region of the sphere, and surface smoothness. Since this can all be determined automatically, the method's main functionality can run with nothing but a single center point on an image.

The main method is organized into three parts: graph setup to generate the basic columns and edges (Section 2.2.1), cost application (Section 2.2.2 and Section 2.2.3), and solving (Section 2.2.4). In graph construction, the object is identified and the overall shape of the graph is assembled. In cost setting, the cost functions for nodes along the graph are set. During solving, a few more edges are added to the graph, the minimum cut is solved, and the boundary is turned into a label volume, with some cleanup. This requires at the very least the uptake image and a point chosen by the user to indicate the object.

Beyond the main method, there is also refinement, an optional step which occurs after solving. The changes from this may require solving the graph again. This will be explained in more detail in Section 2.3.



(a) (b)

(c) (d)

Figure 2.2: The spherical optimal surface finding algorithm. (a) Place columns toward the vertices of a spherical mesh and add the nodes. (b) Add intracolumn edges (red) toward the center with infinite capacity and determine the cost for the nodes (darker blue for greater cost). (c) Add intercolumn edges columns; green with infinite capacity and cyan with low capacity. (d) Convert the node costs into differences with the source and sink and solve the maximum flow. The red boundary is the border, the yellow section is the final voxelized segmentation.

2.2.1 Graph Setup

Setting up the graph involves placing the center point, adding the node locations, and adding the inter- and intra-column edges with appropriate capacities.

For lesion k, the center point ce_k will be the location for the center of the spherical graph. Identifying a lesion is the work of the user. The point chosen by the user, $ce_{k_{user}}$, can either be used directly as ce_k or can be used along with the original PET image I to determine a center point, to improve consistency of the method with variable user input.

In a process called recentering, the highest uptake voxel within a search radius of $sr = 7$ mm of the original point $ce_{k_{user}}$ is used as the new value of the center. This reduces some of the variability from slightly different voxel choices, but can cause problems when segmenting lesions with necrotic centers, which are centered around low-uptake voxels, or when segmenting lesions of a smaller radius than sr that are in close proximity to higher-uptake lesions. As such, it must be left as an option, though it should usually be active.

The graph $G_k = (V, E)$ itself is made by starting with a spherical mesh of $n_{column} = 1026$ evenly-spaced mesh vertices centered around the final center location. Each column has 60 evenly-spaced nodes that represent their corresponding physical location. The radius for this sphere is $r = 60.0$ mm with a gap of $gap = 1.0$ mm between nodes. Node 0 on column i, called $n_{i,0}$, is 1 mm from the center, while node $n_{i,n_{node}-1}$ is 60 mm from the center, and remaining nodes are placed in order between them. These nodes, all representing physical locations, are the vertices of G_k, source and sink aside.

Intracolumn edges are added to E, down along each column toward the centermost node. For every node $n_{i,j}$ where $j > 0$, an edge is added from $n_{i,j}$ to $n_{i,j-1}$ with infinite capacity. These allows selection of nodes along each column for the final segmentation.

Intercolumn edges need to be added to the graph in order to create the smoothness constraints, as explained previously. The hard smoothness constraint of $sc = 5$ prevents two adjacent columns in the solution from being more than sc nodes apart. The soft smoothness constraint of $sp = 0.005$ which adds a cost of sp for every node of separation between two adjacent columns' solutions.

For every node j and j', where $j' = \max(j - sc, 0)$ on every pair of adjacent columns i and i':

1. Add the edge $ed_1 = \{n_{i,j}, n_{i',j'}\}$ with infinite capacity to E, making one half of the hard smoothness constraint.

2. Add the edge $ed_2 = \{n_{i',j}, n_{i,j'}\}$ with infinite capacity to E, making the other half of the hard smoothness constraint. Now, when solved, the nodes of two adjacent columns cannot be more than sc nodes apart.

3. Add the edges $ed_3 = \{n_{i,j}, n_{i',j}\}$ and $ed_4 = \{n_{i',j}, n_{i,j}\}$ with capacity $sp = 0.005$ to E, making the soft smoothness constraint. Now, for a given solution surface, the cost is increased by sp for every node apart two adjacent columns' nodes are.

4. Add the edge $ed_5 = \{n_{i,j}, n_{i',j-1}\}$ with infinite capacity to E, if $j \neq 0$, allowing for selection on the column.

There will be additional edges to include in E, as well as the source node s and the sink node t as required for solving the graph. The cost function $c(i, j)$ must be calculated in order to add these remaining few edges and solve the graph.

2.2.2 Cost Setting

Setting the costs for the nodes is done by evaluating the uptake at the nodes and the local circumstances compared to other nodes on the column, especially those closer to the center of the mesh. The cost at a node is primarily determined by deviation from a threshold.

Within the constraints set by the graph in Section 2.2.1, the cost for each node $n_{i,j}$ determines the surface of the segmented shape, minimizing the sum cost of the selected surface nodes while maintaining the smoothness. The cost function $c(i, j)$ has many considerations, such as the minimum surface size, at what point on a column the object has clearly ended, and what is the ideal location of a border. The cost function relies heavily on a threshold Th which can be calculated automatically or set directly via refinement (see Section 2.3.1). The automatic calculation of Th is complex enough, however, that it is explained separately, in Section 2.2.3. This section will focus on the overall cost function, leaving Th as a variable.

The base cost function, $c_{base}(i, j)$, is strictly a function of the linearly-interpolated uptake at a node, $up(i, j)$. The function is built based on the threshold Th, the value at the center of the mesh, up_{ce}, and the uptake voxels in the region. Ideally, below the threshold, the cost approximately reflects the likeliness of the uptake to be part of the background. Above the threshold, the cost is a linear function with 0 at the threshold uptake and 1 at the center uptake. Assuming that the center is not the boundary and not in a necrotic region (wherein the uptake is below that of the rest of the lesion, or even below that of the background), the uptakes closer to that of the center should be less likely than those closer to that of the threshold. If the center is indeed below the threshold, though, the section of the cost for uptake over the threshold is just equal to 1.

For uptakes below Th, the cost is essentially the likehood of the uptake to be part of the background, modeled by a histogram of the overall spherical region. Restricting to the the radius of the graph, $r = 60.0$ mm of the chosen center point ce_k, produces have a limited set of the voxels from the original volume I. However, just counting the voxels in this volume in I will have a bias, because the dimensions

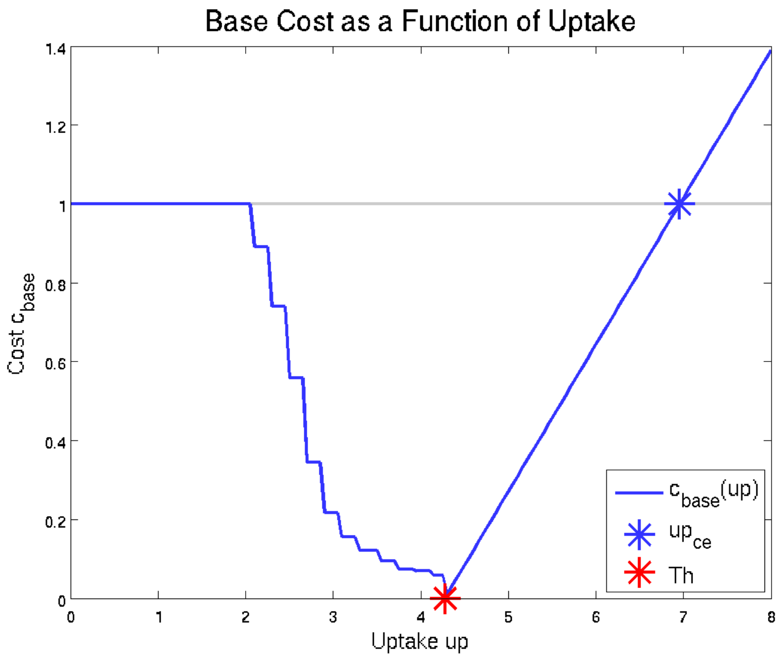


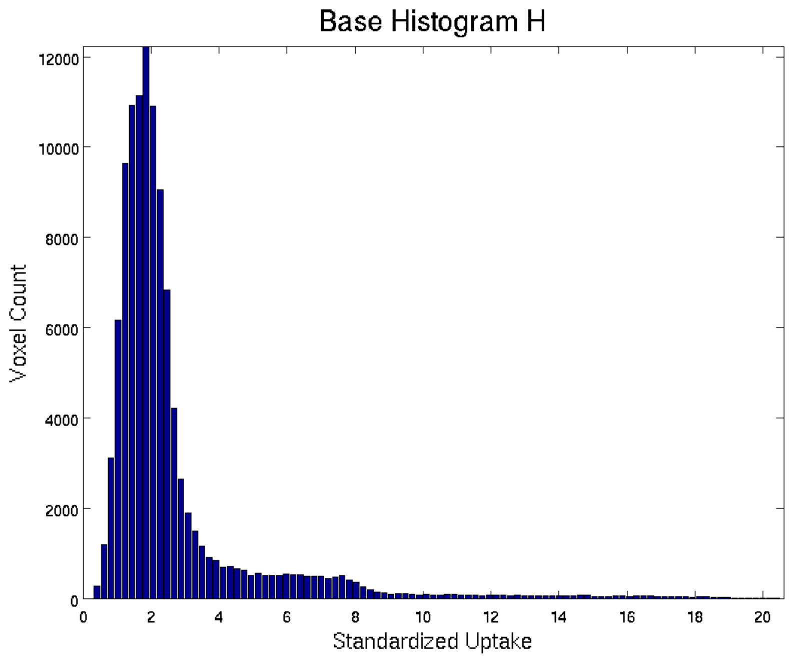
Figure 2.3: The cost as a function of threshold. The lower uptake side follows H' while the upper side is linear. The lowest point where they meet is at Th.

of I are frequently anisotropic. A resampled image called I' is made, isotropic in all dimensions equal to the densest of the original dimensions in I.

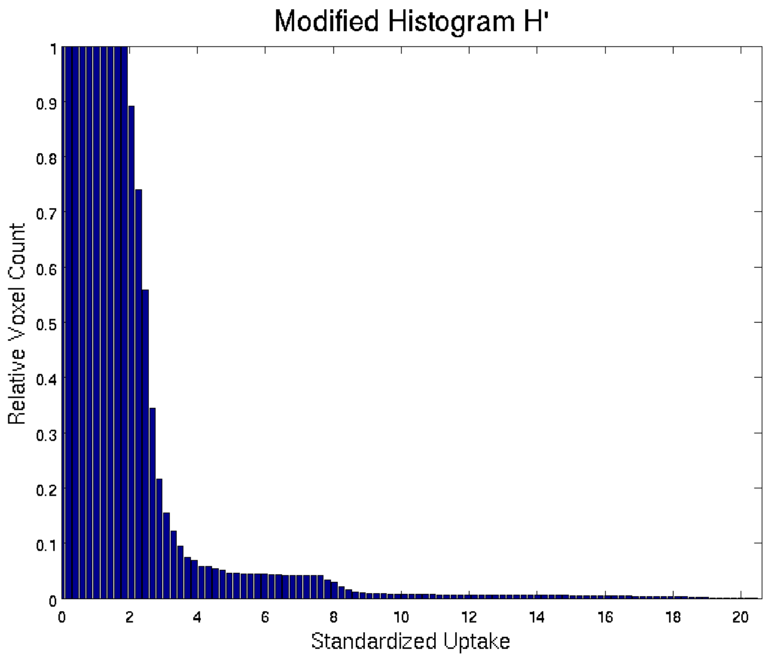
All voxels in I' within r mm of the center point ce_k are included in a 100-bin histogram, H, an example of which is in Figure 2.4a. For H, the lowest bin (at $b = 0$) includes the uptake 0 and the highest bin (at $b = 99$) includes the highest value in the region. The highest count in H is H_{max}. After calculating this histogram, a normalized, extended copy is made, H', an example of which is in Figure 2.4b. The idea is that the histogram monotonically decreases from low uptake to high. H' is calculated in the following way:

$$H'(b) = \begin{cases} H(b)/H_{max}, & \text{if } b = 99 \\ \max(H'(b+1), H(b)/H_{max}), & \text{otherwise.} \end{cases} \tag{2.1}$$

With that, the base cost, $c_{base}(i,j)$, can be determined:



(a)



(b)

Figure 2.4: Histograms of the region around a typical lesion. (a) The original histogram. (b) The modified histogram now forms the profile of the lower side of the cost function.

$$c_{base}(i,j) = \begin{cases} H'(\lfloor \min(b_{max}, (b_{max}+1) * \frac{up(i,j)}{H'_{max}}) \rfloor), & \text{if } up(i,j) < Th \\ 0, & \text{if } up(i,j) = Th \\ \frac{up(i,j)-Th}{up_{ce}-Th}, & \text{if } up(i,j) > Th \\ & \text{ and } up_{ce} > Th \\ 1, & \text{otherwise.} \end{cases} \tag{2.2}$$

The structure of the cost function function naturally gives it a scale from 0 to 1 for the relevant levels of uptake. Some cases where a lesion must connect smoothly to an adjacent, higher-uptake lesion may have costs for the intended region above 1, but they are generally handled by the label avoidance mode described in Section 2.4.1.

For a typical object, the value up_c is higher than most of the values on a column, or at the very least higher than any of the uptakes on the column that would be the boundary. So, values below it are within a 0 to 1 scaling for cost.

As the uptake goes from at the threshold to below, there is a slight gap in cost as it increases up to the equivalent histogram scale, especially lower on the histogram. The effect of this is to make the overall function act more like a real threshold, which segments values only above the threshold, without making it too forceful. An example of the resulting cost profile of this function is in Figure 2.5.

After determining c_{base}, there are a few other changes to make to the cost. A few nodes are ”rejected” due to their unlikeliness, avoiding trivial solutions or clearly disconnected objects. These nodes are rejected by increasing their cost by $rej = 6$, far above the typical scaling. This is applied at close nodes (below $j_{min} = 3$), as well as nodes at and beyond where the uptake has fallen below the median of the 60 mm region around, equal to M_{region}. Nodes below this value or beyond such nodes are affected by the low rejection condition, rc_{low}, at a certain column and node:

$$rc_{low}(i,j) = j > j_{min} \text{ and } \min_{j'=0,1,\ldots,j}(up(i,j')) < M_{region}. \tag{2.3}$$

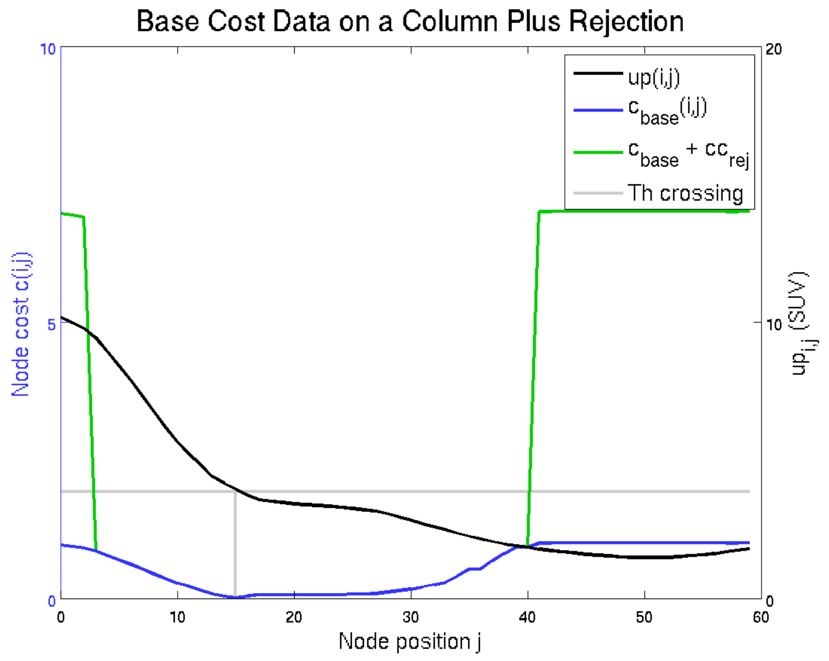


Figure 2.5: A typical cost profile on a single column i, as a function of j, paired with the uptake along the column and markings for the threshold. Shown with and without cc_{reject}.

The first node beyond the close rejection, j_{min}, is not rejected this way in order to avoid cases in which all nodes are rejected, which makes the rejection useless. The condition rc_{low} and the close node limit can be used to put together an equation for the cost change due to rejection, $cc_{reject}(i,j)$:

$$cc_{reject}(i,j) = \begin{cases} rej, & \text{if } j < j_{min} \text{ or } rc_{low}(i,j) = \text{True} \\ 0, & \text{otherwise.} \end{cases} \tag{2.4}$$

This term cc_{reject} is added to c_{base} to make the final cost c, though additional cc terms may be added in other modes or with other refinement. Figure 2.5 shows the final costs from rejection along with the base cost.

2.2.3 Threshold Calculation

The threshold Th is the primary determining factor for the final boundary of the segmentation. It's needed to determine the cost $c(i, j)$ at column i and node j. It is ultimately based on a high-uptake value within the lesion and a low-uptake value outside of the lesion, both of which are calculated automatically from the base point.

Th must be above the background value and below the highest uptake of the object, but there is more to it than just that, something that percentages from the background to the maximum cannot capture. On some lesions, there is a high difference between the lesion and background uptakes, and a fairly high ideal threshold with it, but not proportionally higher with the lesion uptake, which could lead to loss of some of the lesion. On others, the lesion uptake is very close to the background value. A proper approach to the threshold calculation has to work for both of these. A ratio-based technique such as that of Daisne's from 2003 [4] is preferred to the simple percentages, but the method described here uses only a singular point, identifying the object itself. As such, the single point must determine an upper object value and a lower non-object or end-of-object value to use the ratio of, which is not a straightforward process.

While not always accurate, it's generally useful to roughly approximate an uptake object as if it were spherical, which gives a simpler object to recognize the lesion and background uptakes of. This is done using "shells". A shell is a set of all nodes at a certain distance from the center point. That is, a shell sh_j consists of all nodes $n_{i,j}$ for all columns i in the spherical mesh, as seen in Figure 2.6.

$up(i, j)$ is the linearly interpolated uptake at the physical location of the node $n_{i,j}$. The uptake shell $up_{sh}(j)$ is the set of uptakes for the corresponding nodes in sh_j:

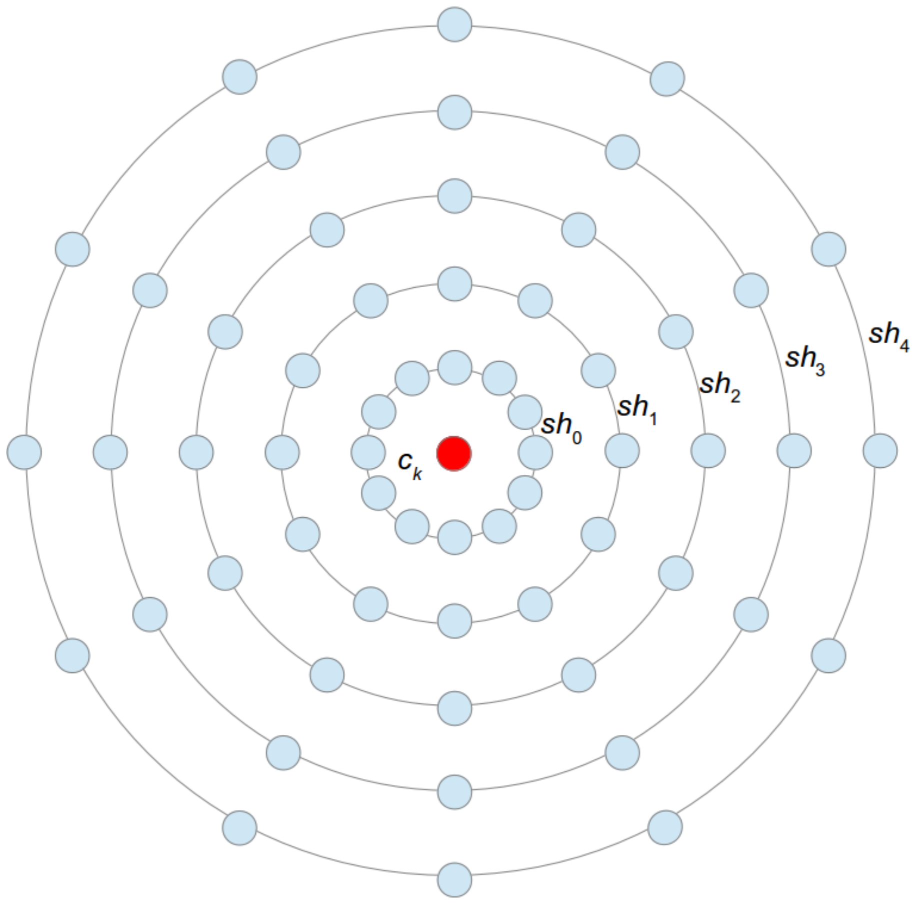


Figure 2.6: Sets of nodes arranged by their shells. All nodes a certain distance from ce_k are in the same shell.

$$up_{sh}(j) = \{up(0,j), up(1,j), up(2,j), ..., up(n_{column} - 1, j)\}. \qquad (2.5)$$

With the individual uptake shells, a single value for each j must be settled on to create a profile of the uptake across the entire region, $up_{profile}$:

$$up_{profile}(j) = \text{med}(up_{shell}(j)). \qquad (2.6)$$

This result, $up_{profile}$, is a simplified view of lesion. It can be used to make some estimations, shown in Figure 2.7. The threshold Th is a function of two values

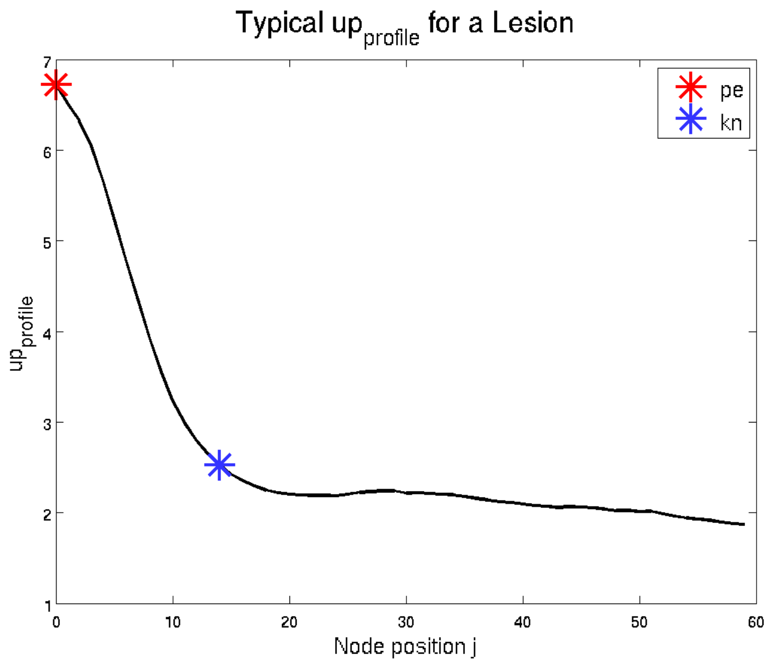


Figure 2.7: An approximate profile made of shell medians. Marked are two specific values to be utilized, pe and kn.

found in this profile: the "peak" pe and the "knee" kn. The peak pe is a measure of the approximate maximum intensity of the lesion being segmented. The knee kn is a measure of the approximate point at which the object fades into the background, but is not intended to be the background itself. pe is straightforward to determine:

$$pe = \max_{j=0,\ldots,n_{node}-1}(up_{profile}(j)). \tag{2.7}$$

kn is more complicated. The intent of this is to find a point where the profile changes from falling rapidly into the background to just being the background, which gives us a fairly narrow target that stays relatively consistent across the objects sought in FDG PET images.

The knee is between the steepest descent inward and the shallowest descent outward. Both of those are needed, and to find those robustly, the gradient of the profile must be calculated, which is shown in Figure 2.8. This is $up'_{profile}$:

$$up'_{profile}(j) = up_{profile}(j+1) - up_{profile}(j-1). \tag{2.8}$$

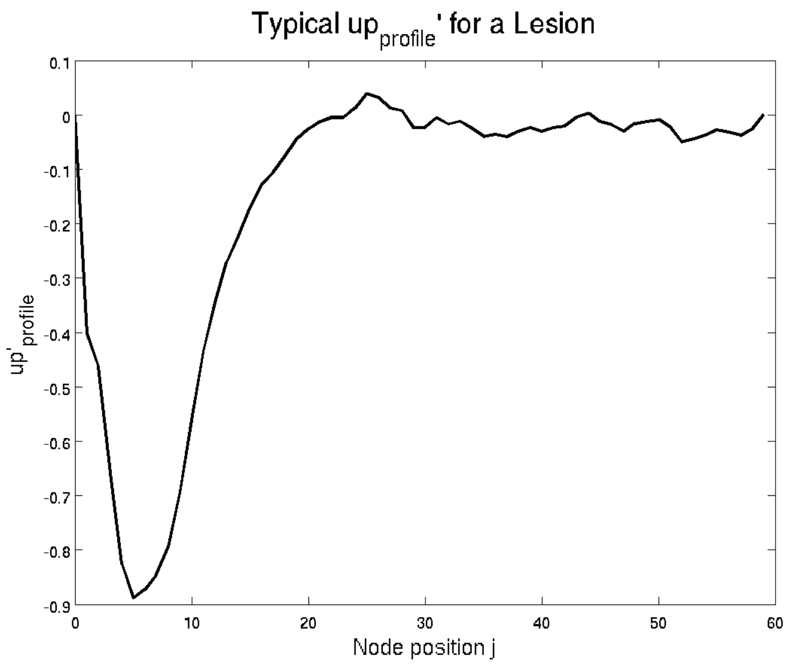


Figure 2.8: The gradient of $up_{profile}$, to determine parts of the slope.

Clearly, this is not defined at $j = 0$ and $j = n_{node} - 1$, but those points will not be needed.

First, the value of steepest descent in $up_{profile}$ is found, with a bias toward the center, shown in Figure 2.9. Along with it, the index j_{low} at this point is found:

$$j_{low} = \arg\min_{j=1,\ldots,n_{node}-2}(up'_{profile}(j) * (n_{node} - \frac{j+1}{n_{node}})). \tag{2.9}$$

The linear bias helps deal with some rare but occurring cases of a significant outside object appearing in the shell profile. This identifies approximately where

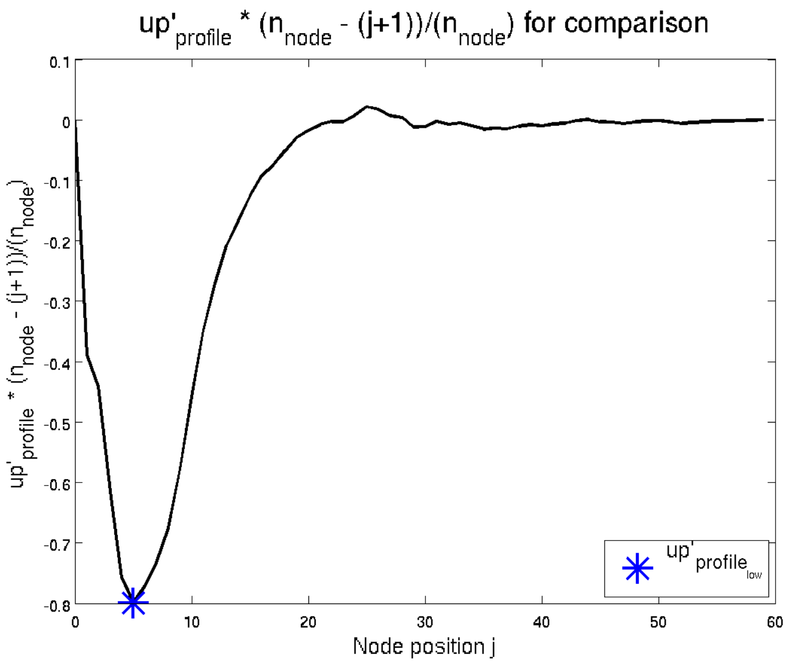


Figure 2.9: The steepest descent is approximately detected, with the center bias.

on the shell where the lesion is changing from high uptake to low rapidly.

Next, the point where the shell approximately levels out should be found. In order to avoid possible issues with a reverberating profile gradient going through the target and back, further calculation is done with a modified version of $up'_{profile}$, such that it never decreases in value after j_{low}. This is up'_{rising}:

$$up'_{rising}(j) = \begin{cases} up'_{profile}(j), & \text{if } j \leq j_{low} \\ \max(up'_{profile}(j), up'_{rising}(j-1)), & \text{otherwise.} \end{cases} \tag{2.10}$$

From this, the latter end of the gradient, where it ends in background, can be found. To avoid possible issues, this no higher than 0, allowing for no other disturbances in the profile. The actual j value isn't needed here; just the gradient value, $up'_{rising_{hi}}$:

$$up'_{rising_{hi}} = \min(0, \max_{j=j_{low}, j_{low}+1, \ldots, n_{node}-1}(up'_{rising}(j))). \tag{2.11}$$

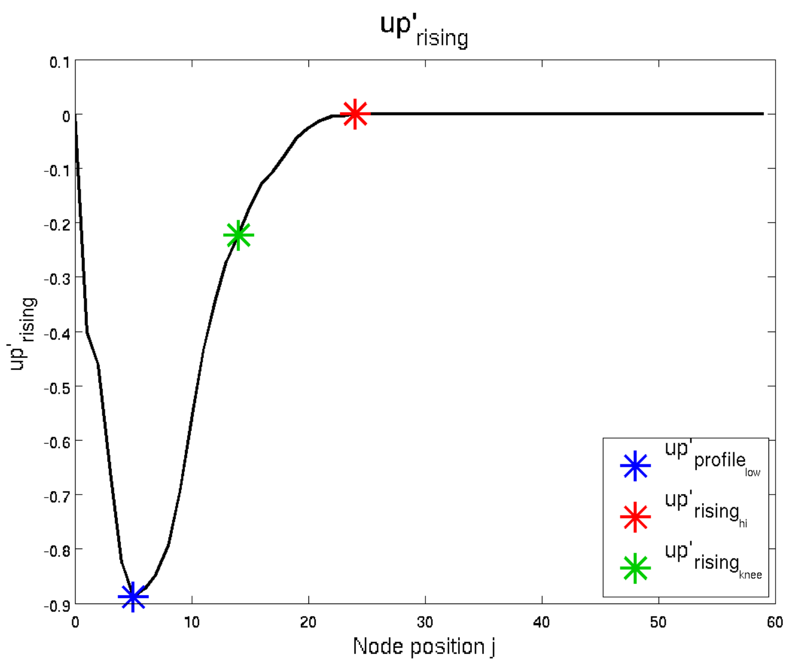


Figure 2.10: The value for $up'_{rising}(j)$ is monotonically increasing for j above j_{low}. Between j_{low} and $up'_{rising_{hi}}$ is $up'_{risingknee}$.

With $up'_{rising_{hi}}$ representing the leveled off portion of $up_{profile}$ and $up'_{profile_{low}}$ representing the steep part, the transition can be found, called $up'_{rising_{knee}}$. These three values are shown in Figure 2.10, and $up'_{rising_{knee}}$ is calculated as follows:

$$up'_{rising_{knee}} = 0.75 * up'_{rising_{hi}} + 0.25 * up'_{profile_{low}}. \tag{2.12}$$

The index in the shell with an index above j_{low} where up'_{rising} is closest to $up'_{rising_{knee}}$ is j_{knee}. As a tiebreaker to deal with multiple equal up_{rising} values equally close to $up_{rising_{knee}}$, use the index closest to where an interpolated up_{rising} function would equal $up_{rising_{knee}}$, rather than just an arbitrary index along the flat line.

Now kn can finally be found:

$$kn = up_{profile}(j_{knee}). \tag{2.13}$$

With that, the peak and knee are calculated and the threshold can be determined. The automatically calculated threshold Th is always between kn and pe. Exactly where is determined by the ratio $\frac{kn}{pe}$:

$$Th_{\%} = 0.8 * e^{\frac{-0.15}{\sqrt{\frac{kn}{pe} * \frac{kn}{pe}}}}. \tag{2.14}$$

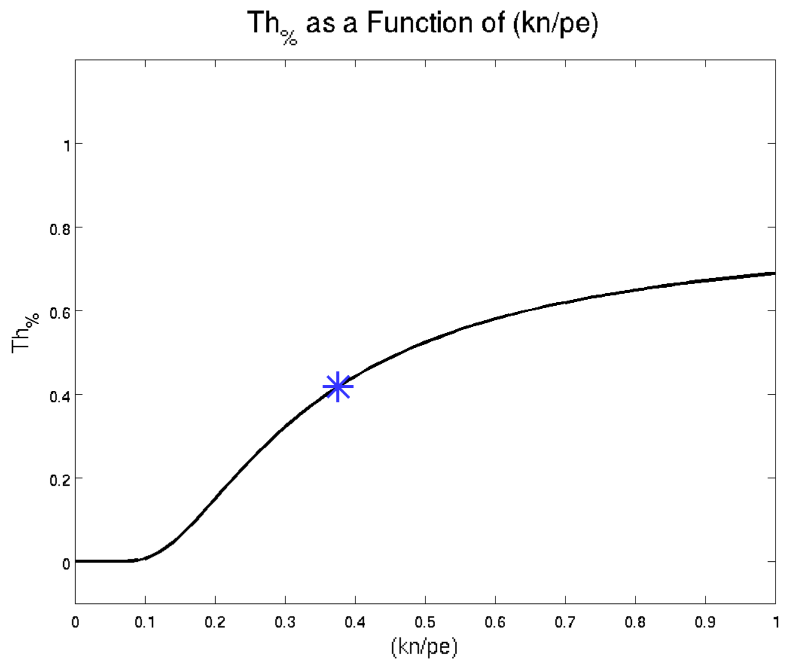


Figure 2.11: The location of Th between kn and pe.

The rather complex equation curve in Equation 2.14 and Figure 2.11 is based off a few concepts:

1. The higher pe is, the closer to kn the threshold should be, to avoid being pulled off by a high-uptake object.

2. The lower pe is, the closer to pe the threshold should be, to preserve the difference between the object and the background.

3. Since Th can never be lower than kn, the function must converge to 0 at low $\frac{kn}{pe}$.

4. Since Th should never be at pe, the function must converge noticeably below 1 at high $\frac{kn}{pe}$.

5. Dividing kn by pe cancels out any issues that could occur from failing to normalize the the image beforehand.

6. It is preferred to underestimate the threshold slightly than to overestimate the threshold.

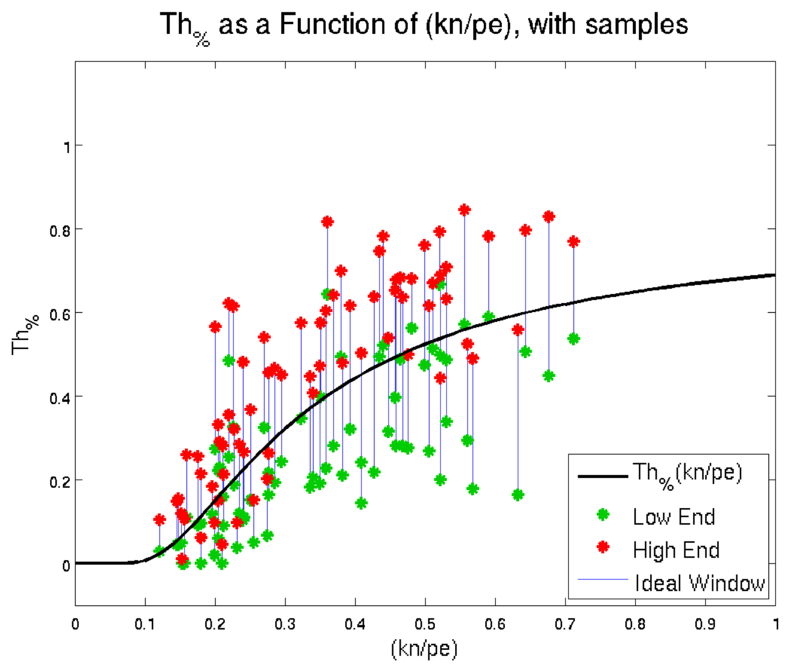


Figure 2.12: The function for $Th_{\%}$ shown against windows for 74 objects. When in error, the preference was generally that the threshold be too low.

The curve was determined heuristically using calculated peak and knee values, in combination with acceptable minimum and maximum threshold values. As visible in Figure 2.12, it is not possible to meet all circumstances. It tends to most often have issues with extraordinarily high uptake objects and those with a "midground" uptake, an area of uptake nearby that is above the background uptake but below the object's uptake.

Furthermore, some alternatives to Equation 2.14 to calculate $Th_{\%}$ can be used for other circumstances. The original equation is based more on segmentations from oncology and is focused on getting all possible voxels of a lesion. However, this can lead to overestimation of volume in imaging modalities with significant smoothing. Some variants based on some standard threshold methods can be used as an alternative to focus more on volume estimation than treatment. These variants are $Th_{\%} = 0.40$ and $Th_{\%} = 0.50$, based on typical 40% and 50% of maximum thresholds, fairly common standards already.

With $Th_{\%}$, finding Th is just a matter of placing it between kn and pe:

$$Th = kn + Th_{\%} * (pe - kn). \tag{2.15}$$

Now Th can be applied to the equations found in Subsection 2.2.3.

2.2.4 Solving

With the cost function c determined for each node and all the main graph edges added to E, the source s and sink t still must be added to finally solve the graph, similar to the manner described in Li's 2006 work [11]. At each node $n_{i,j}$, the cost $c(i,j)$ must be converted to an edge capacity for every node, $c_e(i,j)$:

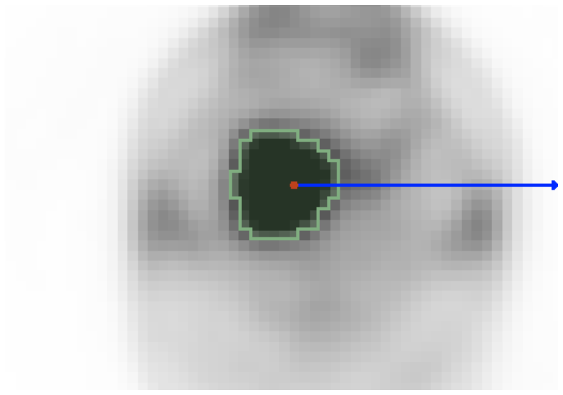
$$c_e(i,j) = \begin{cases} c(i,j) - c(i,j-1), & \text{if } j > 0 \\ -1, & \text{if } j = 0. \end{cases} \tag{2.16}$$

With this calculated, the source and sink nodes can be incorporated into V.

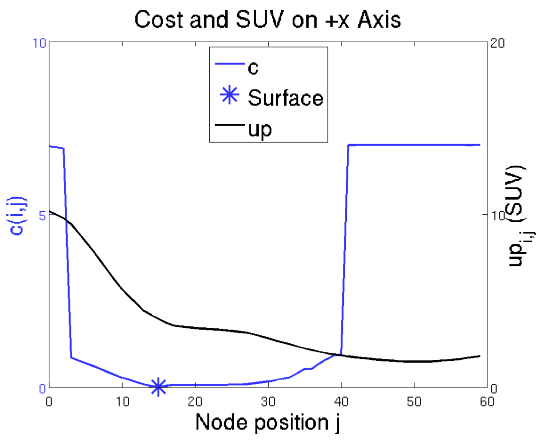
For each node $n_{i,j}$, if $c_e(i,j) > 0$, add the edge $ed_t = \{n_{i,j}, t\}$ with capacity $c_e(i,j)$ to E. If $c_e(i,j) \leq 0$, add the edge $ed_s = \{s, n_{i,j}\}$ with capacity $-c_e(i,j)$ to E. With these added, a minimum cut algorithm can be used on G_k. All nodes to which there is any path with remaining capacity from s is part of the lesion and the outermost such nodes are the surface. All voxels whose centers are located within the boundary are part of the base segmentation for object k.

Voxelization can leave disconnected voxels separate from the main segmentation. The main segmentation can be considered the voxels connected in a 6-neighborhood to the voxel closest to ce_k. After this culling, the remaining voxels are the final active segmentation for object k, which are referred to as S_k.

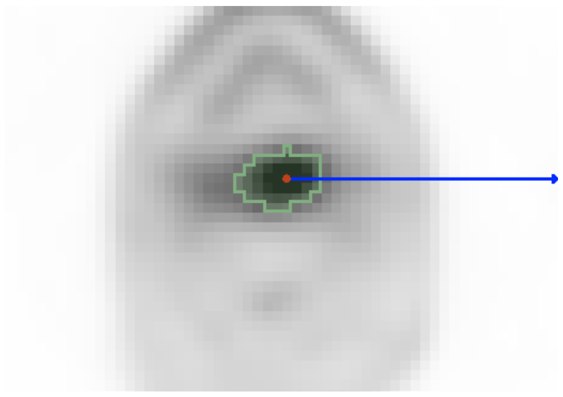
Several examples of segmentations and their cost functions are shown in Figure 2.13.



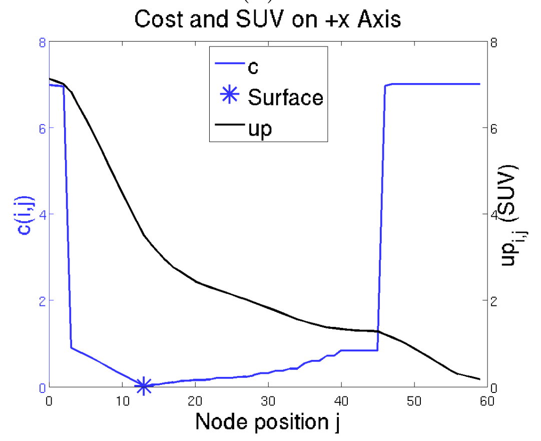
(a)



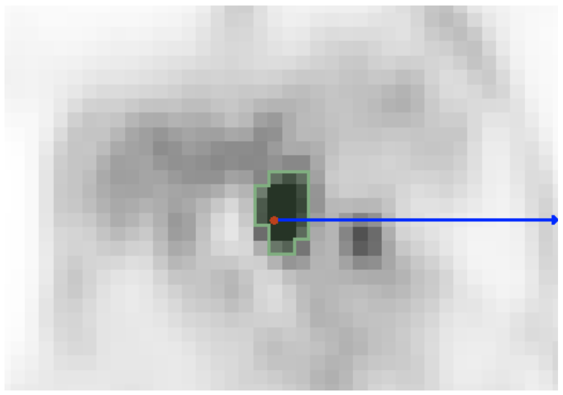
(b)



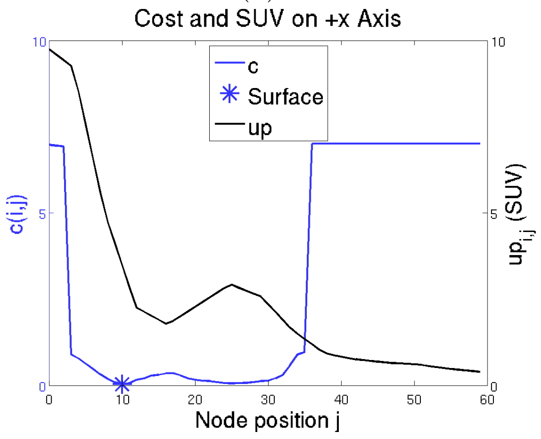
(c)



(d)



(e)



(f)

Figure 2.13: A series of lesions segmented using the methods defined here. All segmentations on the left side such as (a) are paired with their +x axis costs on the right such as (b).

2.3 Refinement Options

If the base algorithm fails or is insufficient, the user can refine the result in any of three ways: threshold refinement, for global surface changes, edge refinement, for narrow surface changes, and sealing, for closing gaps between the new and old objects. Some examples of these in action are in Figure 2.14.

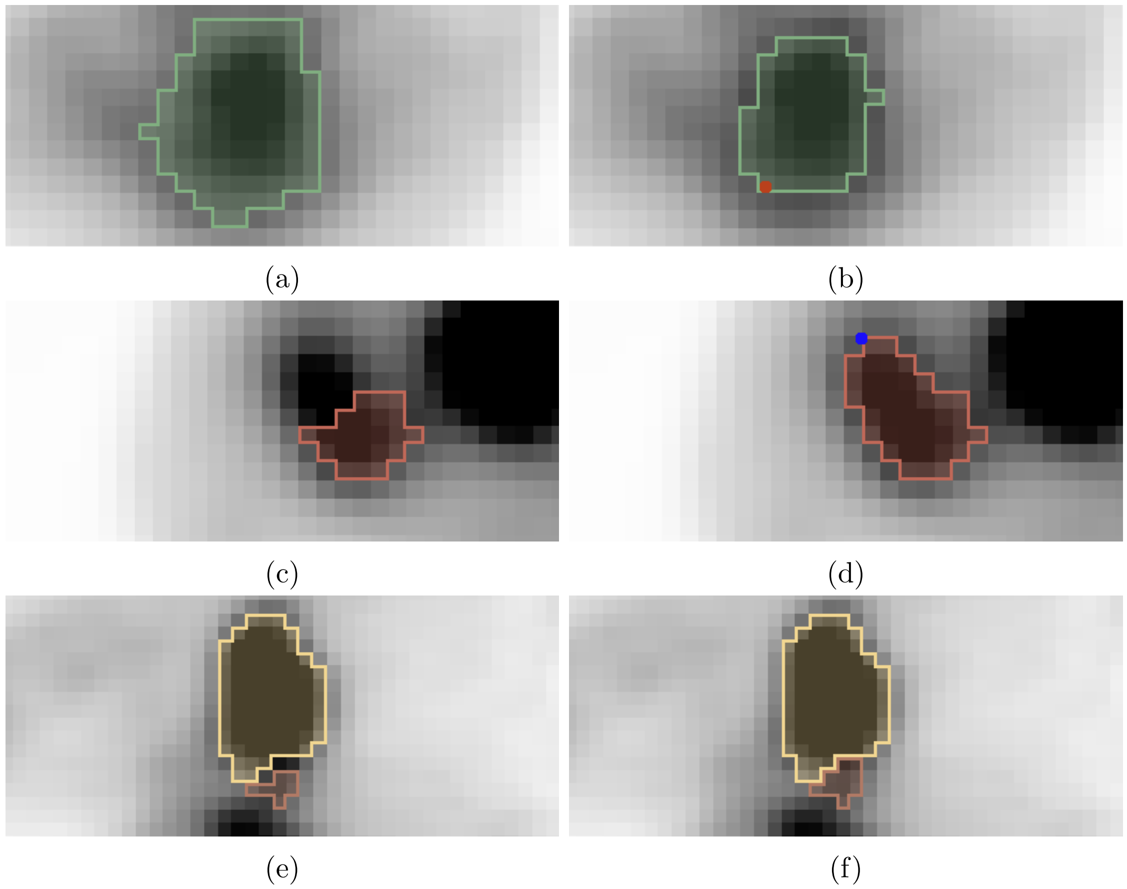


Figure 2.14: Various errors in segmentations, combined with refinement options to solve them. (a) The overall lesion surface expanded outward too much. (b) Threshold refinement changes the boundary to narrow it. (c) Part of the lesion was excluded. (d) The part is reintegrated with a single edge refinement point. (e) A small gap occurs between two adjacent lesions. (f) Sealing fills the gap.

The intent of these refinement options is to give the user fine control over the surface and segmentation, but keep the actual result algorithmic even so. The design of each one is a balance between acting on user input and acting on volume information.

2.3.1 Threshold (Global) Refinement

Threshold refinement changes the value of Th in order to modify the boundaries of the segmentation on a wide scale, likely due to the threshold calculation giving an unsatisfactory result. The default threshold calculation from Section 2.2.3 is decent, but, as shown in Figures 2.12 and 2.15, it won't work in every case.

Threshold refinement is simple and predictable. In cases with a widespread boundary error, the user can place a point for threshold refinement, R_{Th}. In I, the uptake at the voxel closest to R_{Th} is taken as the new value of Th, changing the cost function and thus the entire surface. Furthermore, the closest column to R_{Th} is modified to force the result through the closest node to R_{Th}, which will be called $n_{i_{TR},j_{TR}}$. This is intended to use the information from the user effectively, and possibly save an edge refinement step later. This is accomplished by adding another cost change function, cc_{TR}:

$$cc_{TR}(i,j) = \begin{cases} 1000.0, & \text{if } i = i_{TR} \text{ and } j \neq j_{TR} \\ 0, & \text{otherwise.} \end{cases} \tag{2.17}$$

As with cc_{reject} (Equation 2.4) and with all cc equations to come, this is added to the cost function c_{base} in the calculation of the final cost c. As a result of this cost change, the entire graph must be solved again. The result will now go approximately through the user point, voxel and node spacing aside, along with similar uptake voxels around the shape, as seen in Figure 2.15.

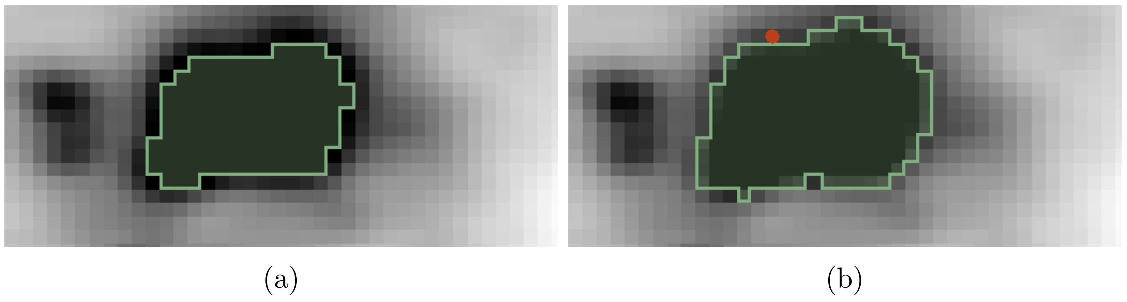


Figure 2.15: A case where the automatic Th is too high. (a) The initial, narrow segmentation. (b) The refined segmentation, with R_{Th} visible in red.

2.3.2 Edge (Local) Refinement

Edge refinement moves the local surface of the segmentation to a new location defined by a specific user point, making a localized change in the overall shape. The intent of this is to fix errors of a narrow scope via a change in the cost. This can solve errors by adding or removing fairly specific parts of objects, as shown in Figure 2.16.

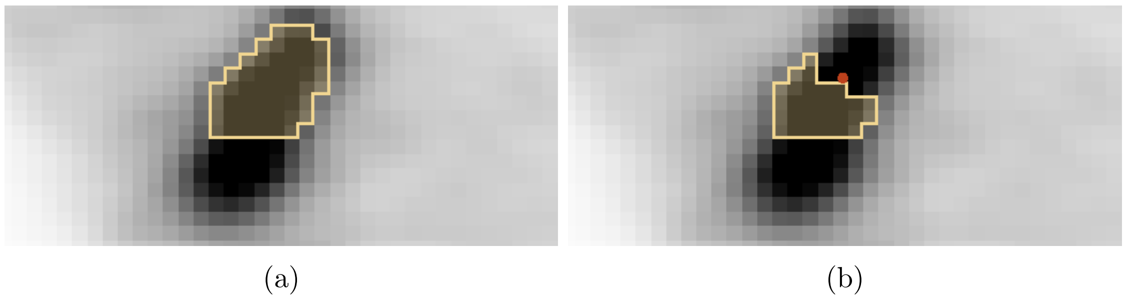


Figure 2.16: A lymph node segmentation where an unintended region is included. (a) The initial segmentation with an extraneous part. (b) The refined segmentation, with the extraneous part removed and a refinement point visible in red.

This is accomplished in a three step process: identify the extent of the new surface segment, apply a cost change to make it attractive, and solve the graph anew with the cost change applied.

The extent of the surface segment is determined based on the columns around

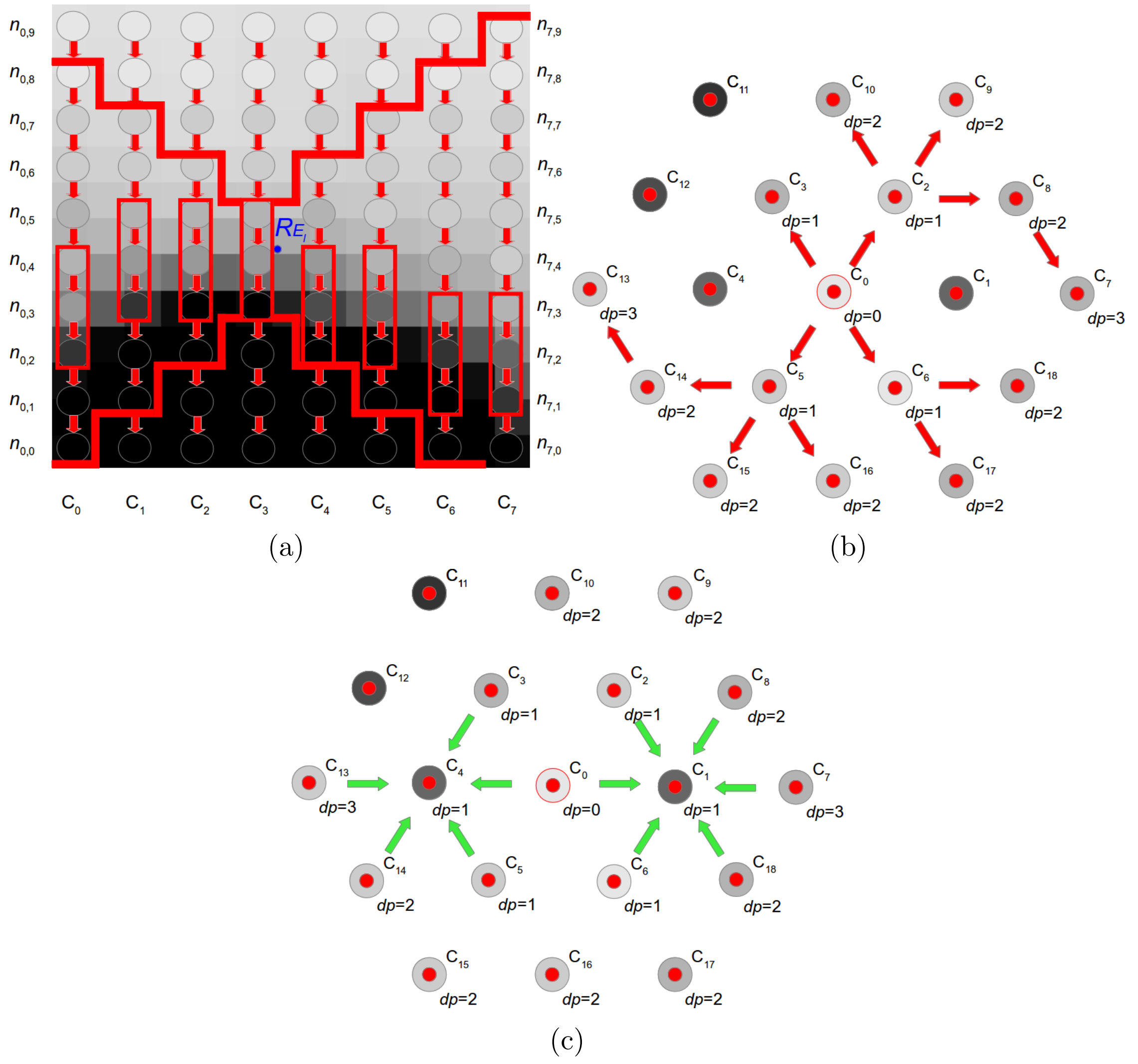


Figure 2.17: An overview of edge refinement propagation. (a) The vector around R_{E_l} on the column marked C_3 is compared to vectors on adjacent columns within range to find the best match for the refinement. The vector in this image is smaller than in practice. (b) Propagation between columns, viewed from the surface of the spherical mesh. The center column propagates out to adjacent similar columns, which propagate out further to columns similar to the original. Propagation flows around dissimilar columns. (c) After the initial propagation, dissimilar columns may be modified as well. If two thirds of their neighbor columns were considered similar, the dissimilar column is modified with the original depth it would've had.

the refinement point specified by the user, R_{E_l}. The extent-marking propagation mechanism, roughly shown in Figures 2.17a and 2.17b, is based on the uptake around R_{E_l}. The node closest to it is known as $n_{i_{ER},j_{ER}}$. A vector of nodes is formed around, made of seven nodes from $n_{i_{ER},j_{ER}-3}$ through $n_{i_{ER},j_{ER}+3}$. This vector of nodes is compared to nearby vectors of nodes and adjacent columns, and if found similar to those, then to columns adjacent to those columns as well, and further out to a limit of five columns outward. This process is intended to find sections of the uptake in the graph nodes that are part of the same surface as the original user's point, so that a specific smooth surface can be incorporated into the final surface result.

For a specific column being refined, the number of propagations made to reach it is known as the depth dp. At $dp = 5$, propagation ceases. To determine if column i' is similar and can be included in refinement, there is a similarity function sim for nodes $n_{i_{ER},j_{ER}}$ and $n_{i',j'}$ and a similarity condition smc also using the depth dp of the second node to go with it. In short, a column and node will be similar if the absolute value of the vector differences between the original vector around $n_{i_{ER},j_{ER}}$ and the vector around the prospective $n_{i',j'}$ are below a certain threshold. The similarity function sim is as follows:

$$sim(n_{i_{ER},j_{ER}}, n_{i',j'}) = \sum_{\Delta j=-3}^{3} (|up_{i_{ER},j_{ER}+\Delta j} - up_{i',j'+\Delta j}|). \tag{2.18}$$

This equation compares nodes against each other as a vector to track threshold and overall features. To take into account regional uptake intensity and proximity, the specific requirement for similarity is first that $sim(n_{i_{ER},j_{ER}}, n_{i',j'})$ be below a threshold th_{sim}:

$$th_{sim} = 0.05 * \sum_{\Delta j=-3}^{3} (|up_{i_{ER},j_{ER}+\Delta j}|). \tag{2.19}$$

Also required is that the node j' be within an intra-column distance from j equal to the depth dp. The similarity condition function smc is then as follows:

$$smc(n_{i_{ER},j_{ER}}, n_{i',j'}, dp) = sim(n_{i_{ER},j_{ER}}, n_{i',j'}) < th_{sim} \text{ and } |j_{ER} - j'| \leq dp. \quad (2.20)$$

Among all nodes $n_{i',j'}$ on a column for which *smc* is true, the node that minimizes the value of the *sim* function is chosen as the node to center the refinement cost changes on the column around.

While similarity is useful, it can leave small numbers of unmarked columns amidst several marked ones, which can result in jagged spots on the surface after applying the cost change. This can be fixed by checking for these narrow sets of unmarked columns and marking them in an additional step. For a dissimilar column to have its cost changed, it must have been skipped in propagation only due to dissimilarity and two thirds of its immediate neighbors must be similar to the original column. In this case, its most similar node will be used, within the original depth dp for which it was considered dissimilar. This is shown in Figure 2.17c.

Once no more columns can be marked as part of the surface, the specific nodes on each column are collected in a list $ERList_{l_n}$, and the associated depth values on another list $ERList_{l_d}$. As such, $[ERList_{l_n}, ERList_{l_d}]$ is a list of nodes paired with their depth values. The presence of $[n_{i,j'}, dp_{i,j'}]$ in the list means that column i will be refined around node $n_{i,j'}$ with an associated depth of $dp_{i,j'}$.

On the original column where $dp = 0$, the closest node is simply taken as the target point, and the cost is changed on that column so all other nodes besides the chosen one are increased in cost by 1000.0, the same as for the single specific column in threshold refinement in Equation 2.17. On other nodes, the cost is decreased around the target node by $3.0 * e^{\frac{-\Delta j^2}{2*dp^2}}$, with Δj being the difference in node from the target node. This decrease is narrow on low depth nodes, but much wider on high depth nodes, allowing for a smoother transition from refined to unrefined in the surface, as well as encouraging selection near the new marked surface, if not directly on it due to smoothness constraints. This function is shown at low and high depths in Figure 2.18.

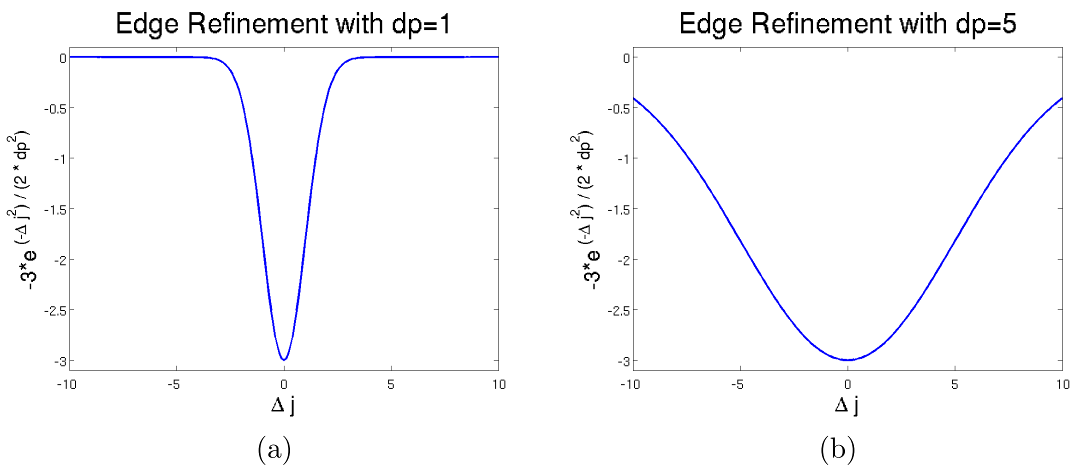


Figure 2.18: The cost change around a refinement point. At lower depths, the cost change is more stark, while it widens out at greater depths, though it is still low enough to move the boundary solution.

Now the cost change function for the refinement point R_{E_l}, cc_{er_l}, can be written in its entirety:

$$cc_{er_l}(i,j) = \begin{cases} 1000.0, & \text{if } (\exists j' \mid [n_{i,j'}, dp_{i,j'}] \in [ERList_{l_n}, ERList_{l_d}] \\ & \text{ and } dp_{i,j'} = 0 \text{ and } j' \neq j) \\ -3.0 * e^{\frac{-(j-j')^2}{2*dp^2}}, & \text{if } (\exists j' \mid [n_{i,j'}, dp_{i,j'}] \in [ERList_{l_n}, ERList_{l_d}] \\ & \text{ and } dp_{i,j'} \neq 0) \\ 0, & \text{otherwise.} \end{cases} \quad (2.21)$$

With these, multiple edge refinement points can be placed on the graph for a lesion, each with their own distinct cost change. Their cumulative effect will just be additive, with the caveat that columns from which propagation originates will not have their costs further changed. That is, if there is a cost change with $dp = 0$ on a column, nodes on the column will, in total, be changed only by +1000.0 or not at all, regardless of other refinement points.

After determining the extent of the modified surface segment and applying the cost change, the only thing left to do is solve the graph. This can be done by simply resolving the graph from scratch with the cumulative effects of all refinement points. However, to save time, the graph can be solved again using the initial solution from before the refinement was applied, or the solution from the previous refinement, if more than one is applied.

2.3.3 Sealing

Sealing is a simple refinement process to fill gaps between segmentations, occasionally left during segmentations of adjacent lesions. When segmenting such adjacent lesions, it is difficult to perfectly place the boundary with respect to voxels, since it is based on physical distance only. As such, there can often be small gaps between objects that should be touching each other, as in Figure 2.19a.

Figure 2.19: A case where a segmentation of two adjacent objects leaves a gap between them. l_{new} is the red label. (a) The single-voxel gaps are visible. (b) The gaps are filled after the sealing is complete.

Sealing solves this issue on a voxel level. When a non-object voxel in L is between an object voxel (label $l \neq l_{bg}$) and a voxel of the active segmentation S_k (label l_{new}), if that voxel corresponds to an uptake in I that is above the threshold Th, then it is changed by sealing. For a voxel at (x, y, z) in L to be "between" two voxels of labels l' and l'' means that, along at least one axis direction, the voxel is adjacent to a voxel of label l' and label l''. As an equation, $btw(L, x, y, z, l', l'')$ is the following:

$$btw(L,x,y,z,l',l'') = ((L(x\pm 1,y,z)=l' \text{ and } L(x\mp 1,y,z)=l'') \text{ or } (L(x,y\pm 1,z)=l' \text{ and } L(x,y\mp 1,z)=l'') \text{ or } (L(x,y,z\pm 1)=l' \text{ and } L(x,y,z\mp 1)=l'')). \tag{2.22}$$

The equation for *btw* is confusing, so Figure 2.20 has been provided to give some examples of what precisely it entails.

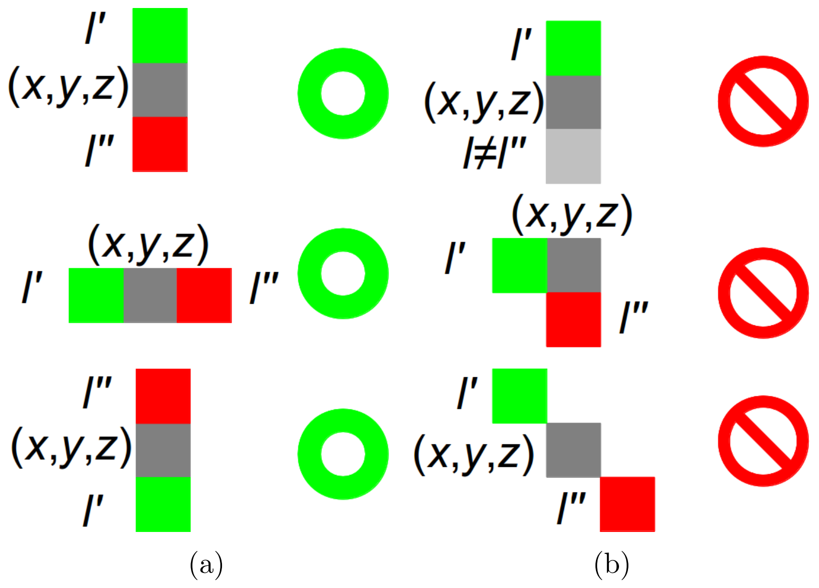


Figure 2.20: An overview of 2D voxel configurations and how *btw* reacts to them. (a) Configurations where (x,y,z) (gray) is between voxels for l' (green) and l'' (red). (b) Configurations where (x,y,z) is not between voxels for l' and l''.

This condition can be written down as an equation, the voxel seal condition vsc for a voxel in (x,y,z) space:

$$vsc(x,y,z) = L(x,y,z) = 0 \text{ and } btw(L,x,y,z,l_{new},l \neq l_{bg}) \text{ and } I(x,y,z) > Th. \tag{2.23}$$

With this condition, the determination of when to seal can be made fairly easily, and then make the final modified L, which is referred to L_{sealed}:

$$L_{sealed}(x,y,z) = \begin{cases} l_{new}, & \text{if } S(x,y,z) = l_{new} \text{ or } vsc(x,y,z) = \text{True} \\ L(x,y,z), & \text{otherwise.} \end{cases} \quad (2.24)$$

This closes single-voxel gaps between the active segmentation S_k and adjacent segmented objects, if they meet the threshold criteria, as shown in Figure 2.19b. Note that this only expands S_k; sealing does not attempt to expand adjacent lesions. Since it is accomplished entirely on a voxel level, there is no need to solve the graph again to make this refinement.

There are additional methods for dealing with this issue, such as label avoidance from Section 2.4.1, which can also reduce the cost of nodes adjacent to the next lesion, but this still relies on voxels matching with the graph. Additionally, paired segmentation of lesions as discussed in Section 4.2 can allow for no gap between the graphs of the lesions and therefore no voxel gap. However, sealing is an effective way of handling it for single-lesion graphs when label avoidance is insufficient.

2.4 Modes

Beyond the algorithm as described in the previous sections, there are alternative modes for the algorithm to work in, based on the circumstances of the lesion. These are for dealing with separate lesions that are near other lesions or with highly necrotic lesions. These are intended to work when the basic algorithm fails, or is expected to fail, dealing with cases not recognizable just from the information in the volume I. These different modes can affect the graph setup, cost setting, solving, and even refinement steps in various ways, and can be used in combination with each other. Some image cases where these modes might be used are shown in Figure 2.21.

1. Label avoidance prevents new lesion labels from overwriting old, as well as preventing the center from being placed too close to other objects.

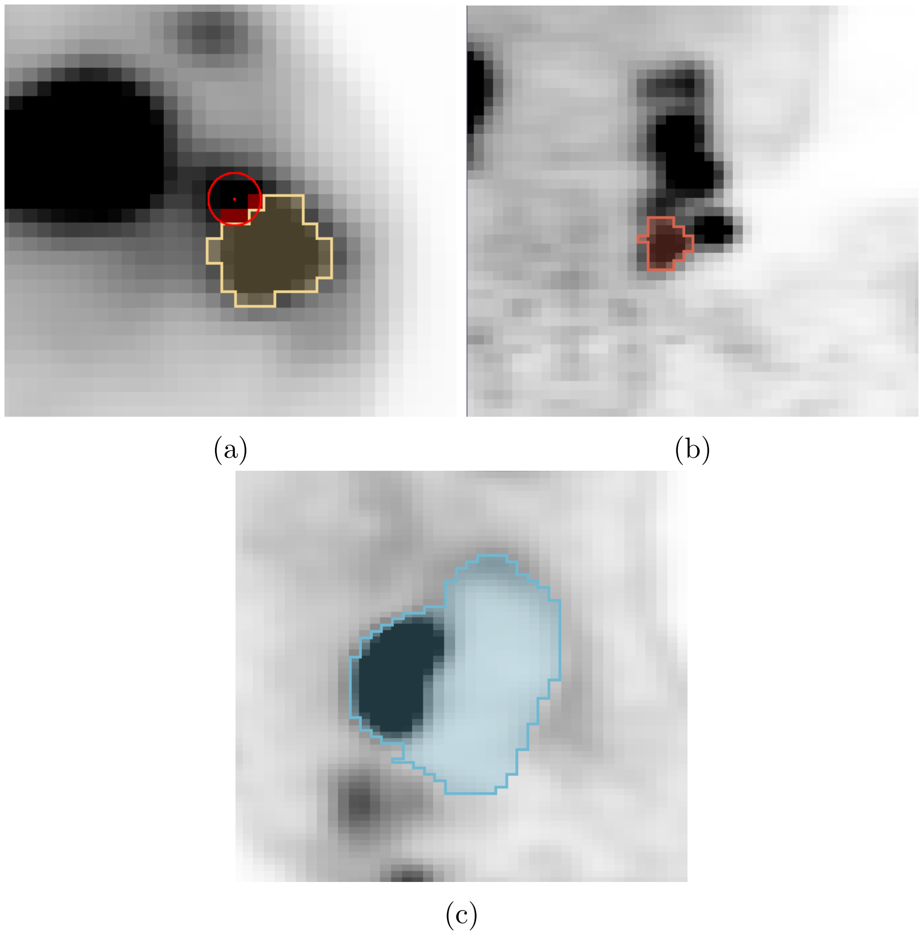


Figure 2.21: Circumstances for the different modes. (a) Label Avoidance: Placing a new object next to an existing one. (b) Splitting: Segmenting one lymph node in a high-uptake chain. (c) Necrotic Mode: Segmenting a large region below the overall region median.

2. Splitting cuts off parts from the target object to get the minimum object like an edge detection algorithm.

3. Necrotic mode makes it easier to segment large necrotic regions where uptake is low.

2.4.1 Label Avoidance

The label avoidance mode, demonstrated in Figures 2.22 and 2.23, is intended for use near other segmented lesions in L that should not be overwritten. It prevents the new segmentation from overwriting other lesions, though the cost function changes ignore nodes that are on labels equivalent to that of the pending active segmentation S_k. It has effects on the graph setup phase, cost setting phase, and solving phase.

As mentioned in Section 2.2.1, there is an option for recentering the center of the graph from $ce_{k_{user}}$ to ce_k. Label avoidance prevents ce_k from being placed on or adjacent (in a 6-neighborhood) to a lesion label in L different from that nearest to $ce_{k_{user}}$, as demonstrated in Figure 2.22. This prevents recentering onto other objects that already exist, and helps reduce "nesting", wherein the boundaries in certain directions are unable to go anywhere due to the presence of another label immediately in that direction. In the case that there is no viable location for ce_k to be recentered to, it is simply set to $ce_{k_{user}}$.

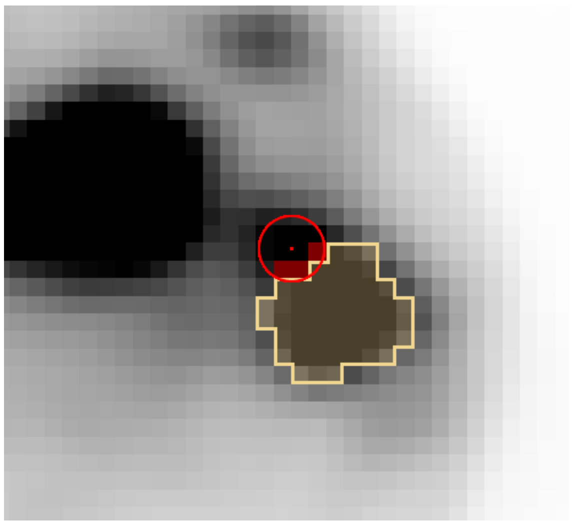


Figure 2.22: The effect of label avoidance on recentering. The recentering radius around $ce_{k_{user}}$ is $r = 7.0$ mm, around two voxels. The point ce_k must be placed near it, but many voxels are blocked (shaded red) due to the presence of the existing yellow-labeled object.

For changes in the cost setting, label avoidance modifies cc_{reject} (Equation 2.4), converting it into $cc_{reject_{la}}$, which is very similar. This new cost change equation for rejection requires knowledge of the closest label to a given node, which is known as $l_{i,j}$ for node j on column i. Suppose that the object label currently being applied (to indicate an object in L) is l_{new} and that the label for no object in L is l_{bg}. If the object on a node in a column is some other label l' not equal to those, then that node and others beyond it are rejected with label avoidance, to avoid segmenting into an existing object. This is used to add a new label rejection condition, rc_{label}, at a given node and column:

$$rc_{label}(i,j) = j > j_{min} \text{ and } (\exists j' \mid 0 \leq j' \leq j \text{ and } l_{i,j'} \neq l_{bg} \text{ and } l_{i,j'} \neq l_{new}). \quad (2.25)$$

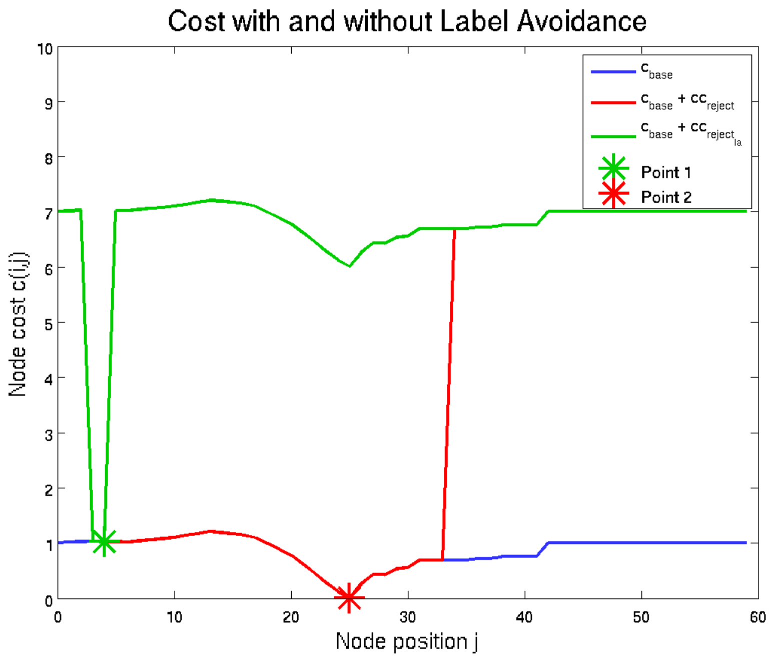
With this, cc_{reject} is modified into $cc_{reject_{la}}$:

$$cc_{reject_{la}}(i,j) = \begin{cases} rej, & \text{if } j < j_{min} \text{ or } rc_{low}(i,j) = \text{True or } rc_{label}(i,j) = \text{True} \\ 0, & \text{otherwise.} \end{cases} \quad (2.26)$$

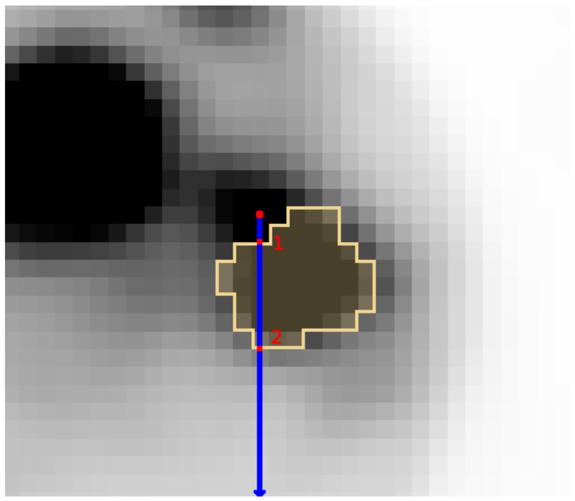
This prevents the graph from placing the boundary into an existing object in L, the existing labels image. An example of this cost change is shown in Figure 2.23.

One more effect of label avoidance is dealing with cases of adjacent lesions with no clear boundary. When one lesion is sought next to another, but there is nothing in c_{base} to give a useful surface location between them, then something must be done.

Figure 2.24 shows one of these special cases. Sometimes, a column has no attractive cost feature such as a minimum from the center until it reaches another lesion label that blocks off further nodes. The solution would default to selecting the closest non-rejected node to the center. However, that leaves a gap between



(a)



(b)

Figure 2.23: The effect of label avoidance on the cost function c. (a) Costs due to label avoidance along an axis, from ce_k. Both cc_{reject} and $cc_{reject_{la}}$ reject the first few nodes, but $cc_{reject_{la}}$ also begins to reject as soon as it encounters the other label. The two points show the different low points in the cost function. (b) The axis on which the costs are changed, with two points marked. It is for the same lesion as in Figure 2.22.

two objects that should be in contact. So, an additional cost change function is added, cc_{seal}, in order to close the gap. The specific condition is that there is never

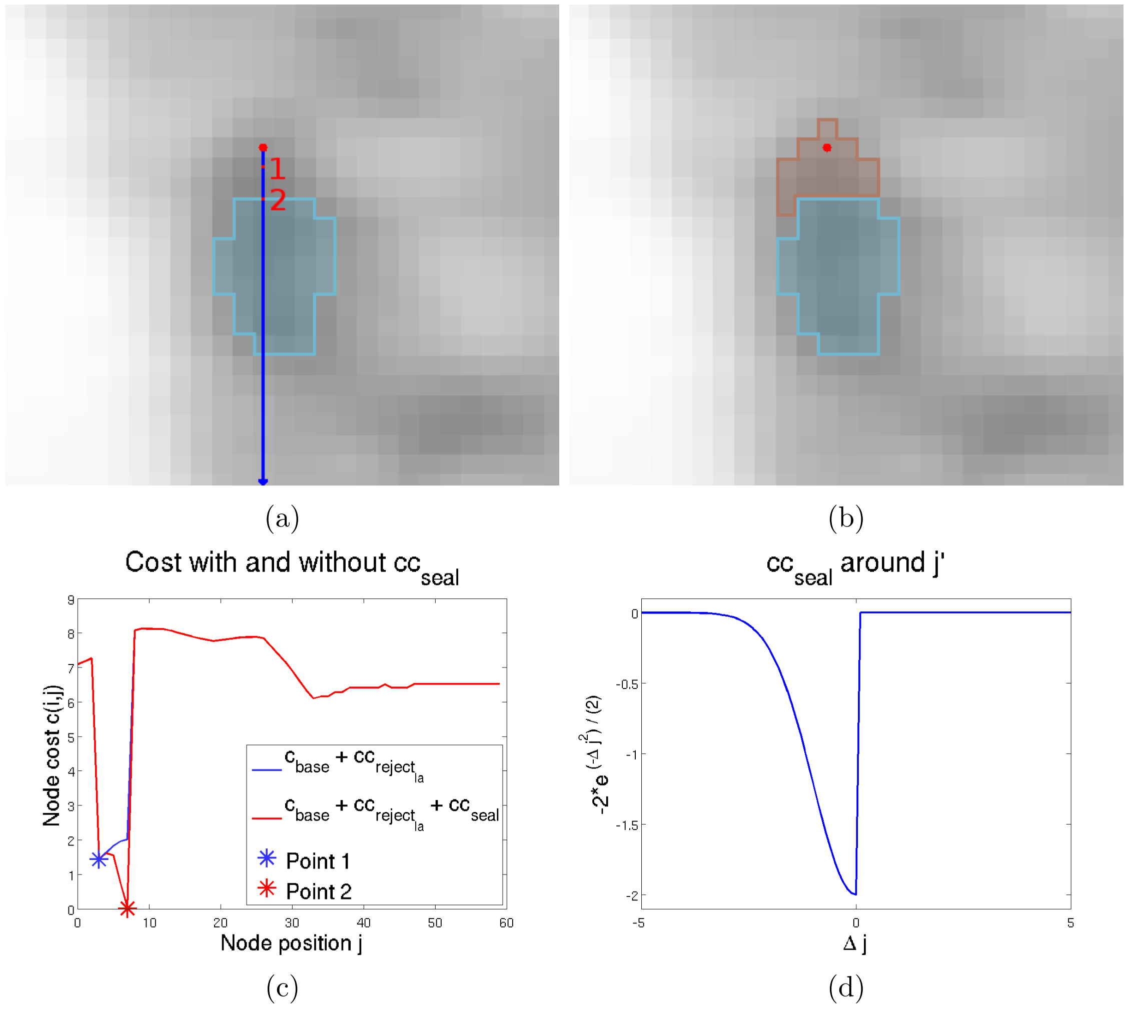


Figure 2.24: An example of a lesion with no real feature between its center and an adjacent lesion. (a) The graph center ce_k and the other lesion in light blue. (b) The segmentation made due to to the sealing effect of label avoidance. (c) The cost change due to cc_{seal} for the marked axis. (d) The general shape of the cost change.

a decrease in cost from the center point until the node that is part of another lesion label. This condition is called the cost seal condition csc for that column on a certain node:

$$\begin{aligned} csc(i,j) = j > j_{min} \text{ and } l_{i,j+1} \neq (l_{bg} \text{ or } l_{new}) \text{ and} \\ (\nexists j', j'' \mid j_{min} - 1 \leq j' < j'' \leq j \text{ and } up(i,j') > up(i,j'')) \text{ and} \\ (\nexists j' \mid 0 \leq j' \leq j \text{ and } l_{i,j'} \neq (l_{bg} \text{ or } l_{new})). \end{aligned} \tag{2.27}$$

This condition checks several distinct things. $j > j_{min}$ prevents the very first possible node from being rejected, which would lead to all nodes being rejected, which is meaningless. $l_{i,j+1} \neq (l_{bg} \text{ or } l_{new})$ determines that the next node is closest to another lesion's label, rather than a background label or a leftover label of the object being applied. $(\nexists j', j'' \mid j_{min} - 1 \leq j' < j'' \leq j \text{ and } up(i,j') > up(i,j''))$ determines that there are no features between the center and j, such as a decrease in cost. $(\nexists j' \mid 0 \leq j' \leq j \text{ and } l_{i,j'} \neq (l_{bg} \text{ or } l_{new}))$ makes it exclusive to the first occurrence of another lesion label, so there can only be at most one node meeting the condition on any column.

With the cases where this condition is met, there is no real cost feature, so one must be added. This is what cc_{seal} adds, as shown in Figures 2.24c and 2.24d, decreasing smoothly below the target node with a depth of $d_{seal} = 2.0$ and a width factor of $\sigma_{seal} = 1.0$:

$$cc_{seal}(i,j) = \begin{cases} -d_{seal} * e^{\frac{-(j-j')^2}{2*\sigma_{seal}^2}}, & \text{if } (\exists j' \mid csc(i,j') = \text{True and } j \leq j') \\ 0, & \text{otherwise.} \end{cases} \tag{2.28}$$

Lastly, during the solving process, if somehow (possibly due to edge refinement) the surface includes voxels labeled for other lesions, they will simply not be overwritten, nor counted as connecting components when removing disconnected voxels in a 6-neighborhood, as was described in Section 2.2.4.

2.4.2 Splitting

The splitting mode is for individually segmenting lesions, especially lymph nodes, that could otherwise leak into unlabeled adjacent lesions, as seen in Figure 2.25. This is accomplished by applying stricter smoothness, reducing node costs around watershed boundaries and uptake minima, and adding a bias toward the centermost cost features. While the standard mode would typically combine lesions that are in close proximity to each other, the splitting mode singles out a lesion. This makes the algorithm work similar to watershed algorithms mentioned in Section 1.2 with regard to boundaries between lesions, but it still acts closer to a threshold-based segmentation for the boundary with the background.

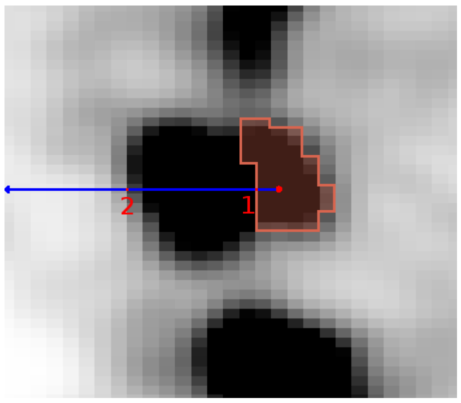
The soft smoothness constraint from Section 2.2.1 is increased from $sp = 0.005$ to $sp = 0.05$, applying a much higher penalty for surface roughness. Parts of a surface that are further out from the rest of the shape are more likely to be cut off by the smoothness, which helps for the relatively spherical lymph nodes. Furthermore, the surface becomes more responsive to the edge refinement presented in Section 2.3.2.

In the cost setting phase, uptake minima along the axis along with watershed boundaries are emphasized in the cost function by adding a cost change function, cc_{split}. There are three conditions for applying a cost change on the function, and each causes a different level of cost change function to be applied. The first splitting condition is for a local minimum uptake, spc_{min}, at a certain node and column:

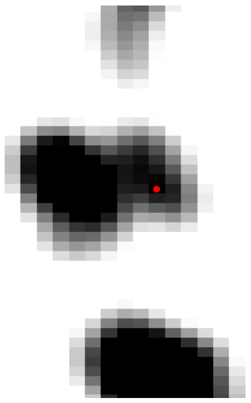
$$spc_{min}(i,j) = up(i,j-1) > up(i,j) \leq up(i,j+1). \tag{2.29}$$

The other two are based on watersheds of different levels. Watershed segmentations must be made of an inversion of I in the spherical region around ce_k. The first ”strong” watershed is calculated with a fill level of 20% of the maximum difference in the spherical region, where the level is the threshold for watershed unions in Lefvre's 2007 work [9]. The second ”weak” watershed is calculated with a fill level of 0% of the maximum, expected to oversegment. An example of these watershed regions is shown in Figure 2.26.

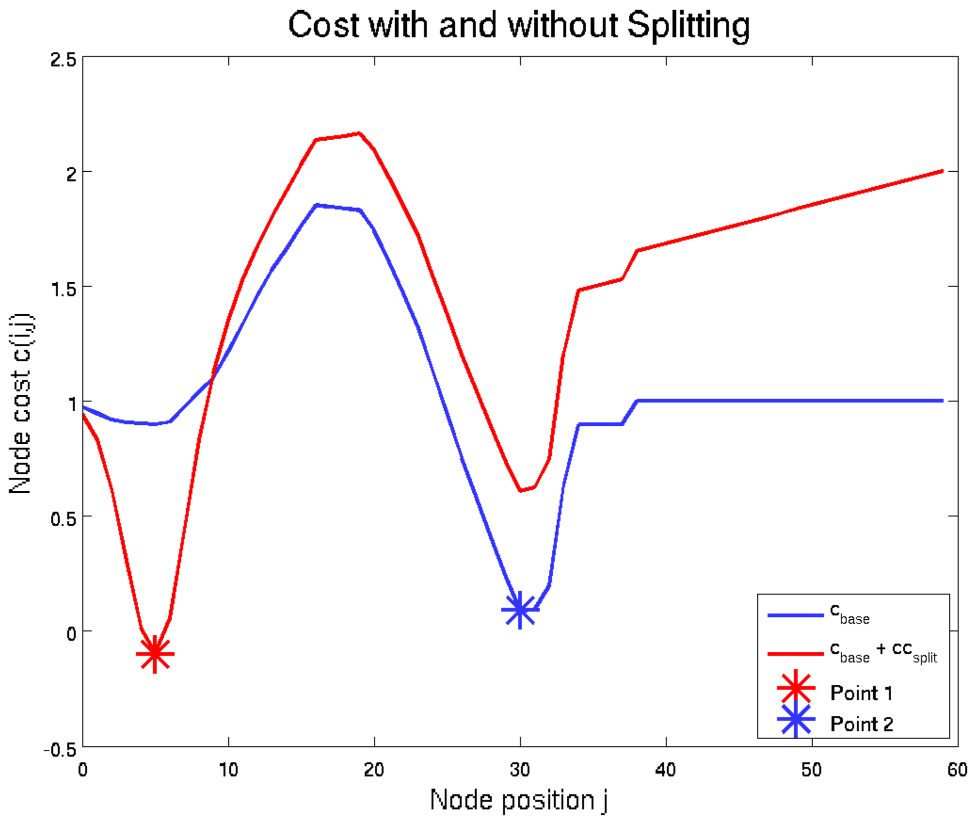
The label of a strong watershed closest to node $n_{i,j}$ is $sws_{i,j}$, while the label for a weak watershed is $wws_{i,j}$. When two adjacent nodes on a column are in



(a)



(b)



(c)

Figure 2.25: A pair of lymph nodes, bridged by high uptake. (a) The segmentation made with splitting mode, and a single column off of ce_k. (b) An adjusted view that better shows the separation between the objects. (c) The base cost along the marked column, and the base cost with the effect of splitting added.

different watersheds and are in the above-threshold part of the column, one of the two splitting conditions for watersheds is triggered, spc_{wws} for the weak watershed

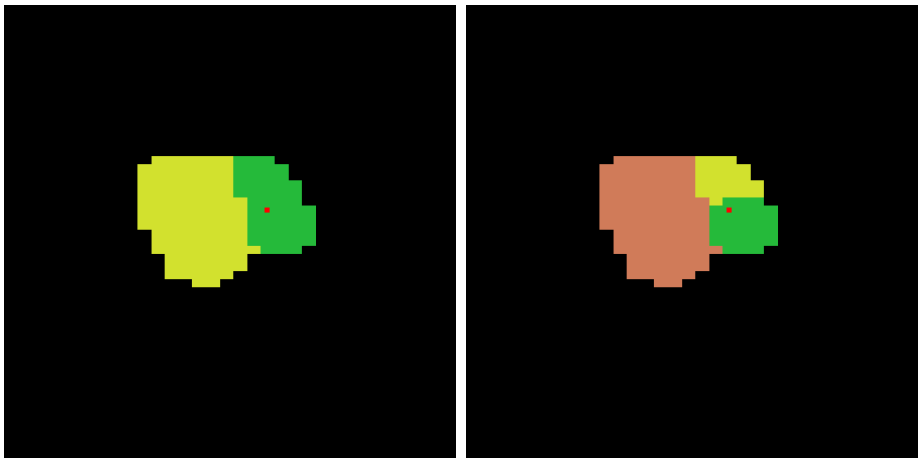

(a) (b)

Figure 2.26: Watersheds for a pair of adjacent lesions. This is the same pair from Figure 2.25. ce_k is marked in red. Those on voxels with an uptake below Th are blacked out, since they do not affect the segmentation. (a) The strong watersheds, cutting cleanly between the two objects. (b) The weak watersheds, making the same cut, but adding a secondary cut. Using both allows detection of minor cuts while emphasizing the major ones.

and spc_{sws} for the strong:

$$spc_{sws}(i,j) = (sws_{i,j-1} \neq sws_{i,j} \text{ and } (\nexists j' \mid up(i,j') < Th \text{ and } 0 \leq j' \leq j)). \quad (2.30)$$

$$spc_{wws}(i,j) = (wws_{i,j-1} \neq wws_{i,j} \text{ and } (\nexists j' \mid up(i,j') < Th \text{ and } 0 \leq j' \leq j)). \quad (2.31)$$

These spcs can all be true on multiple nodes on a given column. A single one is significant enough to affect the outcome, but multiple features at a location, which can happen often, is much more significant and will be taken over a single condition. The uptake minima are naturally low points in c_{base}, but not always significant enough to affect the surface on their own, and they do not necessarily form a complete boundary between lesions. The watersheds may have no effect on c_{base} at all, but form complete, reliable boundaries. Combining them makes for effective detection and prioritization of edges between lesions.

Each node at which a splitting condition is met (or which has a "feature") on is a splitting node. The sets of splitting nodes, snl_{min}, snl_{sws}, and snl_{wws}, hold all of these features' locations. Subsets for each column are useful for the cost equations, so they are divided into snl_{min_i}, snl_{sws_i}, and snl_{wws_i}. The contents of the set are combined in a superset snl_i consisting of all nodes with features on column i.

At the location of a node with a condition met, a cost change is applied with a depth d based on the type of splitting node. This cost function, $notch$, is a function of the difference in j and the depth d, with a common width factor $\sigma = 2.0$. It is shown in Figure 2.27. This allows for the splitting feature to affect the surface solution even if smoothness constraints prevent the exact splitting node from being part of the surface:

$$notch(\Delta j, d) = -d * e^{\frac{-(\Delta j)^2}{2*\sigma^2}}. \tag{2.32}$$

The value of d varies with the feature type. For the uptake minimum, $d_{min} = 0.4$. For the weak watershed, $d_{wws} = 0.5$. For the strong watershed, $d_{sws} = 0.2$. It seems off at first that $d_{sws} < d_{wws}$, but weak watershed features occur on any node a strong watershed feature occurs, and their effects are additive for a depth of 0.7 at the location, possibly even with a minimum at the same or nearly the same node. Each feature type has an independent effect on the cost:

$$cc_{split_{min}}(i,j) = \sum_{n_{i,j'} \in snl_{min_i}} notch(j - j', d_{min}), \tag{2.33}$$

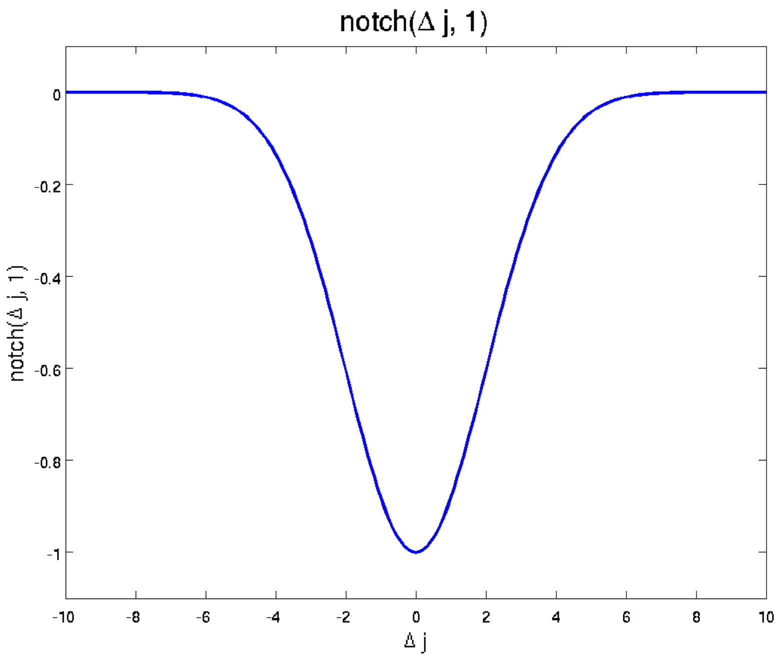


Figure 2.27: Profile of the notch equation around each feature. It makes a smooth decline into the target node and back out, giving some leeway in the final surface selection to deal with smoothness constraints.

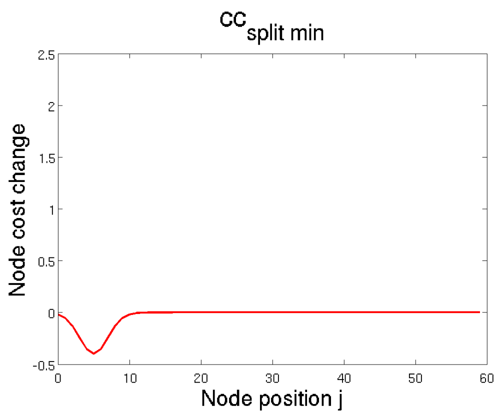
$$cc_{split_{wws}}(i,j) = \sum_{n_{i,j'} \in snl_{wws_i}} notch(j - j', d_{wws}), \tag{2.34}$$

$$cc_{split_{sws}}(i,j) = \sum_{n_{i,j'} \in snl_{sws_i}} notch(j - j', d_{sws}). \tag{2.35}$$

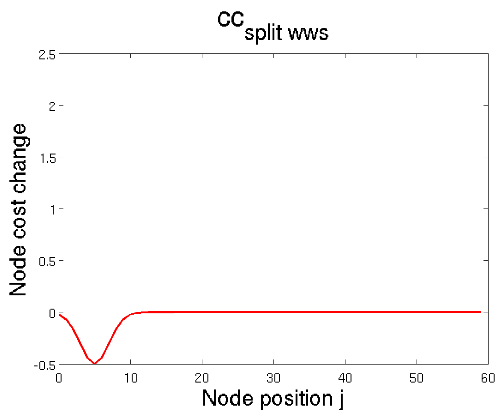
These are added together, along with a linear bias toward the center-most features that only applies to columns with features. This bias is a linear value from 0 to a maximum of 1.0 at the outermost node. This bias plus the individual feature cost changes makes up cc_{split}:

$$cc_{split}(i,j) = \begin{cases} cc_{split_{min}}(i,j) + cc_{split_{wws}}(i,j) + \\ cc_{split_{sws}}(i,j) + \frac{j+1}{n_{node}}, & \text{if } snl_i \neq \emptyset \\ 0, & \text{otherwise.} \end{cases} \tag{2.36}$$

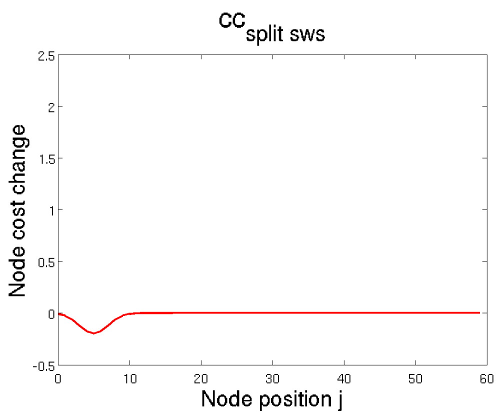
This equation is shown in its various parts in Figure 2.28. As with all cc



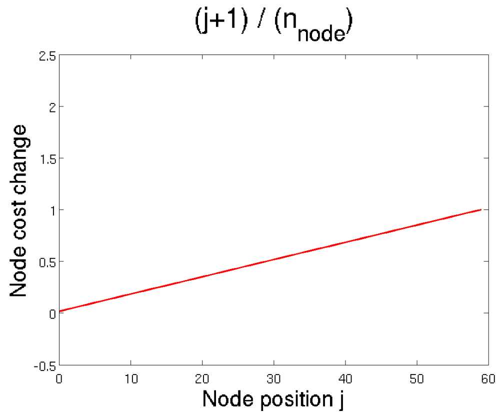
(a)



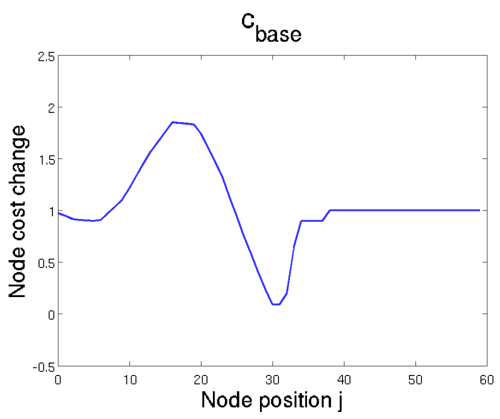
(b)



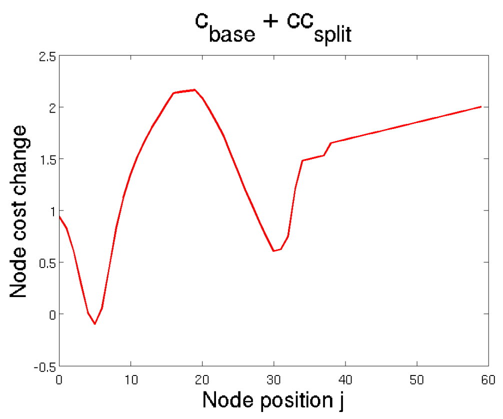
(c)



(d)



(e)

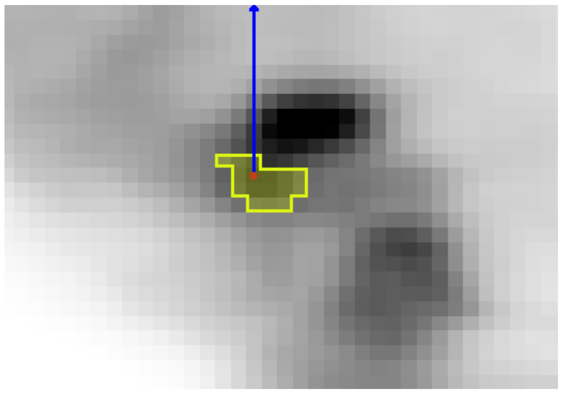


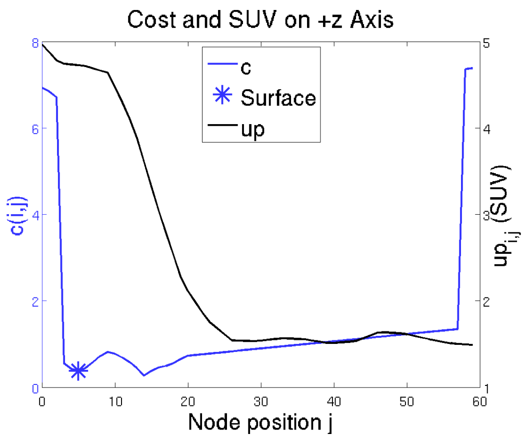
(f)

Figure 2.28: Cost changes due to splitting, by component. They are added together with the base cost to make the modified cost from Figure 2.25. (a) The cost changes due to uptake minima along the column. (b) The cost changes due to crossing a strong watershed. (c) The cost changes due to crossing a weak watershed. (d) The cost changes due to center bias. (e) The base cost. (f) The combined final costs.

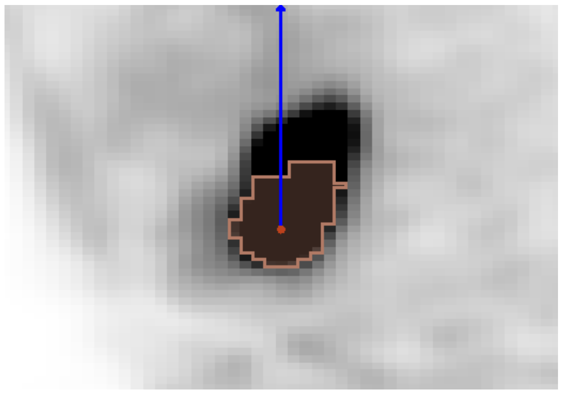
equations, this is added to the c_{base} function to make the final c function that the graph will be solved based on. The effect of this and the change to the soft smoothness constraint, when splitting is active, cause the overall result to be far more effective at separating out lymph nodes from clusters.

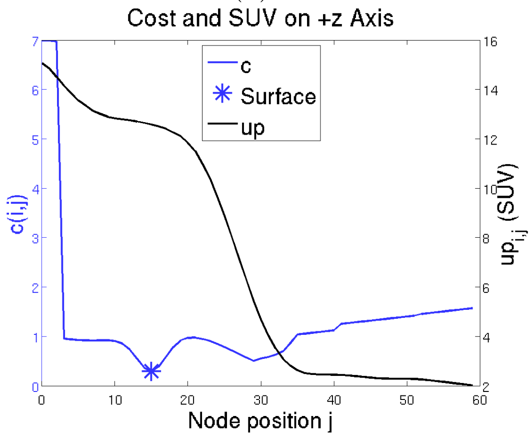
Several examples of this mode's effect in general are shown in Figure 2.29.


(a)

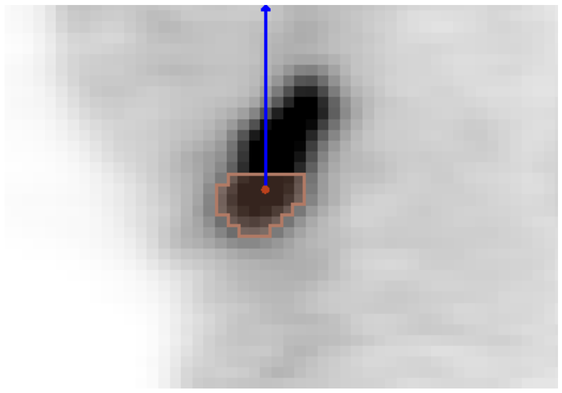


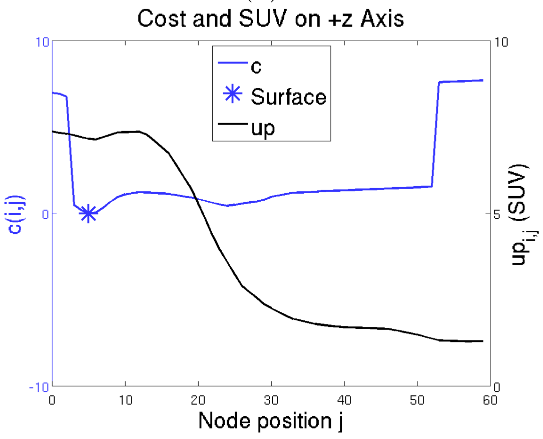
(b)


(c)



(d)


(e)



(f)

Figure 2.29: A series of lesions segmented with splitting. All segmentations on the left side such as (a) are paired with their +z axis costs on the right side such as (b).

2.4.3 Necrotic Mode

Necrotic mode is intended for rather large low-uptake sections of a tumor or lymph node (as in Figure 2.30). The standard method is intended for objects with an uptake above that of the background. While many necrotic objects can be segmented sufficiently just using the edge refinement in Section 2.3.2 due to the algorithm's minimal use of the contents of the surface, there are cases extreme enough where that becomes very tedious or impossible.

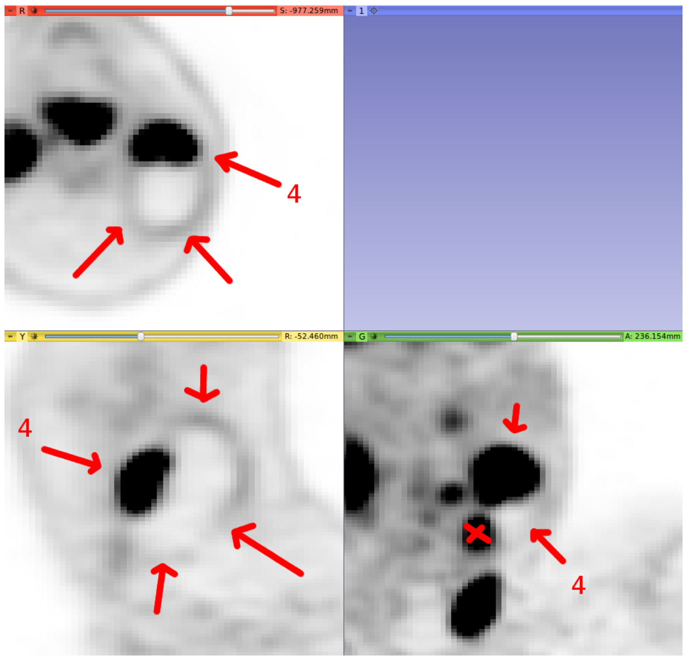


Figure 2.30: A massive necrotic part of a lymph node has to be included along with the high-uptake part.

The intended use of this mode is to segment the low-uptake part of a tumor that is partially or entirely necrotic while the high-uptake part is segmented with more standard modes. It could be used to segment both high and low uptake parts together when combined with edge refinement, however. If there is a non-necrotic portion, that should be segmented first. This mode prevents low uptake from causing the function to fail completely due to rejection, and, if label avoidance is also active, causes it to seal much more readily to other uptakes nearby.

For the most part, this mode doesn't specifically promote segmentation of necrotic regions, but rather reduces the restrictions that prevent them. It affects the cost setting and one of the refinement options, the voxel-wise sealing, described in Section 2.3.3.

In the cost setting phase, this mode, too, modifies cc_{reject} (Equation 2.4) by changing the base rejection condition it relies on, rc_{low} (Equation 2.3). Instead of rejecting as soon as the uptake goes below the regional median M_{region} as in Equation 2.3, the uptake must first go above the threshold, which is first done at node j_{Th_i} on column i:

$$j_{Th_i} = \min_{j=0,1,\ldots,n_{node}-1}(j \mid up(i,j) > Th). \tag{2.37}$$

With this, the exact condition for low rejection with necrotic mode can be given:

$$rc_{low_{nec}}(i,j) = j > j_{min} \text{ and } \min_{j'=j_{Th_i},j_{Th_i}+1,\ldots,j}(up(i,j')) < M_{region}. \tag{2.38}$$

Now the rejection for being too low won't happen immediately on necrotic regions. A center ce_k placed inside this region will be able to functionally segment the region, as long as the threshold isn't too low. The threshold will need to be adjusted, using threshold refinement (Section 2.3.1).

In order to improve sealing to other objects, necrotic mode also interacts with label avoidance. Specifically, the function for sealing to other objects with no feature between them, cc_{seal} (Equation 2.28), as well as the cost seal condition for it, csc (Equation 2.27), are modified as well.

The modification to csc is simple, though with csc's base complexity, the result is still a rather complex equation. To convert csc into csc_{nec}, the variant when label avoidance and necrotic mode are on, basically the condition against a decrease in uptake is slackened. It is not checked for along the column until the uptake along the column has gone above the threshold Th, similar to the equation for $rc_{low_{nec}}$:

$$\begin{aligned}csc_{nec}(i,j) = j > j_{min} \text{ and } l_{i,j+1} \neq l_{bg} \text{ and } (\nexists j', j'' \mid j_{Th_i} < j' < j'' \leq j \text{ and}\\ j_{min} - 1 \leq j' \text{ and } up(i,j') > up(i,j'')) \text{ and } (\nexists j' \mid 0 \leq j' \leq j \text{ and } l_{i,j'} \neq l_{bg}).\end{aligned} \tag{2.39}$$

This is essentially the same as the original $csc(i,j)$, but with one clause modified. Instead of avoiding an increase in uptake from near the center until the label, it simply avoids an increase in uptake from the node $j = j_{Th_i}$ until the label, allowing for uncertain increasing and decreasing in the low-uptake necrotic region. Furthermore, instead of treating labels of l_{new} like the background, they are treated like other labels instead, allowing them to be sealed to.

$$nsc(i,j) = l_{i,j+1} = l_{new} \text{ and } \forall j' \leq j \mid l_{i,j'} = l_{bg}. \tag{2.40}$$

With just necrotic mode, cc_{seal} only recognizes it.

$$cc_{seal_{nec}}(i,j) = \begin{cases} -d_{seal} * e^{\frac{-(j-j')^2}{2*\sigma_{seal}^2}}, & \text{if } (\exists j' \mid nsc(i,j') = \text{True and } j \leq j') \\ 0, & \text{otherwise.} \end{cases} \tag{2.41}$$

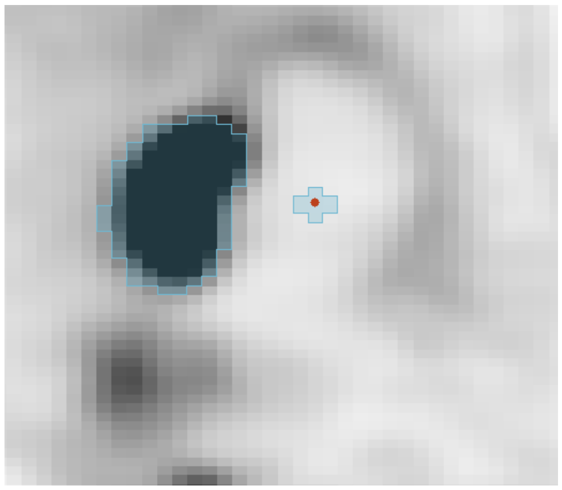
With both label avoidance and necrotic mode active, cc_{seal} uses the earlier of either node of the two sealing conditions for the sealing point:

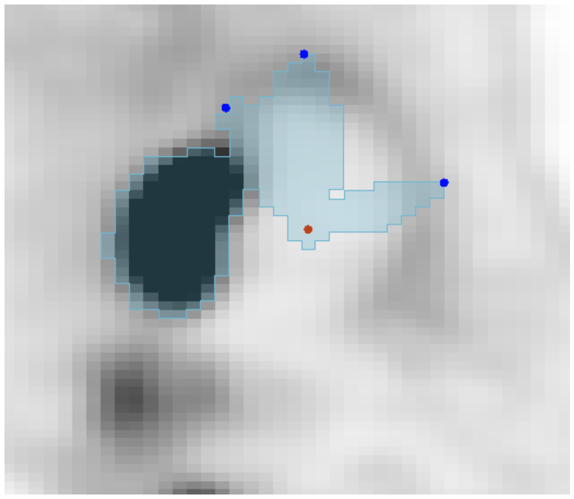
$$cc_{seal_{la,nec}}(i,j) = \begin{cases} -d_{seal} * e^{\frac{-(j-j')^2}{2*\sigma_{seal}^2}}, & \text{if } j' = \min(j_a, j_b | (\exists j_a \mid csc_{nec}(i,j_a) = \text{True}) \\ & \text{ or } (\exists j_b \mid nsc(i,j_b) = \text{True})) \text{ and } (j \leq j') \\ 0, & \text{otherwise.} \end{cases} \tag{2.42}$$

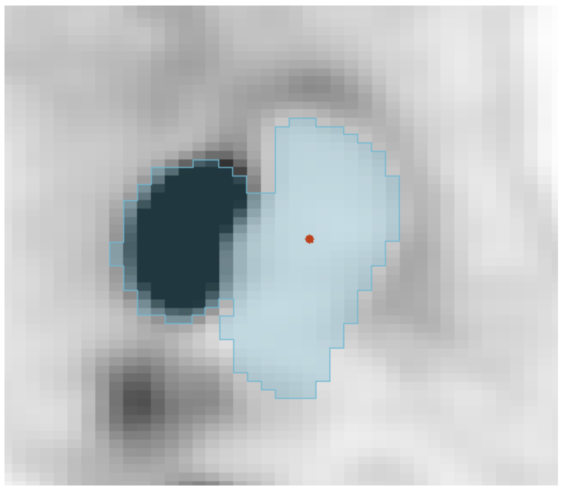
The result of this is that for a partially necrotic object, if the non-necrotic section is segmented beforehand, the necrotic part can fairly easily seal into it to close gaps. This is the intended use of this mode.

Aside from these effects on the cost setup, necrotic mode also affects one of the refinement options, sealing (Section 2.3.3). Here, necrotic mode just removes the requirement that the voxel being sealed at be above the threshold. This allows for more thorough sealing of necrotic voxels.

An example of the overall effect of using necrotic mode as intended is shown in Figure 2.31.


(a)


(b)


(c)

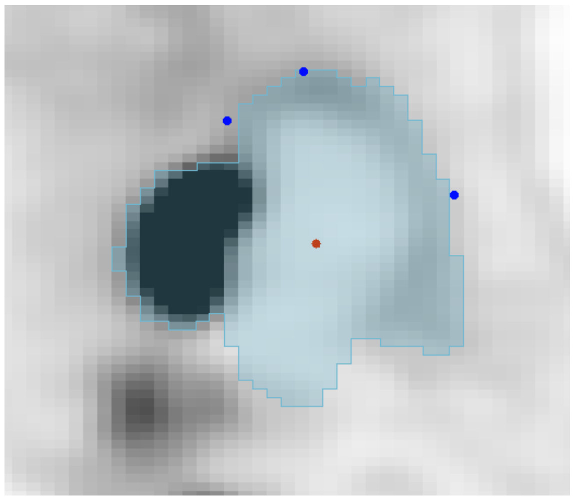

(d)

Figure 2.31: A partially necrotic lesion. The necrotic side is attempted with and without necrotic mode, and with and without edge refinement. Red points are centers; blue points are edge refinemen points. (a) Necrotic mode is off, no refinement is performed. The algorithm stops short due to low uptake rejection. (b) Necrotic mode is off, refinement is attempted. The algorithm fails to spread significantly around the refinement points. (c) Necrotic mode is on, no edge refinement is performed. The strictly necrotic part is reasonably segmented. However, the algorithm doesn't include the moderate-uptake boundary of the object. (d) Necrotic mode is on, refinement is applied. The base segmentation is decent and the refinement spreads effectively to capture the boundary.

2.5 Adaptations

The algorithm was designed primarily for real data and furthermore based on radiation oncology segmentations of real data. This results in a focus on expanding to catch the entire possible extent of a lesion. This focus, however, leads to some overestimation of volume. In order to better estimate volume for things such as phantom images, a few adaptations should be applied.

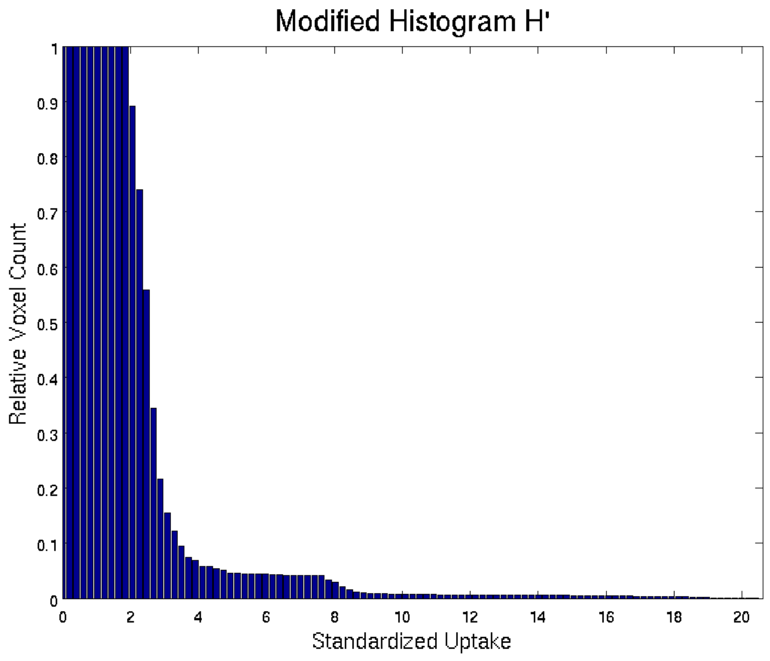
Due to their artificial nature, phantoms may have little or no transition from object to background, compared to images of real data. Due to the structure of the cost function as described in Section 2.2.2, the algorithm relies on the presence of this transition of uptake, which forms the uncertainty for the low-uptake side of $c_{base}(i,j)$ (Equation 2.2). An unnaturally uniform background and border region results in next to no transition voxels between the "lesion" and the background, as shown in Figure 2.32. Since a threshold is generally above the background, this means that for uptakes between the threshold Th and the background, there's little to no cost difference between nodes. This causes the soft smoothness constraint (Section 2.2.1) to seriously affect segmentations.

An adaptation for this known as the "linear background" adaptation handles this. The c_{base} function is modified into c'_{base}, which replaces the histogram-based portion with another linear relationship between uptake and cost:

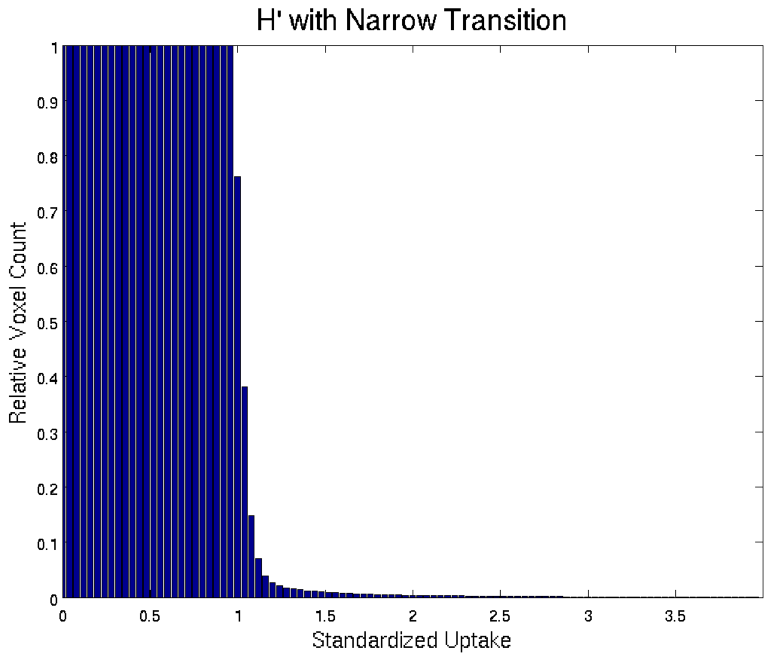
$$c'_{base}(i,j) = \begin{cases} \frac{Th-up(i,j)}{Th}, & \text{if } up(i,j) < Th \\ 0, & \text{if } up(i,j) = Th \\ \frac{up(i,j)-Th}{up_{ce}-Th}, & \text{if } up(i,j) > Th \\ & \text{ and } up_{ce} > Th \\ 1, & \text{otherwise.} \end{cases} \tag{2.43}$$

This prevents the minimally-featured histogram from causing surface decisions that are essentially based only on smoothness for uptake below the threshold.

Other phantoms can have abnormally high noise. This can lead to individual



(a)

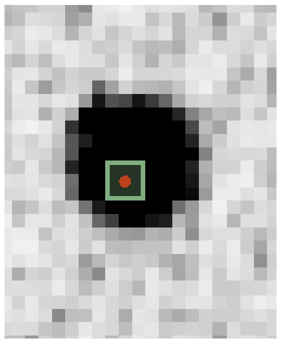


(b)

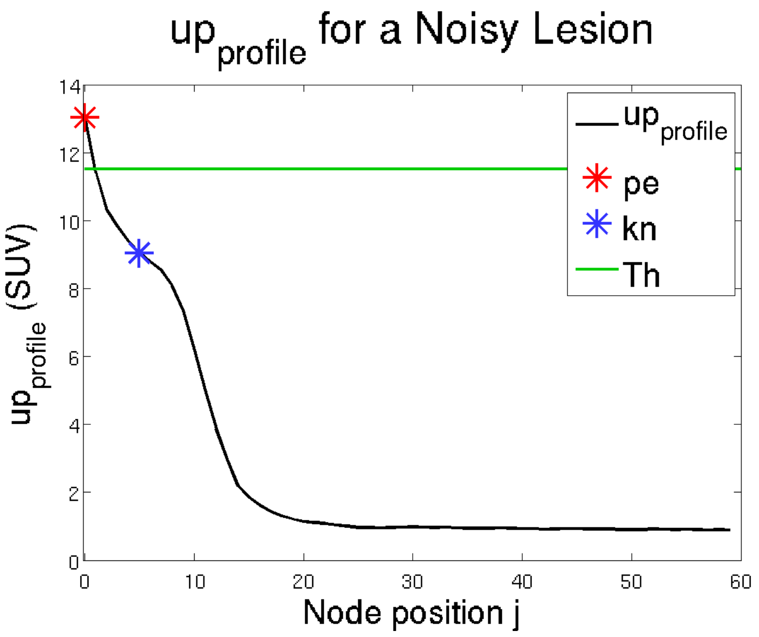
Figure 2.32: Histograms of uptake in the region around a graph center. (a) A typical histogram in real data. (b) A histogram for a phantom. The transition from near 0 to 1 is much narrower in this graph.

voxels that are much higher or lower than the rest of the nearby region. Especially when combined with recentering, this can significantly influence the automatic threshold Th, as shown in Figure 2.33. To deal with this, an additional "threshold de-noise" adaptation was designed.

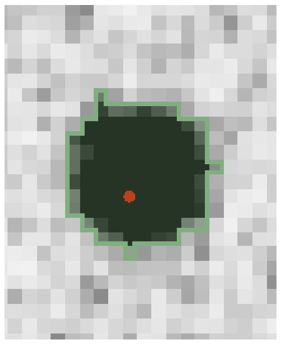
The effect of this is that the threshold calculations from Section 2.2.3 that rely on shells use modified uptake values. Instead of the shells being medians of the node samples of I directly, a copy of I is filtered through a median filter with a 26-neighborhood to form I_m, which is then sampled at the nodes, and the medians on each shell used for the profile and threshold calculation. This prevents noisy scans from hindering the threshold calculation, but doesn't use I_m in the overall process, thereby preserving any edge information that could be useful in other applications of this adaption.



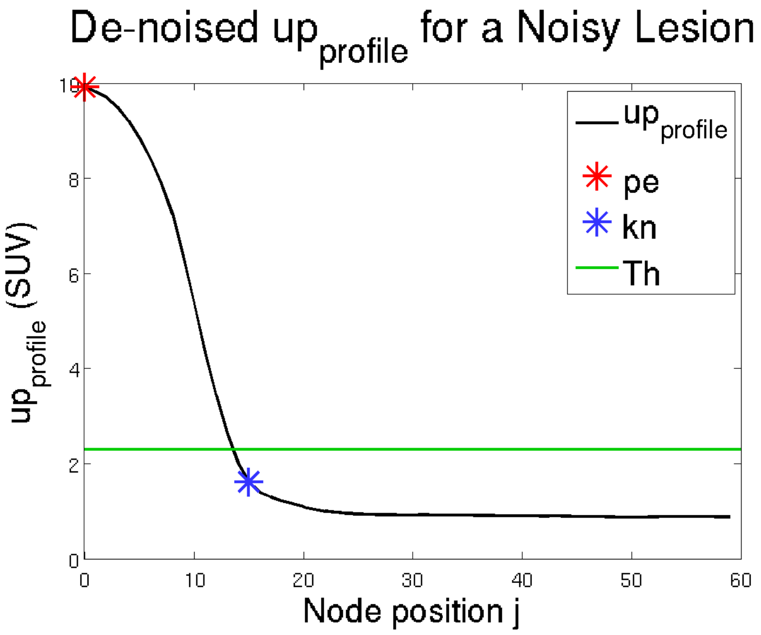
(a)



(b)



(c)



(d)

Figure 2.33: Segmenting a noisy object without and with threshold de-noise. (a) The surface cuts off very quickly due to a high threshold. (b) The high threshold is caused by an effective extra knee in the data. (c) The surface contains the entire object. (d) The de-noise adapation's median calculation removes the extra knee and gives a much more sensible threshold result.

CHAPTER 3
VALIDATION

An experiment has been designed to test this algorithm against manual segmentation, the current typical gold standard for segmentation. In this section, results are given that can be used to determine the speed, consistency, and accuracy of the algorithm. These results come from application of the algorithm to a set of real image data and a set of phantom image data.

3.1 Real Data

There were 60 total FDG PET scans in the set of real data. Of these, 59 were pre-treatment scans, and one was a post-treatment scan. All scans were within the pharynx region; some at the tonsil, some at the oropharynx, some at the base of the tongue, some at the pyriform sinus, some at the hypopharynx, and some at the nasopharynx. The scans had varying TNM staging, but with no metastasis. The various lesions in the scans add up to 59 primary tumors and 171 lymph nodes, for a total of 230 different lesions. The scans and individual objects all had varying characteristics. Some scans had only a single, obvious primary tumor, as in Figure 3.1a, while others had many lymph nodes in hard-to-separate clusters of lesions, as in Figure 3.1b.

Previously, a physician looked through all of the scans to identify center points of the different objects, as well as identify which was a primary tumor, and to determine the extent of the more complicated objects so that indicator images could be made. After those were reviewed and some center points revised, every primary tumor and lymph node in the scans was identified as a separate object.

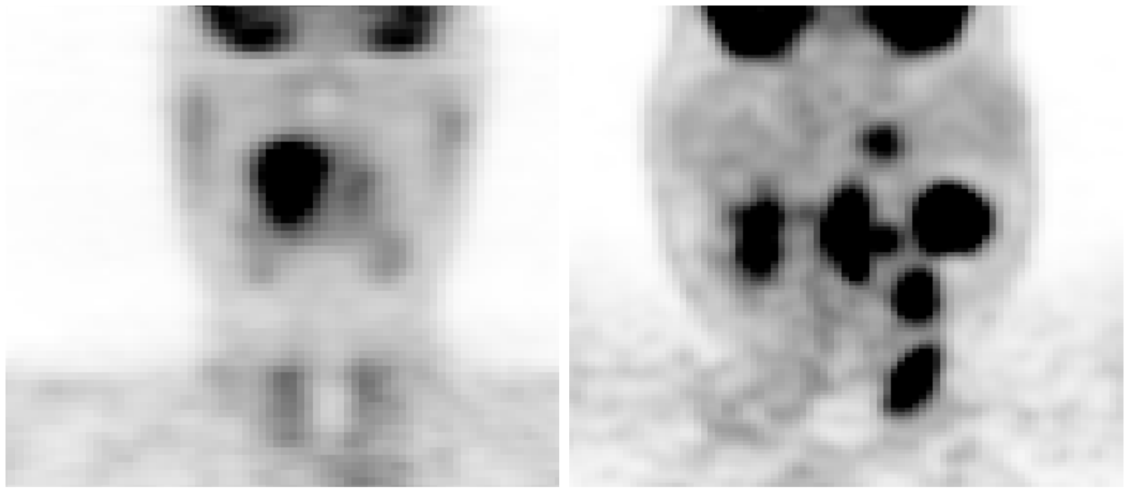

(a) (b)

Figure 3.1: Two different cases to compare complexity. (a) A simple case with a single primary. (b) A very complex case with multiple lesions, including primary cancer and hot lymph nodes, that must be to split apart.

3.1.1 FDG PET Experimental Methods

There are 60 scans, divided into three sets 20 scans, with each set of approximately the same overall complexity. With these scans, an experiment was performed that could directly test four things:

1. How accurately can an expert user make segmentations with the algorithm, compared to with manual delineation?
2. How consistent is an expert user making the same segmentation twice with the algorithm, compared to with manual delineation?
3. How consistent are two expert users making the same segmentation with the algorithm, compared to with manual delineation?
4. How quickly can an expert user make segmentations with the algorithm, compared to with manual delineation?

For each user, there was a randomized order for segmentation. Every other scan was segmented using a different method, starting with the semiautomated method. Every 20 segmentations made would cover all 20 scans in a case, half with one method and half with another. The next 20 segmentations would cover them with the previously skipped method, in a different order. The next 40 scans would be another ordering in the same vein. This occurred for all three sets of 20 scans. This results in 460 segmented lesions per method per user. To reduce the effect of learning to use the tool during the experiment, the users were given 10 data sets to train with that were not included in the experimental data, each segmented with the algorithm (with some guidance) and manually.

The experiment itself was conducted using 3D Slicer. The manual segmentations were made with a standard tool in Slicer that filled in a closed shape on an individual slice of a label volume. The semiautomated segmentations were made with a custom-built tool, shown in Figure 3.2, with specific checkboxes for the different modes of operation and refinement options in the algorithm.

To ensure reasonable use of the tool's various options, instructions were given regarding general choice of options for the tool:

1. Aim toward the center of the object. Use shift to move all views to it. Check the center with at least two views (shift on the center in one, then the other) to be sure your target is near the center.

2. Always use recentering (mentioned in Section 2.2.1), unless it causes serious problems.

3. For primary tumors (lesion 1), avoid splitting, if you can. This is because primary tumors are often large and inhomogeneous.

4. For lymph nodes (lesion 2 or higher), use splitting, if you can. It also adapts the shape to be more spherical overall. Note that edge refinement is generally

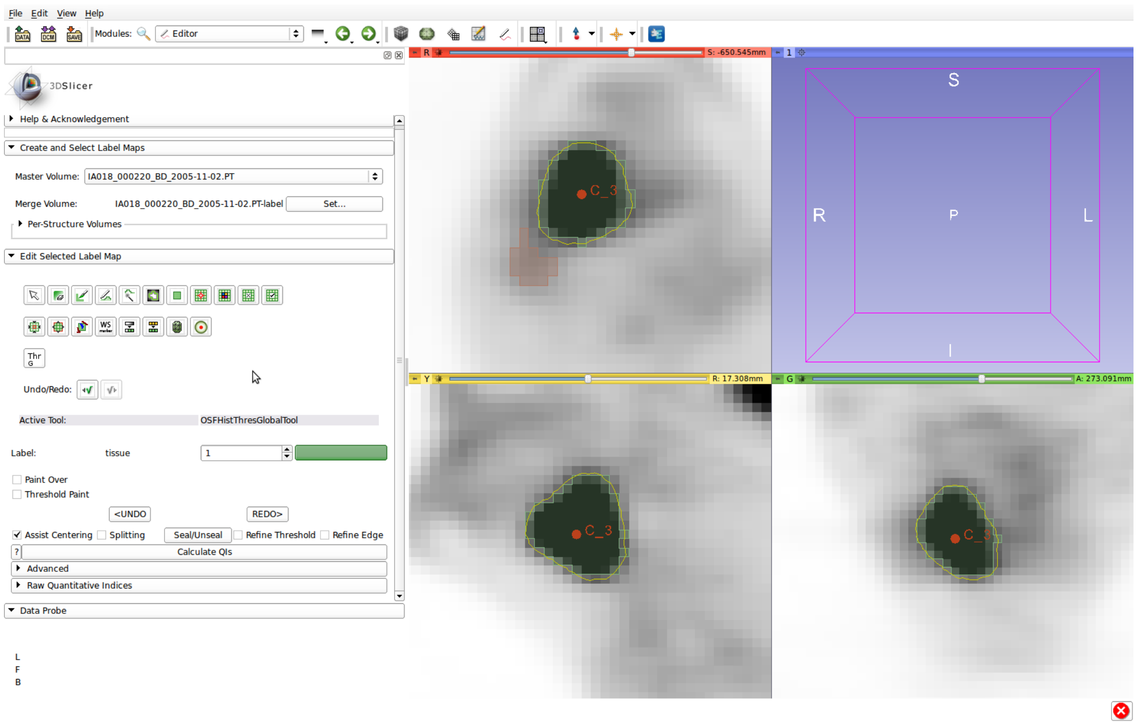

Master Volume: IA018_000220_BD_2005-11-02.PT
Merge Volume: IA018_000220_BD_2005-11-02.PT-label
Active Tool: OSFHistThresGlobalTool
Label: tissue 1
[✓] Assist Centering [ ] Splitting [ ] Refine Threshold [ ] Refine Edge
[ ] Paint Over
[ ] Threshold Paint

Figure 3.2: The 3D Slicer tool used for the experiment.

stronger with splitting on. This is because lymph nodes are frequently in closer proximity to other lymph nodes.

5. Label avoidance (Section 2.4.1) is on by default, and should be left on unless absolutely necessary.

6. If you can't get two objects to segment separately from each other, try another order. Smaller objects that are clearly separate are a good choice to start with. Larger objects that a smaller object seems arbitrarily partitioned off of should be done first to give a specific area for the smaller one, as well.

7. Use edge refinement for narrow changes in the surface and threshold refinement for widespread error due to bad threshold.

8. If you can't get all of a complex shape just with refinement, make a merged segmentation by just placing another center point. Note: Sealing will have to

be done for each center point before leaving it, if it is to be done.

9. Seal after other steps, if desired. Refining will undo sealing. Undoing to a previous state will undo sealing unless you just Redo.

10. When splitting apart lesions, you may modify the gray value transfer function to see the otherwise invisible uptake boundaries between lesions, but it must be returned to normal for segmentation afterwards to prevent added variation of the lesion-to-background surface.

11. When attempting to segment a highly necrotic object, standard methods generally don't work well.

    (a) Segment the high uptake (dark) parts as normal, as if the low uptake (light) parts aren't there.

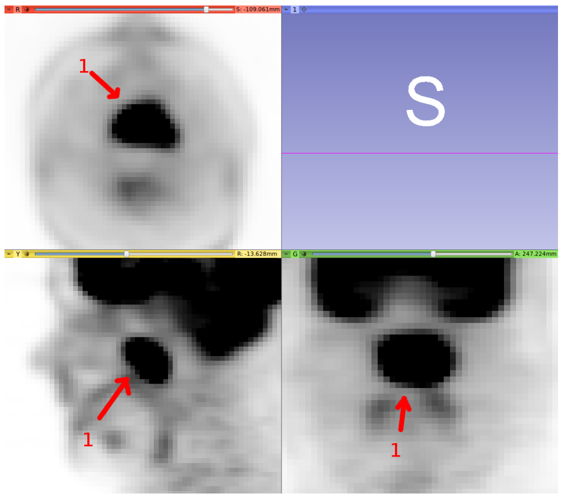
    (b) With label avoidance on, and recentering and splitting all off, open the Advanced menu and check Necrotic Region.

    (c) Aim toward the center of the low uptake parts and make a merging segmentation. The automatic threshold will quite probably be off.

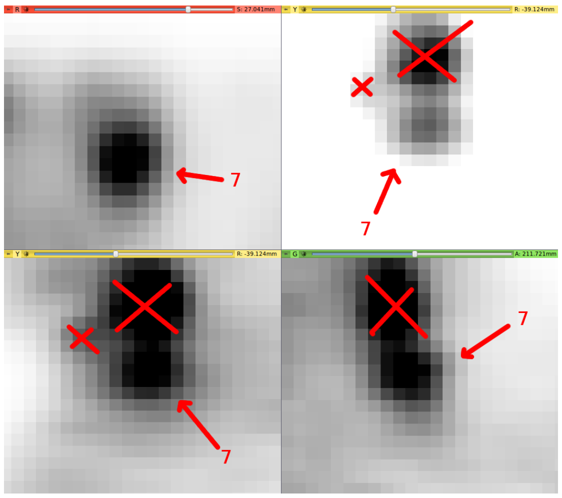
    (d) Refine as needed to include any necessary fringes and to connect to the high uptake parts.

    (e) Seal to complete.

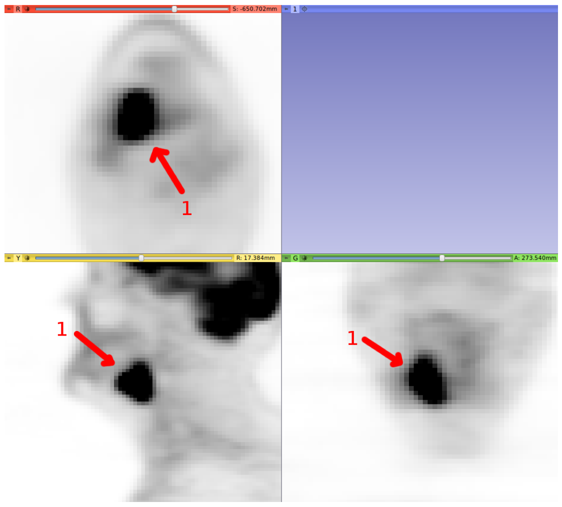
For each case, there was a set of images to specifically, but imprecisely identify the objects to be segmented and the appropriate labels for them, which also implied the type of lesion; object 1 was always the primary tumor. All of these were made manually, with enough information to understand what the intended extend was, but not enough to give away a specific center point or boundary point. A few examples are available in Figure 3.3.



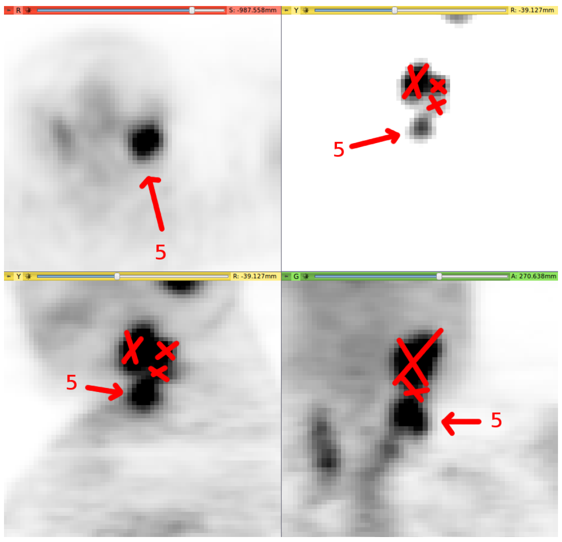
(a)



(b)



(c)



(d)

Figure 3.3: Indicators for various objects, some simple and some complex.

3.1.2 Performance Metrics

The Dice coefficient is used to assess volumetric overlap. It is used here to measure consistency through reporting similarity between distinct trials of individual lesions. It is also used to compare individual trials to an independent image. This comparison, along with percentage volume error, is used to measure accuracy of segmentations. Given two segmentations, A and B, the Dice coefficient DC is twice the size of their intersection over the sum of each ones size:

$$DC = \frac{2|A \cap B|}{|A| + |B|}. \tag{3.1}$$

Due to the lack of a ground truth for the real data, the independent image used to measure accuracy is instead built as a consensus of manual segmentations from the users with the Simultaneous Truth and Performance Level Estimation (STAPLE) algorithm from Warfield's paper in 2004 [18]. For each lesion, there were three such consensus images generated: one of both users' manual segmentations (the 4-image consensus) and one each for the individual users' manual segmentations (the 2-image consensuses). The result STAPLE gives is a series of probabilities, thresholded at 0.50 to make the final consensus lesion. Figure 3.4 shows one such consensus and its source parts.

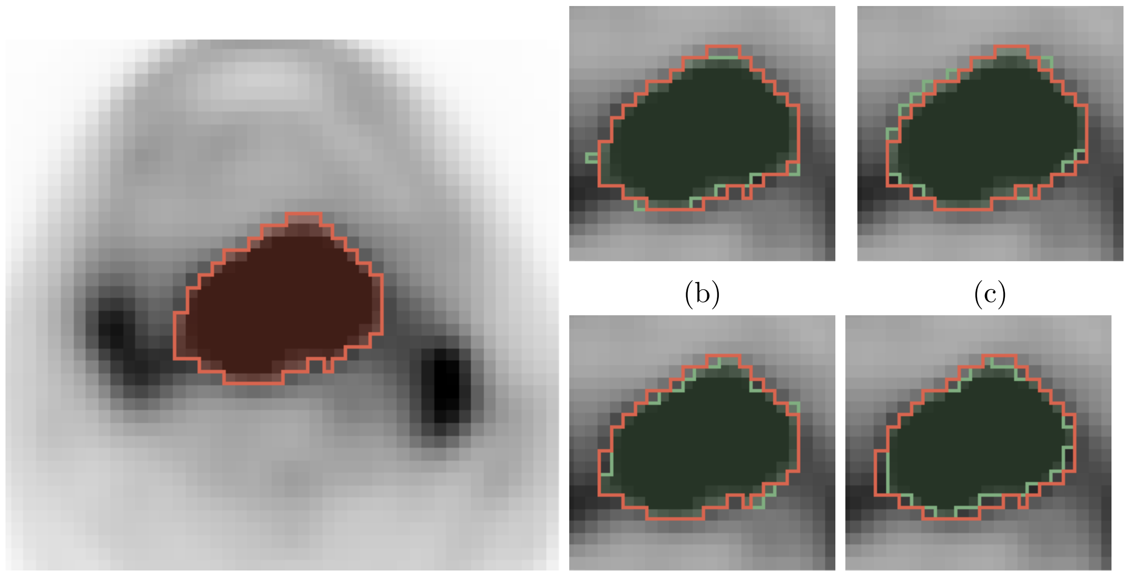


Figure 3.4: A 4-image consensus image compared to its components. (a) The consensus image. (b), (c), (d), (e) The component images (green), with the outline of the consensus (red) shown over them.

Time is used, per complete contouring for all lesions in the scan as measured and per lesion by dividing the time for a scan by the number of lesions, to determine speed. Additionally, the number of actions, including or excluding discarded actions, also gives a measurement for speed.

3.1.3 FDG PET Experimental Results

From the experiment, the time to segment all objects in a scan was recorded for each method and trial, along with the various tools used for the semiautomated algorithm, and the segmentation image itself. From this series of experiments, data relevant to the four questions for the validation has been assembled.

The first is an accuracy determination. For this, the segmentations are all compared to a consensus image, to measure how the manually generated contours compare to the semiautomated contours. The second and third measures are both consistency results and are both important. For these, individual lesions from segmentations are simply compared to the same lesions of other segmentations for the same scan, either for the same user or between trials of between users. The fourth measurement is a time measurement, nominally the speed of accomplishing the contouring task. For this, the manual and semiautomated times for cases are compared, per scan and averaged per lesion. Furthermore, the number of distinct actions per lesion for the semiautomated tool is included for time measurement.

3.1.3.1 Accuracy of Segmentation

Determining accuracy requires some kind of independent reference standard. The consensus images mentioned in Section 3.1.2 are used for this purpose. Semiautomated segmentations are compared to the 4-image consensus images. In addition, the manual segmentations are compared to the 2-image consensus from the opposite user to avoid bias issues, these conensuses are made using the manual segmentations themselves. Since the ground truth each is being compared to is different, a direct comparison of manual similarity to consensus and semiautomated similarity to consensus is not helpful, and so no test is included comparing manual and semiautomated results. The Dice coefficient comparisons between segmentations are shown in Figure 3.5 and Table 3.1.

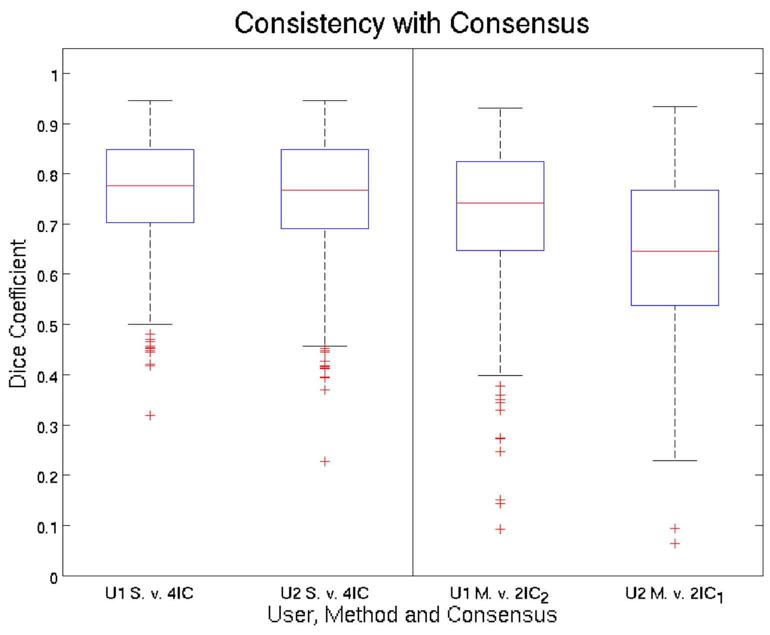


Figure 3.5: The similarity of the segmentations to consensus images. Semiautomated segmentations are compared to the 4-image consensus. Manual segmentations are compared to the opposite user's 2-image consensus.

Table 3.1: Similarity of Segmentations vs. Consensus

	Dice Coefficient	
Segmentation	Median	Mean ± Std. Dev.
User 1 Semiauto	0.7755	0.7640 ± 0.1113
User 2 Semiauto	0.7670	0.7574 ± 0.1207

Segmentation volume is a common measurement of lesion size, so it's useful to compare the volumetric error against a consensus image. These can help determine if there's any consistent bias compared to the consensus image. Figure 3.6 and Table 3.2 show this. Lastly, it's of interest to compare the percentage error to the volume of each object, to check for trends between the two. This comparison is shown in Figures 3.7 and 3.8.

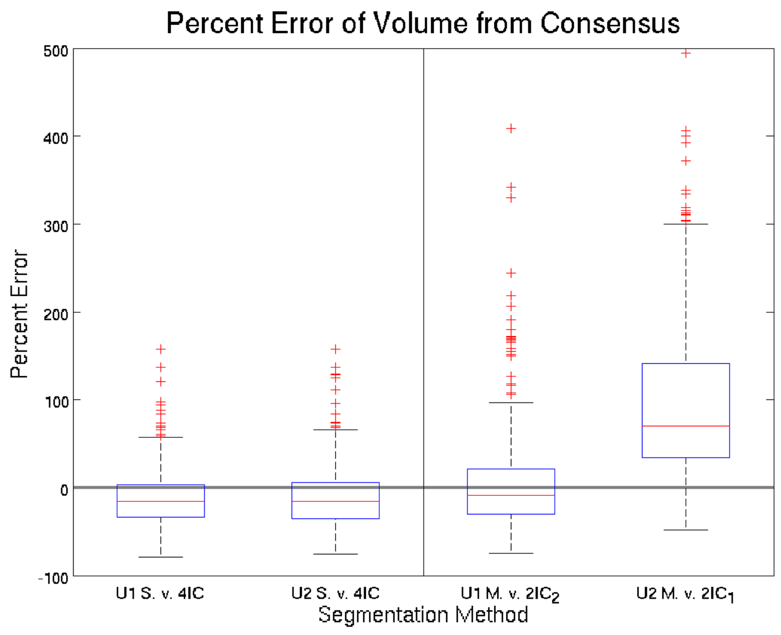
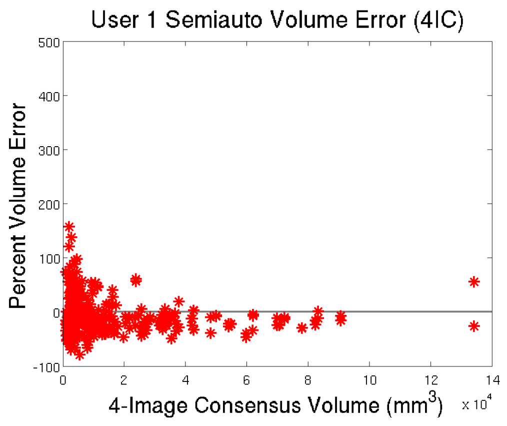


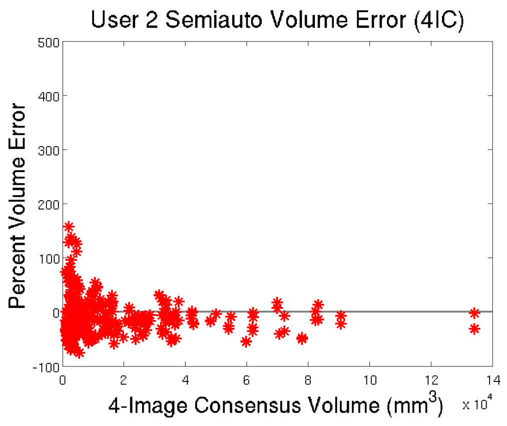
Figure 3.6: The percentage error in volume of the segmentations. Semiautomated segmentations are compared to the 4-image consensus. Manual segmentations are compared to the opposite user's 2-image consensus.

Table 3.2: Percentage Volume Error of Segmentations vs. Consensus

	Percent Error	
Segmentation	Median	Mean ±Std. Dev.
User 1 Semiauto	-15.67%	-10.21% ± 33.81%
User 2 Semiauto	-15.56%	-10.43% ± 34.08%

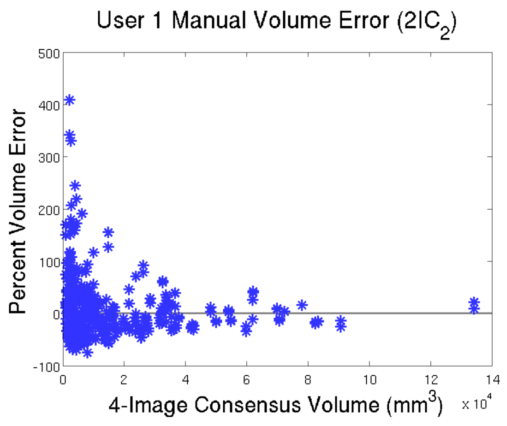


(a)

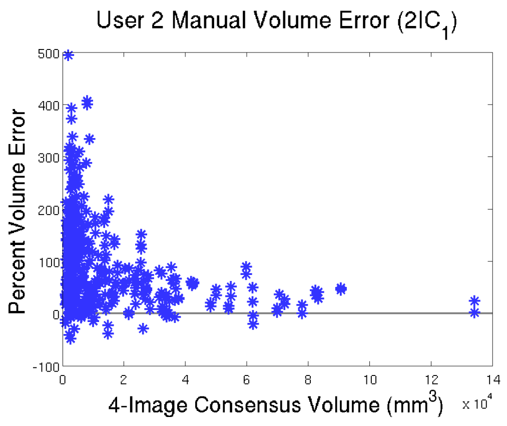


(b)

Figure 3.7: The percentage volume error for semiautomated segmentations compared to the volume of the lesion. (a) Semiautomated results for user 1 vs. 4-image consensus. (b) Semiautomated results for user 2 vs. 4-image consensus.



(a)



(b)

Figure 3.8: The percentage volume error for manual segmentations compared to the volume of the lesion. (a) Manual results for user 1 vs. 2-image consensus of user 2. (b) Manual results for user 2 vs. 2-image consensus of user 1.

3.1.3.2 Intraoperator Consistency

For a single user, a comparison is made between the first and second trial for every lesion with each method to get a Dice coefficient per lesion, method and user. The results are in Figure 3.9 and Table 3.3.

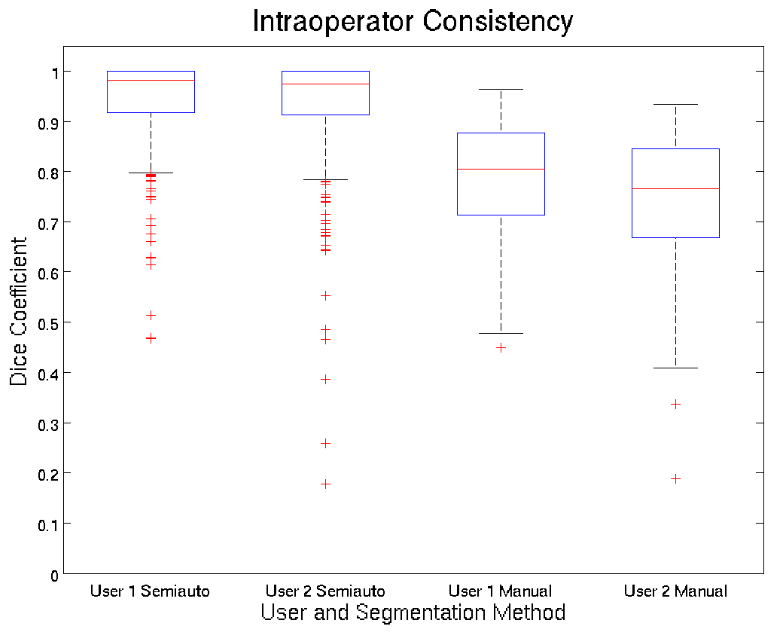


Figure 3.9: The consistency between trials for each user and each segmentation method.

Table 3.3: Intraoperator Consistency

	Dice Coefficient	
Segmentation	Median	Mean ± Std. Dev.
User 1 Semiauto	0.9814	0.9373 ± 0.0982
User 2 Semiauto	0.9736	0.9254 ± 0.1256
User 1 Manual	0.8040	0.7881 ± 0.1108
User 2 Manual	0.7663	0.7513 ± 0.1234

The paired signed rank test gives probabilities $p_{u1} << 0.05$ and $p_{u2} << 0.05$ of the difference between manual and semiautomated distributions per user being coincidence, implying them to be significantly different. The result is in favor of the semiautomated segmentations having higher consistency. Figure 3.10 gives an example comparing consistency of manual segmentations by one user to consistency of semiautomated segmentations by the same user.

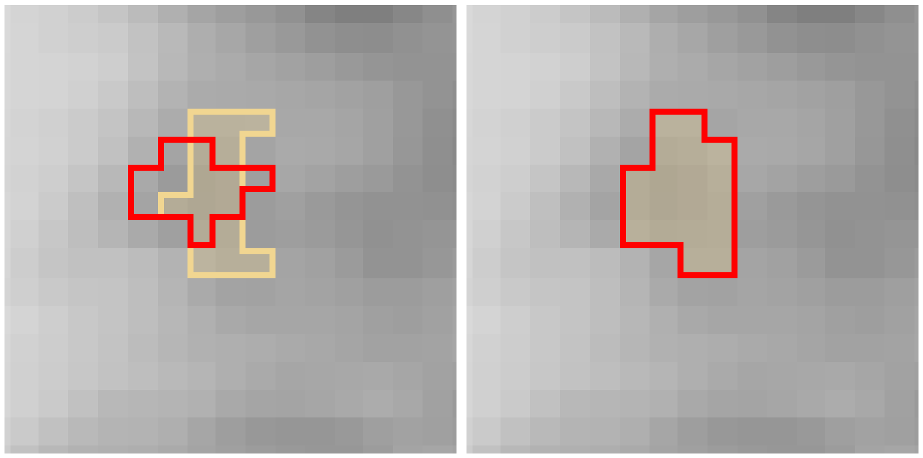

(a) (b)

Figure 3.10: An example of intraoperator consistency and inconsistency. The first trial is outlined and slightly shaded yellow. The second trial is outlined red. (a) The manual segmentations are significantly different. (b) The semiautomated segmentations are identical.

3.1.3.3 Interoperator Consistency

A similar test is used to compare consistency between users. Because of how users can have a learn between trials, these comparisons are for the same trial for either user: trial 1 to trial 1, trial 2 to trial 2. The results are in Figure 3.11 and Table 3.4.

Table 3.4: Interoperator Consistency

	Dice Coefficient	
Segmentation	Median	Mean $\pm$ Std. Dev.
Semiauto	0.9610	0.9091 $\pm$ 0.1301
Manual	0.6912	0.6902 $\pm$ 0.1317

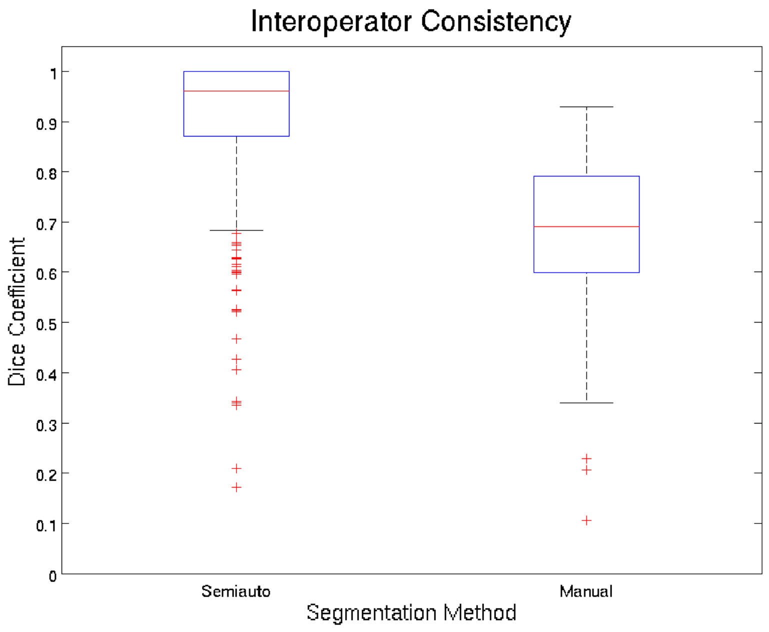


Figure 3.11: The consistency between users for each segmentation method.

The paired signed rank test comparing automated to manual results gives a probability $p << 0.05$ of the differences in results being coincidence, implying that the semiautomated segmentations having a significantly different (higher) Dice coefficient. Figure 3.12 shows an example of the consistency of manual segmentations and semiautomated segmentations between users.

Besides this, the interoperator variability can be seen in the accuracy results from Section 3.1.3.1. From the similarity results with consensus images from Figure 3.5 and Table 3.1, a signed rank test gives $p_m << 0.05$ chance of the difference between values for the manual results being a coincidence, while another signed rank test gives $p_s = 0.0502 > 0.05$ chance for the same difference between semiautomated results being coincidence. The two users' manual results' consensus similarity is significantly different, while their semiautomated results' consensus similarity is not. For volumetric error percentage from Figure 3.6 and Table 3.2, the signed rank test gives $p_m << 0.05$ that the manual results are different by coincidence, while they

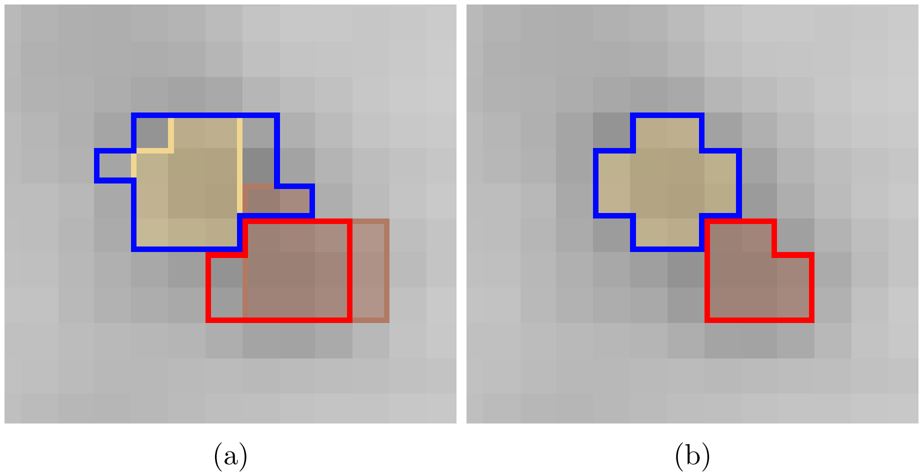


Figure 3.12: Two segmented lesions, with their other users' outlines in red or blue. (a) The manual segmentations are significantly different. (b) The semiautomated segmentations are identical.

give $p_s = 0.6648 > 0.05$ chance of coincidence that the semiautomated results differ by coincidence. Again, the two users' manual results are significantly different from each other, while their semiautomated results are not.

3.1.3.4 Time for Segmentation

Time was measured for each complete segmentation, but by counting lesions per segmentation, an average time per lesion can be calculated (for each scan). Multiplied by the scans on that lesion, an average time per lesion is calculated for all lesions across all scans. The overall data per segmentation is in Figure 3.13. Table 3.5 has the time per segmentation and the calculated time per lesion.

Scans vary significantly in the number and complexity of objects, as well as the stage of the tumors. The paired signed rank test results were $p_{u1} << 0.05$ and $p_{u2} << 0.05$ chance that the differences between manual and semiautomated times were coincidence. In each case, the signed rank implied that the semiautomated times for that user were significantly lower.

Also of note is the number of actions for a segmentation. This varies with

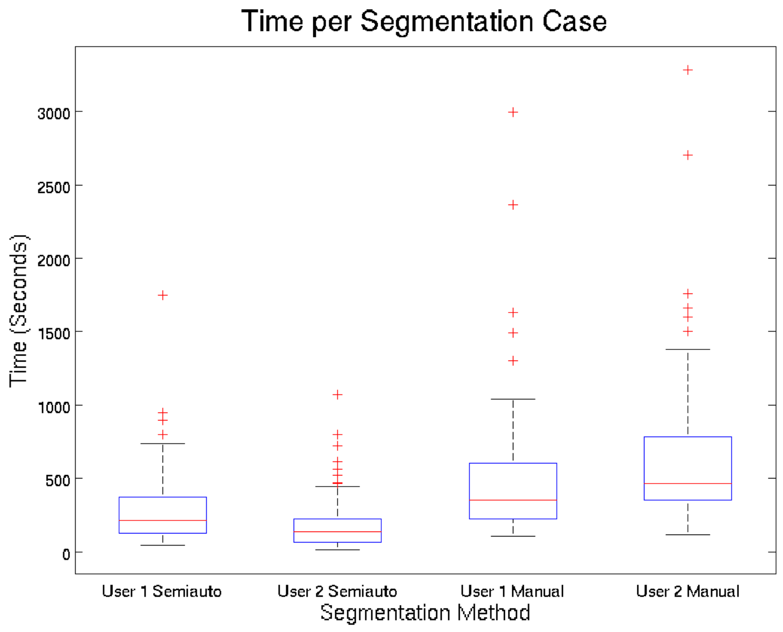


Figure 3.13: The time to segment all objects in a scan, for both methods and both users.

Table 3.5: Time for Segmentation and Individual Lesions

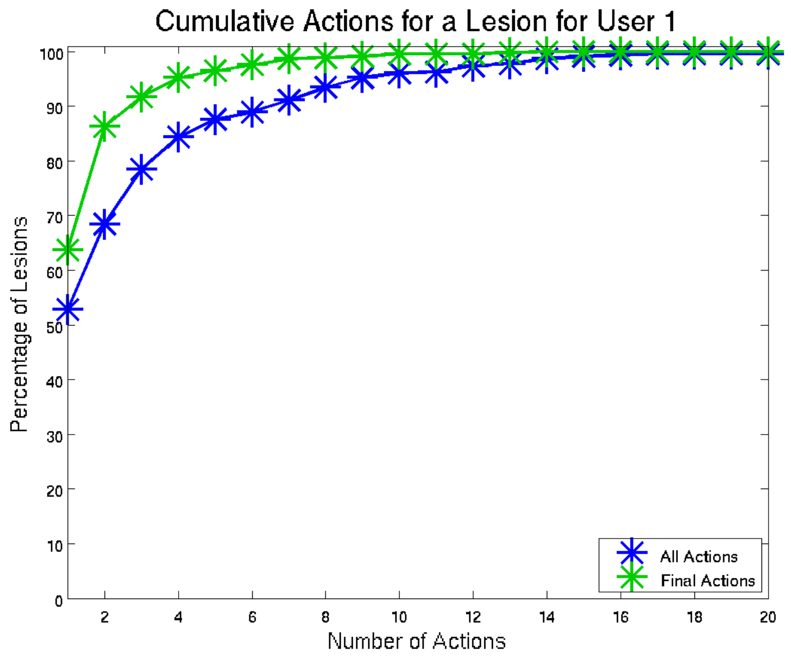
		Seconds	
Segmentation	Scope	Median	Mean ± Std. Dev.
User 1 Semiauto	Segmentation	211.98	277.45 ± 234.47
User 2 Semiauto	Segmentation	134.10	177.91 ± 174.29
User 1 Manual	Segmentation	352.47	477.59 ± 412.14
User 2 Manual	Segmentation	463.20	623.49 ± 471.80
User 1 Semiauto	Lesion	41.42	72.38 ± 90.06
User 2 Semiauto	Lesion	27.01	46.41 ± 57.49
User 1 Manual	Lesion	70.59	124.59 ± 160.41
User 2 Manual	Lesion	98.71	162.65 ± 185.55

how the user applies the tool, and also with the inclusion or exclusion of discarded actions, which were undone via the tool's built in Undo/Redo queue. Figure 3.14 depicts the number of actions, including and excluding discarded actions. Table 3.6 also has the information.

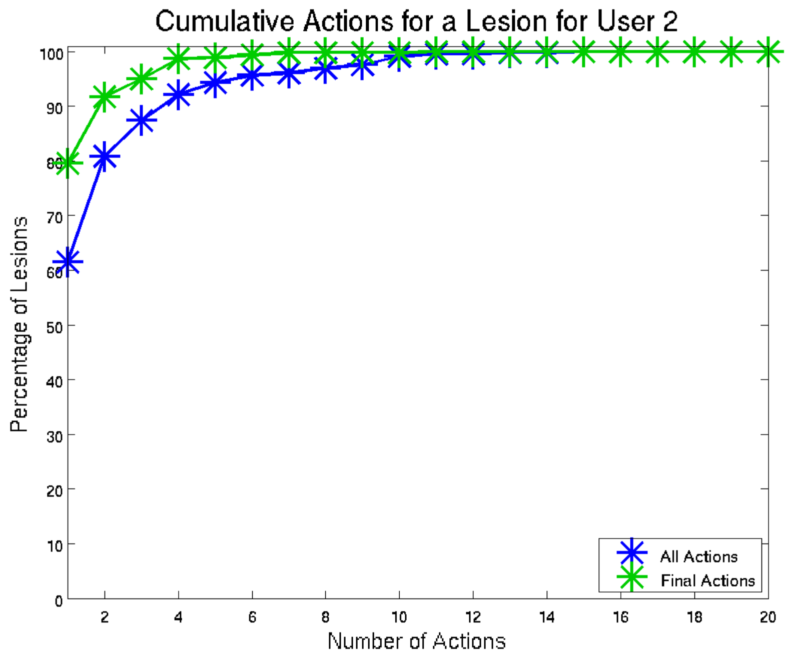
Table 3.6: Actions for a Lesion

		Number of Actions			Percent of Cases Segmented	
User	Actions	Median	Mean	For 90%	W/ 1 Action	W/ 2 Actions
User 1	All	1	2.7935	7	52.83%	68.48%
User 2	All	1	1.9957	4	61.52%	80.87%
User 1	Final	1	1.7370	3	63.70%	86.30%
User 2	Final	1	1.3761	2	79.57%	91.74%

Figure 3.15 has examples of lesions segmented with one action. Figure 3.16 has examples of lesions that needed a relatively large number of actions to segment.



(a)



(b)

Figure 3.14: Percentage of lesions completed with each number of actions, including those that were discarded (blue) and excluding them (green). (a) Plot for user 1. (b) Plot for user 2.

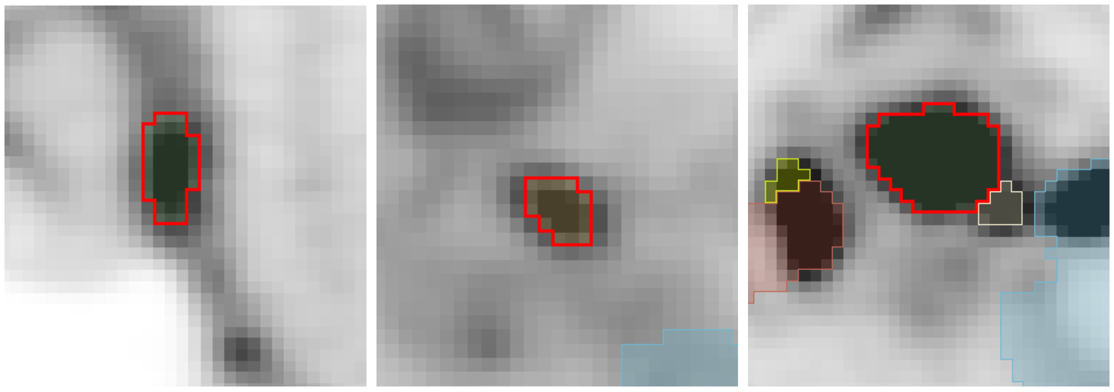

(a) (b) (c)

Figure 3.15: Lesions that required only one action. The specific lesion is outlined in red. (a) A simple object on its own. (b) Again, a simple object on its own. (c) Near another object, but with a distinct enough natural boundary.

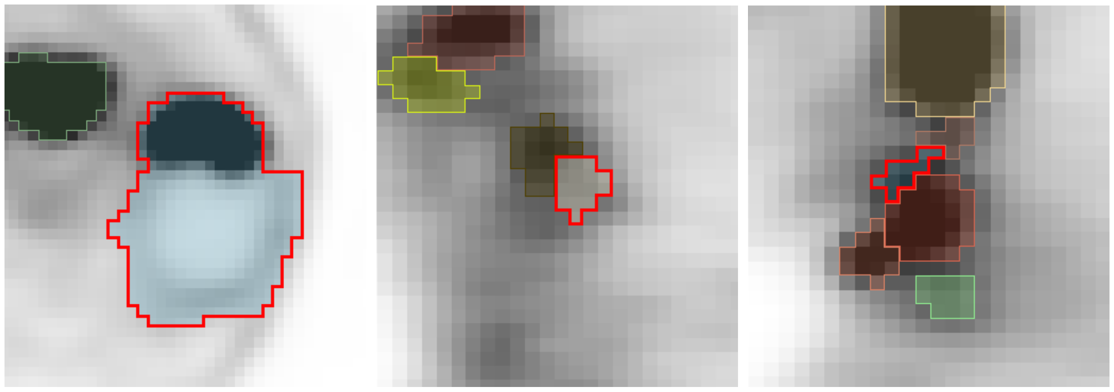

(a) (b) (c)

Figure 3.16: Lesions that required many actions. The specific lesion is outlined in red. (a) Complex due to the large necrotic region and the border around it. Resolved with 25 actions, six of which were final. (b) Complex due to an adjacent object with uncertain boundary and the uncertain boundary with the background. Resolved with four actions, all four of which were final. (c) Complex due to interaction with multiple nearby lesions and the awkward shape lead to many actions that were applied and undone. Resolved with 16 applied actions, only one of which was final.

3.2 Phantom Data

A set of 44 different scans was used for phantom data testing. There are two subsets based on different reconstructions, each with 22 volumes. The first set of 22 was very noisy, with frequent high uptake voxels, and voxel size of 2.7 × 2.7 × 3.3 mm. This set was provided by the University of Washington. The second set of 22 was essentially the opposite, instead with a reconstruction algorithm that heavily smoothed out all of the ”lesions”, and voxel size of 3.4 × 3.4 × 2.0 mm. This set was provided by the University of Iowa. For each set of 22, 11 are low contrast and 11 are high contrast. For those sets of 11, one has longer scan time (or high-statistics) and the other ten have shorter scan time (or low-statistics). This makes up to eight noticeably different scan varieties. Furthermore, each volume has six ”lesions” to use. An overview of the eight different scan varieties available is shown in Figure 3.17.

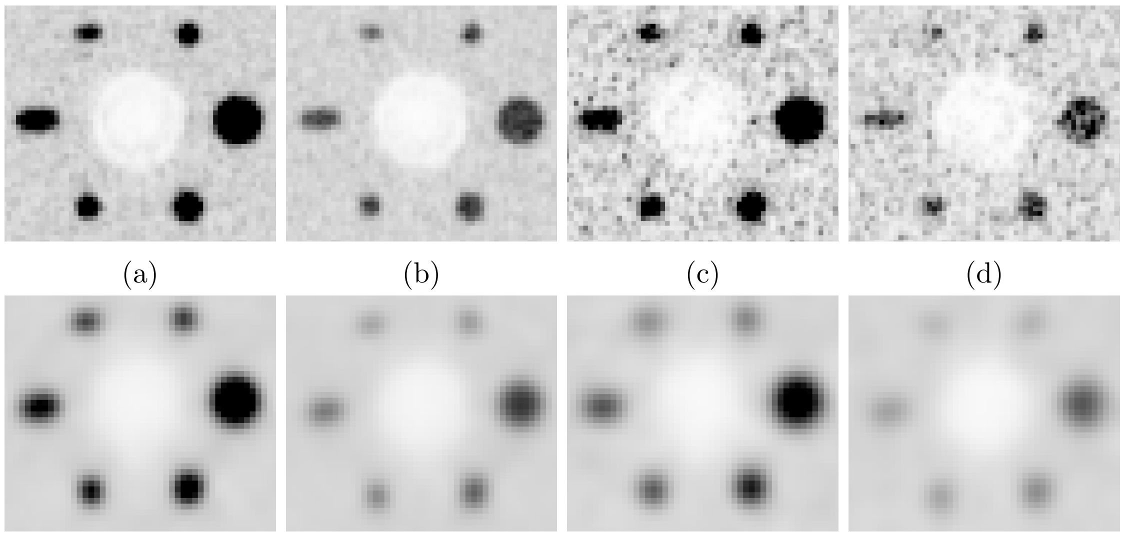


Figure 3.17: An overview of the various phantom images used. (a) Washington, high statistics, high contrast. (b) Washington, high statistics, low contrast. (c) Washington, low statistics, high contrast. (d) Washington, low statistics, low contrast. (e) Iowa, high statistics, high contrast. (f) Iowa, high statistics, low contrast. (g) Iowa, low statistics, high contrast. (h) Iowa, low statistics, low contrast.

Each of the six lesions was a different size. With regards to volume, Lesion 1 $>$ Lesion 2 $>$ Lesion 3 $=$ Lesion 4 $>$ Lesion 6 $>$ Lesion 5:

1. Lesion 1 is a sphere of radius 14 mm.

2. Lesion 2 is a vertical ellipsoid of radii $8.5 \times 8.5 \times 17$ mm.

3. Lesion 3 is a vertical ellipsoid of radii $6.5 \times 6.5 \times 13$ mm.

4. Lesion 4 is a horizontal ellipsoid of radii $13 \times 6.5 \times 6.5$ mm.

5. Lesion 5 is a horizontal ellipsoid of radii $10 \times 5 \times 5$ mm.

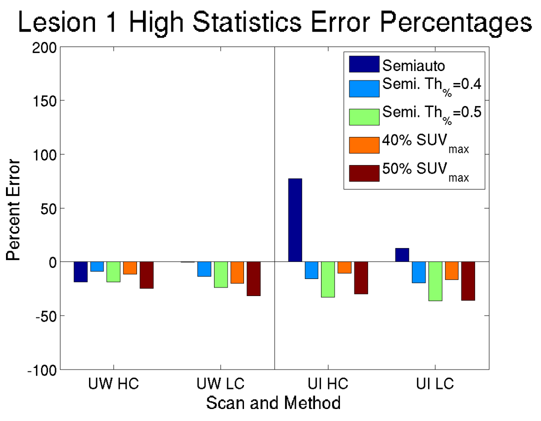
6. Lesion 6 is a sphere of radius 6.5 mm.

For this study, all the adaptations from Section 2.5 are applied. The linear background adaptation helps deal with the smoothed out data from the University of Iowa phantoms and the minimal uptake transitions in all the high-statistics phantom scans, while the threshold de-noise adaptation helps deal with the noise in the University of Washington phantoms, though both were applied for all phantom images. Furthermore, the alternatives for calculating $Th_{\%}$ discussed near the end of Section 2.2.3, $Th_{\%} = 0.4$ and $Th_{\%} = 0.5$, have been included to study the impact of the threshold selection method. For comparison, segmentations were made using standard 40% of maximum uptake and 50% of maximum uptake methods, each on a background SUV of 1.0.

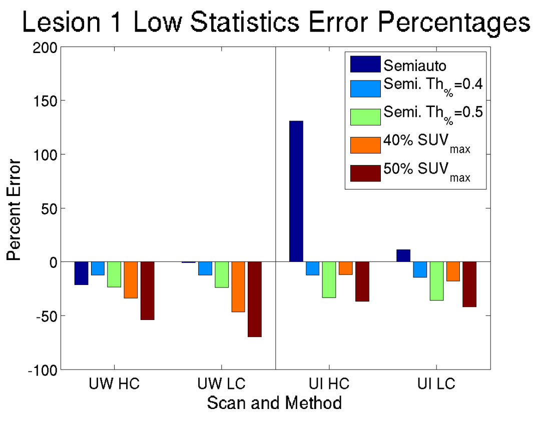
With the appropriate settings on, the segmentation was made by placing a $ce_{k_{user}}$ approximately in the center of the lesion with recentering active. For two specific instances, an additional variant was made with a threshold refinement point.

3.3 Phantom Experimental Results

The percentage error in volume was computed for each segmentation of each lesion. For the high-statistics phantoms, there was only a single such scan. For the low-statistics phantoms, there were 10 scans, of which the median percentage error is shown in the following graphs. The terms ”UW HC” and similar refer to the originating university and the contrast level, so ”UW HC” would be the high contrast scan from the University of Washington, the noisier of the two sets. ”UI LC” would be the low contrast scan from the University of Iowa, the smoother of the two sets. These results are shown in Figures 3.18 through 3.23.

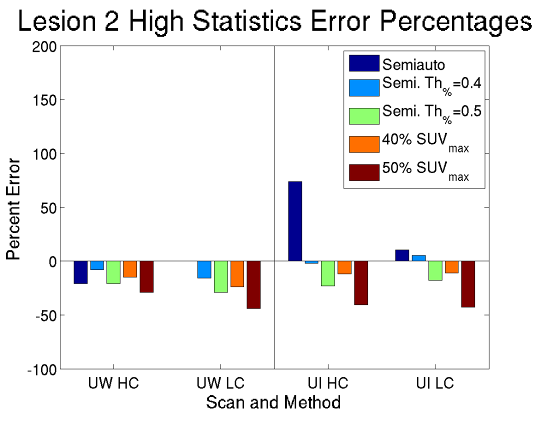


(a)

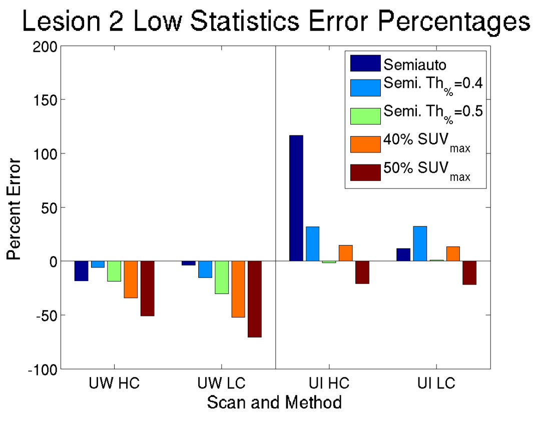


(b)

Figure 3.18: The percentage error of the segmentations for lesion 1 of the phantoms. (a) High statistics. (b) Low statistics, median of 10 copies.

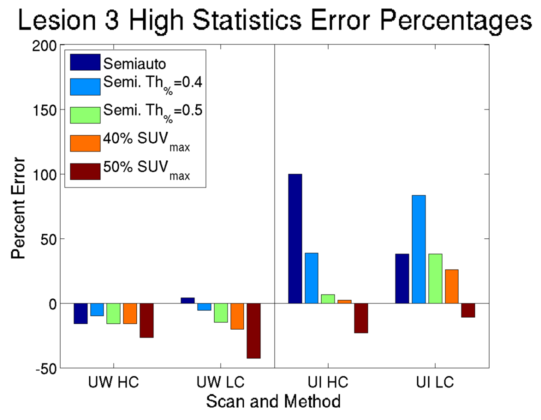


(a)

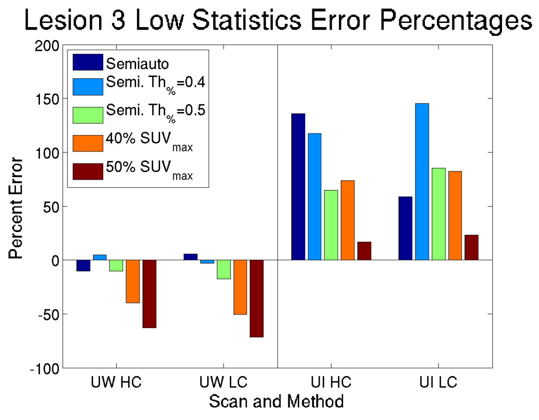


(b)

Figure 3.19: The percentage error of the segmentations for lesion 2 of the phantoms. (a) High statistics. (b) Low statistics, median of 10 copies.

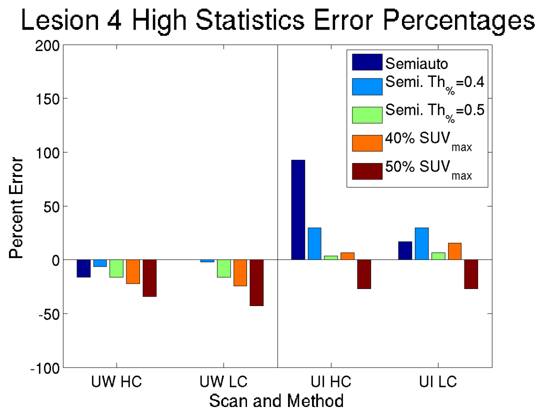


(a)

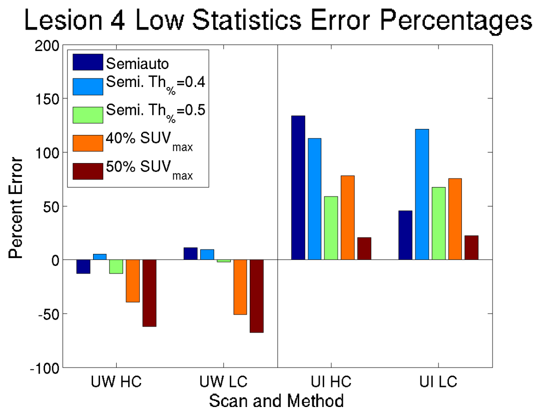


(b)

Figure 3.20: The percentage error of the segmentations for lesion 3 of the phantoms. (a) High statistics. (b) Low statistics, median of 10 copies.



(a)



(b)

Figure 3.21: The percentage error of the segmentations for lesion 4 of the phantoms. (a) High statistics. (b) Low statistics, median of 10 copies.

In addition, for lesion 5, high statistics, simple threshold refinement was applied, when relevant. The result of this is in Figure 3.24. Since the University of Washington phantoms were already approximately accurate, this was only performed on the Univeristy of Iowa phantoms.

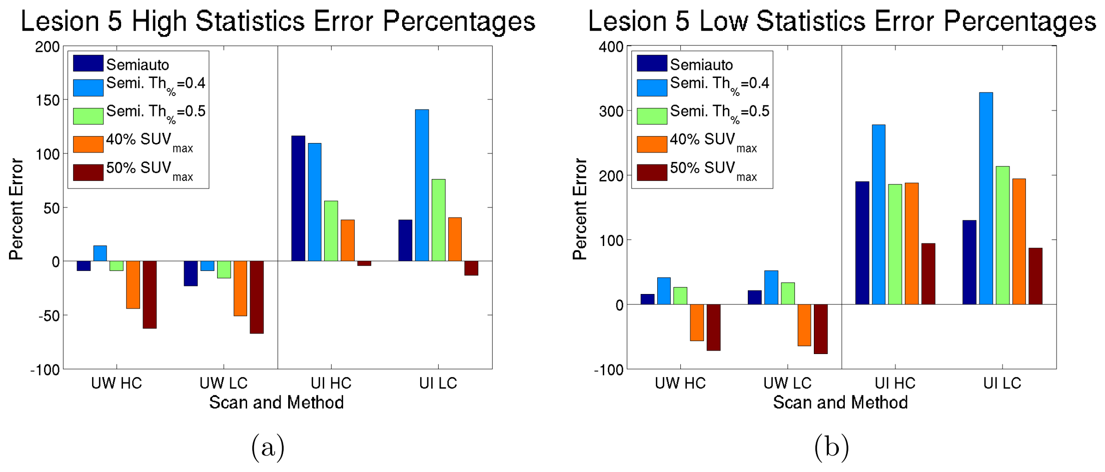


Figure 3.22: The percentage error of the segmentations for lesion 5 of the phantoms. (a) High statistics. (b) Low statistics, median of 10 copies.

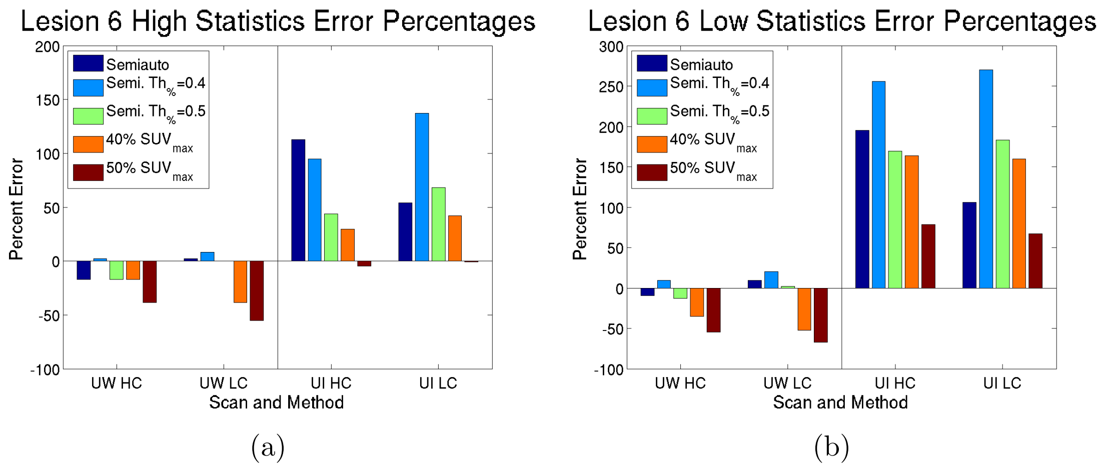


Figure 3.23: The percentage error of the segmentations for lesion 6 of the phantoms. (a) High statistics. (b) Low statistics, median of 10 copies.

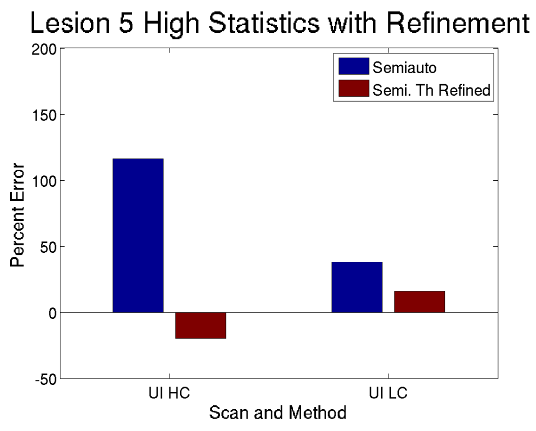


Figure 3.24: The percentage error of the segmentations for lesion 5 of the University of Iowa phantoms for high statistics, using the normal algorithm with and without threshold refinement.

CHAPTER 4
DISCUSSION

With the results of the tests of the algorithm available, the results can be discussed and a determination made on the success of the algorithm on meeting the goal from Section 1.3. This is determined by the results from both the real data tests and the phantom data tests.

4.1 Segmentation Performance

4.1.1 Real Data

Overall, the algorithm was designed using real PET data, so that's where the bulk of the experimental data and discussion comes from. Ultimately, these help to determine accuracy, consistency, and speed.

4.1.1.1 Accuracy

Accuracy is the most complicated aspect to review, for reasons already mentioned related to finding a "correct" segmentation. For real data, the consensus images are decent, but probably would do better with more users available.

From the results in Section 3.1.3.1, a few things are clear. Figure 3.5 shows the similarity the semiautomated method has with the 4-image consensus to be as good or better than the similarity the manual method has with the 2-image consensus. A direct comparison, however, is very difficult to justify, due to the difference in ground truth between them. Similar results are gleaned from Figure 3.6. In both cases, the semiautomated segmentations have much lower standard deviation for results, but this is likely affected as well by the different ground truth. An ideal comparison for accuracy would require much more elaborate statistical analysis to determine.

For comparison, the threshold-based region growing methods of Li's 2008 work [10] had a best average volumetric error of 11.0% and a worst of -79.0% across the phantom scans included. The best results are comparable to the percent error for the semiautomated results, -10.2% for user 1 and -10.4% for user 2, a slight underestimation as opposed to the slight overestimation in Li's work, but still similar. The hybrid PET-CT segmentation methods from Han's 2011 study [7] had a Dice coefficient of 0.86 ± 0.051 on their introduced method and 0.78 ± 0.045 on a pure PET method. The pure PET method is comparable to the semiautomated method here with Dice coefficients 0.76 ± 0.11 for user 1 and 0.76 ± 0.12 for user 2. These results aren't directly comparable due to the significant differences in data and circumstances, but give an idea of where the algorithm stands.

Figures 3.7 and 3.8 give some insight into the outliers seen for all users and methods. Over- and underestimation occurred more frequently for the lowest-volume lesions in the data sets. This caused the relatively high standard deviation in the percent volume error. On a small volume, even a few voxels difference is a much greater percentage. This effect was more muted for the semiautomated segmentations, but that may be because of the difference in ground truth. Overall, for real data, accuracy for the semiautomated segmentations is reasonable, and not too different from that of the manual segmentations.

4.1.1.2 Consistency

For consistency, there is no question that the semi-automated method is superior. The results presented in Section 3.1.3.2 and Section 3.1.3.3 are self explanatory, with fairly large and statistically significant increases in consistency by using the semiautomated method over manual segmentation, and no statistically significant difference in accuracy between users for the semiautomated method.

For intraoperator variation, 43.9% of the lesions segmented by user 1 and 40.9% of the lesions segmented by user 2 were identical between trials in the semiautomated segmentations, while none of them were for either user in the manual segmentations. For interoperator variation, 35.7% of the lesions of the same trial were identical between users in the semiautomated segmentations, again with none of them identical in the manual segmentations. Interoperator varation was slightly higher than intraoperator variation, as evidenced by those numbers, but that's somewhat to be expected. Even with that slightly greater difference, the interoperator variation with the semiautomated method was lower than either user's intraoperator variation with the manual method.

The ultimate result of this is that the algorithm presented here significantly improves consistency over manual segmentation, both intra- and interoperator.

4.1.1.3 Speed

Speed is another performance metric where the semiautomated algorithm has a clear advantage. Figure 3.13 and Table 3.5 show that, for each user, the time was significantly lower for the semiautomated method.

The differences between users are more pronounced for these results. User 1 was more experienced in the manual tracing at the start of the process, and it shows with the noticeably lower time for manual segmentations as compared to user 2, who had less experience with it. User 2, on the other hand, more readily adapted to quickly using the semiautomated tool, if with slightly lower overall consistency than user 1, and thus had a much greater difference in time between manual and semiautomated segmentation. As seen in Table 3.6 and Figure 3.14, both users completed the majority of lesions in a singular action, but user 2 typically had to perform fewer actions per lesion. Despite this, looking at Table 3.1, the two users semiautomated results were both nearly equivalent in their accuracy. As such, with greater experience in using the algorithm, a user could certainly take fewer actions to get just as good of results, as user 2 was able to.

Even with this variation between users in tool use, the time difference between methods was still significant for each user. The maximum time for the either user's semiautomated segmentation was far below the maximum time for either's manual segmentation, and the medians for each were both below, user 2's especially due to the aforementioned reasons. In total, of the 120 segmentations per user and method, user 1 had six segmentations, all from distinct data sets, that were completed faster with the manual method and user 2 had exactly one such segmentation. On average, user 1 was about 40% faster with the semiautomated method and user 2 was about 70% faster.

As such, it is safe to say that the segmentation algorithm was indeed much faster than manual segmentation of the same cases, and therefore is a faster method overall.

4.1.2 Phantom Data

The phantom data tests focus on something very different from the real data tests. The base algorithm was based on oncology segmentations, which tends to err on the side of overstimating size when the overall image is smoother. Other variants tested showed differing results for the various scan types. What this test shows is how well strategies may work across different scanners and protocols. Figures 3.18 through 3.23 show the results of the various segmentation methods.

While the algorithm using Equation 2.14 to calculate $Th_{\%}$ is respectable across all "lesions", statistics, and contrast levels for the noisier phantom, it is shown to overestimate the volume in washed out phantom scans like those from University of Iowa. The high contrast phantoms in particular have more uptake leaking beyond the actual boundaries of the ”lesion”, which the algorithm includes based on its design. While it’s a decent function for the imaging of the real data, it’s useful to have other variants, such as the $Th_{\%} = 0.4$ and $Th_{\%} = 0.5$ versions of the algorithm, to better deal with other scanners or protocols. These other variants are better for the larger objects in the washed out scans, they still may not work particularly well on the smallest ones. However, they demonstrate significantly different behavior for volume measurement, showing how a change in the $Th_{\%}$ equation, while keeping the base algorithm framework intact, can allow it to be adapted to different scanners and protocols.

For comparison, typical threshold methods are also included. For the larger objects, the variants of the algorithm perform approximately as well or better than those, while their performance falls off for smaller objects. That said, performance in general tends to fall off for smaller objects (in the real data, Figures 3.8 and 3.7).

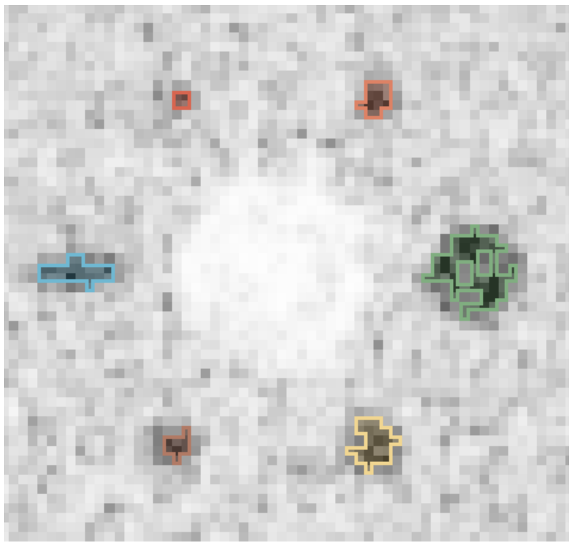


Figure 4.1: Examples of pure threshold segmentations of the high-noise phantoms. Lone high-uptake voxels can result in a threshold that is wildly off, while lone low-uptake voxels can result in holes on the segmentation.

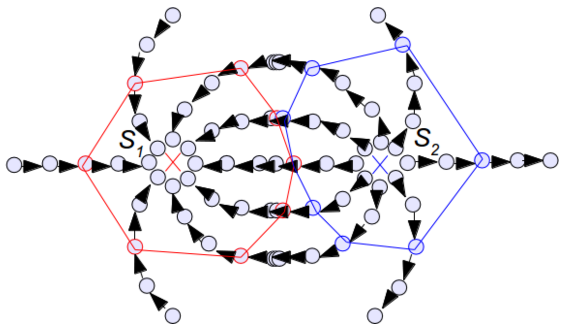
As noted, the algorithm and its few variants were all successful at making reasonable segmentations in the presence of serious noise, as compared to the threshold methods. Figure 4.1 shows the result of the threshold method on a University of Washington phantom. For the larger objects in particular, but for ultimately for all objects, the noise causes significant reduction in volume measured by threhsold methods due to high points increasing the threshold and low points leaving holes. The algorithm's framework, however, avoids this problem entirely.

Lastly, one thing to keep in mind about the algorithm is that it can be readily refined to solve errors left by the automated portion. Figure 3.24 demonstrates this for a lesion that had significant volumetric error with the standard algorithm. The error was virtually eliminated with a singular action.

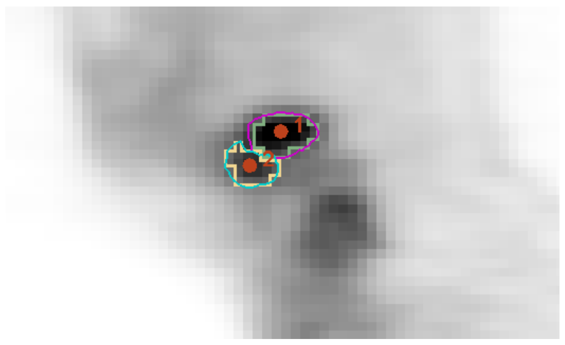
Overall, the algorithm's performance in this test shows that it is effective on some scanners and protocols and can be adapted to different others.

4.2 Future Work

While there are multiple ways of dealing with separate segmentation of nearby lesions, such as splitting, label avoidance and sealing, these tend to be noticeably affected by the order in which the objects are segmented. The ideal solution to this would be a method for mutual segmentation of objects, such that neither had an "advantage" in the segmentation. A prototype version of this was developed outside of the main work of this project. It works by making two surfaces in the graph and warping the columns to direct out from one point and into another, rather than straight out. While reasonably effective, it is not optimized or ready for use and testing for validation like the main method. Furthermore, expanding it to more than two objects will lead to much greater complexity. Figure 4.2 shows the basic graph construction of the paired segmentation, along with a simple example.



(a)



(b)

Figure 4.2: An example of two-object mutual segmentation. (a) The approximate graph structure before inter-column edges, with surfaces S_1 and S_2. Columns of the two non-spheres bend toward each other on electric field lines. (b) An example segmentation of two adjacent objects.

CHAPTER 5
CONCLUSION

FDG PET imaging is important for radiation treatment planning and quantitative analysis of lesions. There are a variety of methods for segmentation of lesions in PET volumes, but in practice, for all its issues with time and consistency, manual segmentation by an expert is far and away the most common method.

The goal was to make an algorithm for FDG PET lesion segmentation that is faster and more consistent across users and trials than manual segmentation, while still being approximately as accurate. The purpose of this goal has been to allow for a method that can effectively replace manual segmentation in practice to improve consistency for quantitative analysis.

The accuracy is, if not identical, not too far off from manual segmentation and fair compared to other standard methods, allowing this method to be used in place of manual segmentation without serious handicap. The speed is superior, allowing for reduced time spent making the segmentation. Finally, the consistency is far superior, meaning that segmentations made with this method will be more reliable for quantitative analysis in the future.

As such, this algorithm is directly useful for segmentation in order to reduce the time and variability of segmentation, without a sacrifice in effective accuracy.

REFERENCES

[1] R. Adams and L. Bischof. Seeded Region Growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):641–647, June 1994.

[2] M. Aristophanous, B. C. Penney, M. K. Martel, and C. A. Pelizzari. A Gaussian mixture model for definition of lung tumor volumes in positron emission tomography. *Medical Physics*, 34(11):4223–4235, 2007.

[3] S. Belhassen and H. Zaidi. A novel fuzzy C-means algorithm for unsupervised heterogeneous tumor quantification in PET. *Medical Physics*, 37(3):1309–1324, March 2010.

[4] J. Daisne, M. Sibomana, A. Bol, T. Doumont, M. Lonneux, and V. Gregoire. Tri-dimensional automatic segmentation of PET volumes based on measured source-to-background ratios: influence of reconstruction algorithms. *Radiotherapy & Oncology*, 69(3):247–250, 2003.

[5] L. Drever, W. Roa, A. McEwan, and D. Robinson. Comparison of three image segmentation techniques for target volume delineation in positron emission tomogrophy. *Journal of Applied Clinical Medical Physics*, 8(2):93–109, 2007.

[6] X. Geets, J. A. Lee, A. Bol, Max Lonneux, and V. Grgoire. A gradient-based method for segmenting FDG-PET images: methodology and validation. *European Journal of Nuclear Medicine and Molecular Imaging*, 34(9):1427–1438, 2007.

[7] D. Han, J. Bayouth, Q. Song, A. Taurani, M. Sonka, J. Buatti, and X. Wu. Globally Optimal Tumor Segmentation in PET-CT Images: A Graph-Based Co-Segmentation Method. In *Proceedings of the 22Nd International Conference on Information Processing in Medical Imaging*, volume 22 of *IPMI'11*, pages 245–256. Springer-Verlag, 2011.

[8] J. Kuhnigk, V. Dicken, L. Bornemann, A. Bakai, D. Wormanns, S. Krass, and H. Peitgen. Morphological Segmentation and Partial Volume Analysis for Volumetry of Solid Pulmonary Lesions in Thoracic CT Scans. *IEEE Transactions on Medical Imaging*, 25(4):417–434, April 2006.

[9] S. Lefèvre. Knowledge from Markers in Watershed Segmentation. In *Proceedings of the 12th International Conference on Computer Analysis of Images and Patterns*, CAIP'07, pages 579–586. Springer-Verlag, 2007.

[10] H. Li, W. L. Thorstad, K. J. Biehl, R. Laforest, Y. Su, K. I. Shoghi, E. D. Donnelly, D. A. Low, and W. Lu. A novel PET tumor delineation method based on adaptive region-growing and dual-front active contours. *Medical Physics*, 35(8):3711–3721, August 2008.

[11] K. Li, X. Wu, D. Z. Chen, and M. Sonka. Optimal Surface Segmentation in Volumetric ImagesA Graph-Theoretic Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):119–134, January 2006.

[12] S. Z. Li. Markov random field models in computer vision. In *ECCV '94 Proceedings of the Third European Conference-Volume II on Computer Vision - Volume II*, ECCV '94, pages 361–370. Springer-Verlag, 1994.

[13] J. Rivest, S. Beucher, and J. Delhomme. Marker-controlled segmentation: an application to electrical borehole imaging. *Journal of Electronic Imaging*, 1(2):136–142, 1992.

[14] D. A. X. Schinagl, W. V. Vogel, A. L. Hoffman, J. A. van Dalen, W. J. Oyen, and J. H. A. M. Kaanders. Comparison of five segmentation tools for 18F-fluoro-deoxy-glucose-positron emission tomogrophy-based target volume definition in head and neck cancer. *International Journal of Radiation Oncology Biology*Physics*, 69(4):1282–1289, November 2007.

[15] T. Sheperd, M. Ters, R. R. Beichel, R. Boellard, M. Bruynooghe, V. Dicken, M. J. Gooding, P. J. Julyan, J. A. Lee, S. Lefvre, M. Mix, V. Naranjo, X. Wu, H. Zaidi, Z. Zeng, and H. Minn. Comparative Study With New Accuracy Metrics for Target Volume Contouring in PET Image Guided Radiation Therapy. *IEEE Transactions on Medical Imaging*, 31(11):2006–2024, November 2012.

[16] S. Sun, M. Sonka, and R. R. Beichel. Lung Segmentation Refinement based on Optimal Surface Finding Utilizing a Hybrid Desktop/Virtual Reality User Interface. *Computerized Medical Imaging and Graphics*, 37(1):15–27, January 2013.

[17] C. Vachier and F. Meyer. The Viscous Watershed Transform. *Journal of Mathematical Imaging and Vision*, 22(2-3):251–267, May 2005.

[18] S. K. Warfield, K. H. Zou, and W. M. Wells. Simultaneous Truth and Performance Level Estimation (STAPLE): An Algorithm for the Validation of Image Segmentation. *IEEE Transactions on Medical Imaging*, 23(7):903–921, 2004.

[19] H. Zaidi and M. Abdoli. Comparative methods for PET image segmentation in pharyngolaryngeal squamous cell arcinoma. *European Journal of Nuclear Medicine and Molecular Imaging*, 39(5):881–891, May 2012.