Automating Brain Tumor Detection in MRI Images

Wesley Chen, Brandon Sim

December 13, 2012

Contents

1 Introduction and Motivation 3

2 Previous Work 3

3 Methodology 3
3.1 Watershed Segmentation 4
3.2 Symmetry Analysis 5
3.3 Expansion to Full Image Set 5

4 Results 6
4.1 Watershed Segmentation 6
4.2 Symmetry Analysis 8
4.3 Expansion to Full Image Set 8

5 Discussion 9

6 Conclusion and Further Work 10

7 References 10

8 Appendix A: Watershed Segmentation Images 11

9 Appendix B: Symmetry Analysis Images 12

10 Appendix C: MATLAB code 14
10.1 Watershed Segmentation Code 14
10.1.1 watershedMethod.m 15
10.2 Symmetry Analysis Code 19
10.2.1 filename.m 19
10.3 Anomaly Detection Code 19
10.3.1 filename.m 19
10.4 Miscellaneous Code 19

10.4.1 python.py 19

1 Introduction and Motivation

Magnetic resonance imaging, or MRI, is one of the most accurate and available types of medical imaging that is used to detect diseases throughout the body, ranging from severe bleeding to cancerous growths (3). The prevalence of MRI scans for diagnosis in the medical field is well represented by the sheer number of expensive yet critical MRI equipment - there are over 10,000 MRI machines in just the United States alone (4). When a doctor administers an MRI scan. the machine stores many 2D slices of the brain. These scans are also taken from various axials. From this collection of slices, the doctor must look through each frame for possible irregularities that could confirm the existence of a tumor. The person reading the MRI image must flip through each of the hundreds of frames systematically - there is no shortcut to knowing about where a tumor could be. Often times, the doctor must look at the same images multiple times even if her or she does not see a tumor just to make sure a small tumor (which could be as large as just spanning 3 frames and only tens of pixels) was not missed. The requirement of such a methodical but tedious search inspired us to develop approaches to automate the reading of a set of MRI images. Such a project lends itself very well to linear algebra. We are able to represent grayscale MRI images as matrices of numbers, allowing us to use matrix manipulations and linear algebraic algorithms. For the scope of this project, we limited our dataset to MRI images of the brain. We explored traditional image processing techniques as well as developed our own more specialized technique to identify possible tumor candidates from a given 2D image (one slice of the brain). We then extended this search to an entire stack of images representing the full 3D brain. Although we recognize that our algorithm will probably not fully replace a doctor's screening of MRIs, for liability and accuracy concerns, we do hope that it will both serve as a valuable backup "safety net" for the doctor, as well as be able to suggest likely tumor candidates that the doctor can focus special attention on.

2 Previous Work

Other authors of literature in this field have employed a large range of image processing and statistical methods in hopes of automating various aspects of MRI scan analysis. These approaches range from combinations of image processing filters, segmentation, dimensionality reduction algorithms, to statistical and machine learning approaches (7,8). In this paper, we experiment with a watershed segmentation technique first, a more traditional image processing technique. We then create a new approach that utilizes the symmetry of the brain, and use the two approaches to verify and strengthen our results.

3 Methodology

We were given a complete data set of a MRI brain scan for one patient with a relatively small tumor, courtesy of Dr. Steven Shufflebeam of Massachusetts General Hospital. Using a small tumor as our test case, we believe that larger tumors will only be more obvious to identify. We tried two independent approaches to attempt to identify a positive tumor case out of a given 2D frame: the first was using watershed segmentation to filter various foreground objects, hoping to identify the tumor as one of the fewer remaining objects. A brain tumor usually appears as a denser, white area which should be filtered out by the segmentation algorithm. Because this algorithm relies on 3D detail which our 2D scan did not have, we tried another approach which relies on the symmetry of the brain. In the body of this paper, we detail at a high level the methods of the two approaches we used; in our appendices, we include images detailing the results of each step.

3.1 Watershed Segmentation

The goal of the watershed segmentation method is to map the image to a topological equivalent, and "flood out" various "basins"to represent different levels of foreground. The first step would be to convert to grayscale. The intensity of the grayscale code would be mapped to relative height. Conceptually, once this grayscale mapping is complete, the whitest pixels would represent areas of maximum height (or intensity or density in the case of an MRI image). Using a gradient magnitude should result in proper segmentation. However, due to small differences in gradient, this results in an overly segmented product (see results section below).

To eliminate such detail, a set of image erosion and dilations must be conducted to eliminate some foreground detail. The foreground is selected through a morphological structuring element provided by the `strel()` function of MATLAB's image processing toolkit. We chose to use "disk" shaped elements as the round object best represents the structures we would find in the brain. We then calibrated the size of this element and this is where we controlled the level of segmentation. This element is used by the image manipulation functions below.

After this element is created, we used other Matlab image processing functions (namely `imerode()` and `imdilate()`) to help remove foreground detail and provide a map of topological height most prominent foreground areas. `Imerode()` and `imdilate()` are complimentary functions that look at every subset of the disk shaped element within the full image matrix to find areas with very similarly colored pixels - combining them into a gradual gradient. In other words, it dilates the detail of the foreground to produce more uniform areas that won't over-segment when running a watershed algorithm.

After the establishment of the foreground markers, the markers were superimposed over the original image. Another sequence of erosion-dilation was applied to help smooth out the foreground markers superimposed onto the original image. Then, a grayscale thresholding function was run to select the background areas. Anything darker than a certain threshold would be marked as "background". Then, the `watershed()` function was called to create ridges (the valleys of the basin area - also our background). This background marker image was then superimposed onto the image with the foreground marker to create the fully marked image. In our MRI case, because the background is uniformly black, this step was not necessary but was kept in our program for comprehensiveness (will still work on any image).

Finally, we visualized the result by using a colored watershed visualization which filled each object with a different color. This color map was then superimposed with a 50% transparency over our original image. Then, each color would be a different component to identify as a possible tumor candidate.

3.2 Symmetry Analysis

The second approach we used was a more unique one designed to take advantage of the highly symmetric nature of the brain. Our hypothesis was that, while many features of the brain are symmetric in the brain, tumors would most likely not be symmetric. This reasoning led us to create the following algorithm:

1. We first detect the axis of symmetry of the brain with respect to the image by finding the ellipse that has the same normalized second central moment as the region in question and calculating the orientation of its semimajor axis with respect to the x-axis of the image.
2. We detect the edges using the Canny edge detection operator, which we choose over other edge detection algorithms such as the Sobel operator because it uses hysteresis thresholding rather than fixed thresholding. Although it does require more computational time to run, it results in an algorithm that is more robust to noise and avoids multiple edge lines.
3. We detect all closed edges; that is, edges which form a curve that completely enclose some area.
4. We reflect each closed object detected in the image across the axis of symmetry, and calculate the overlapping number of pixels for each set of objects, pairwise. Those pairs of closed objects with the lowest overlapping area are the areas with highest likelihood of being tumors.
5. Finally, as an additional checking measure, we generate ellipses with the same normalized second central moment as each of the Canny-filtered closed regions and find their centroids. This step will be the basis of the second part of our symmetry analysis and will be discussed more in detail later.

3.3 Expansion to Full Image Set

Because both the watershed segmentation approach and the approach described above required us to input the exact 2D frame where the tumor was located, these approaches are not easily generalizable; indeed, it requires a human to manually find the slice which has the tumor on it to feed as input into the algorithms. To remedy this problem, we modified our symmetry algorithm to take a series of images (for example, as frames of a GIF file, or as a series of image files) and output possible regions in frames that the algorithm has detected as possible tumor candidates. Below, we describe the modified algorithm:

1. For each image in the series of images, we use the ellipse-fitting methods described above in order to find the centroids of all enclosed regions in the image after applying the Canny edge detection operator.
2. We reflect each of these centroids, pairwise, across the calculated axis of symmetry, and assign each pair of centroids a score based on the Euclidean distance, in pixels, between them.

3. For each centroid, we take its lowest distance out of all its pairwise interactions with the other centroids; we then find the centroid with the highest of these lowest distances and mark that as the best candidate in each frame. This is because we want the centroid whose lowest distance away from any of the other centroids (or reflected centroids) is the highest, as this is an indicator of a higher probability of being a tumor (or at least, a non-structural object).
4. For each set of adjacent frames in the series of images, we calculate the Euclidean distance between the best candidates of those frames, and output all frames that contain best candidates whose distances in adjacent frames is less than 1 pixel, some arbitrary threshold which can be set by the user (the higher the threshold, the more sensitive and the more false positives).

4 Results

For both cases, we started with the 2D MRI snapshot below that we selected from the stack as it clearly shows the tumor - the white area located in the bottom left quandrant.

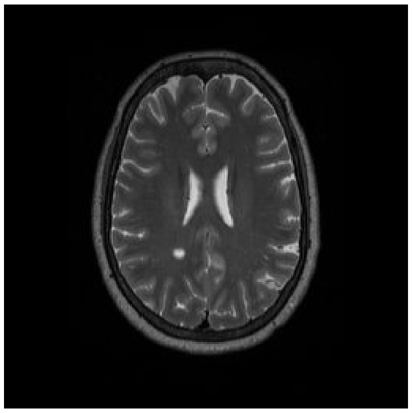


Figure 1: Original 2D Tumor Image

4.1 Watershed Segmentation

Because the watershed segmentation algorithm is designed for more complex (3D), we will also show the output for a given sample colored image, a lily pad landscape, in addition to the MRI image. The end results of the segmentation algorithm are shown below.

The following is the watershed segmentation of the lilypad image.

For results of each individual step, please refer to Appendix A.

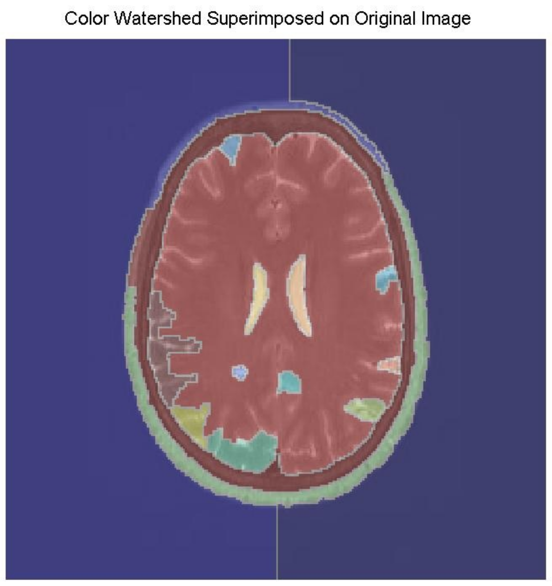


Figure 2: Colored Watershed Segmentation of Processed MRI image. Notice that the tumor region is successfully identified as a unique area. however, due to the 2D subject of an MRI image, the algorithm overcomplicates the foreground and identifies various other regions as false positives as well (see Discussion)

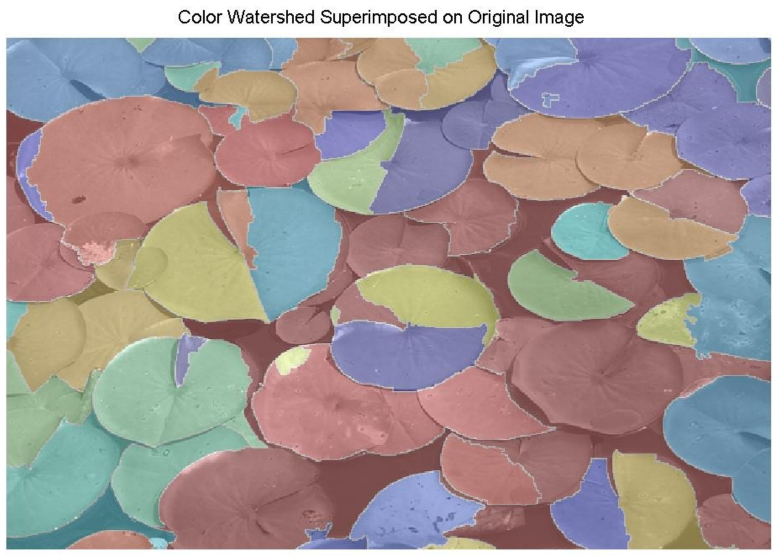


Figure 3: Colored Watershed Segmentation of Processed Lilypad Landscape. With this 3D example, one can see that the image algorithm successfully reduces the 3D subject of the image to unique regions and represents them with different colors

4.2 Symmetry Analysis

The following image is the output of our reflections. It is clear that there is an area overlap for the nontumor regions and no area overlap for the tumor.

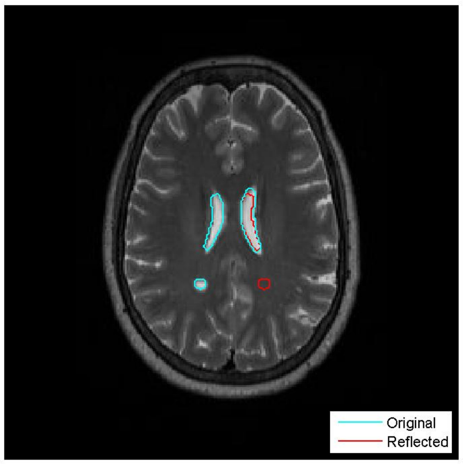


Figure 4: Result after reflection, single slice.

4.3 Expansion to Full Image Set

Our algorithm as described above pointed out frames 79, 80, and 81 as frames with tumor candidates. These frames are included below:

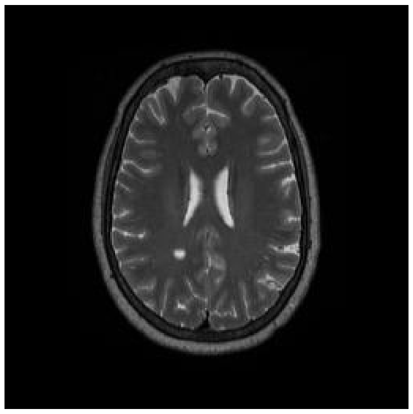


Figure 5: Frame 79

As can be seen, the tumor is indeed on these frames!

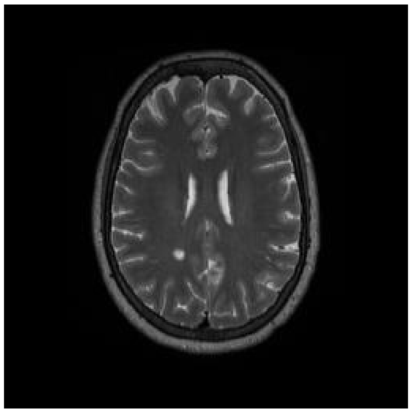


Figure 6: Frame 80

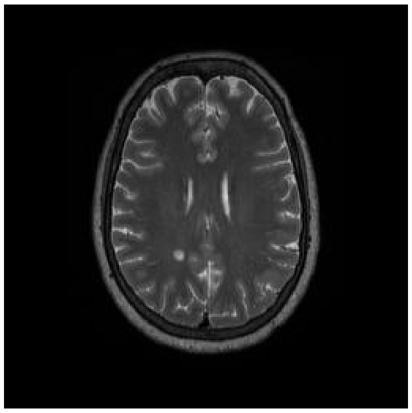


Figure 7: Frame 81

5 Discussion

The results from the image processing methods show much cleaner results for the symmetry analysis approach. This is most likely because the image processing algorithm assumes 3D level of detail for the image (selecting a foregournd and a background). When the 2D MRI image does not represent a 3D picture, there is a loss of data that helps the image filters work. thus we see some random articfacts in the watershed approach.

The symmetry analysis, as a much more simple algorithm, gave us remarkably better results. We recognize, however, that this algorithm is limited by the symetry. Therefore, we cannot extend this algorithm to process other parts of the body that are assymetrical, unlike that of the image processing algorithm.

We are also aware that individually, both algorithms need to be semi-calibrated to give us the best output. When run using auto-thresholding, the watershed segmentation method has a tendency to provide false positives due to the fact that the tumor is not the absolute brightest structure recorded by the MRI. It would be difficult to teach our algorithm to ignore such areas because sometimes, the tumor could in fact be the brightest area. However, this problem can be reduced by running the algorithm at various thresholds for the morphological structure element used to simplify the image. Regardless, providing false positives is more acceptable than a false negative, as a doctor would be able to quickly filter through the highlighted possibilities for tumor candidates.

The edge detection algorithm requires a similar calibration step - setting the hystersis threshold for the Gaussian filter. The same process to test various thresholds for repeated results would help reduce these errors. Because of the simplicity of this design, this algorithm seems to be favored with our current results. One weakness of the symmetry based edge detection is that there may be false negatives (if the tumor is bordering another edge), an error that is less desirable in medical applications.

Because the algorithms are independent in their approaches, successive application of the two would only improve our confidence. Through such cross-checking, the agreed parts would have the highest probability of being the area of interest. Combining both methods would increase our detection rate while minimizing false positives.

6 Conclusion and Further Work

Our project has demonstrated two possible independent directions to the automation of MRI reading, a watershed segmentation approach and a symmetry-based approach. We were able to successfully detect the tumor in our training data (one slice) as well as our test data (entire stack), using the symmetry approach. In the future, we hope to integrate our two approaches, as well as apply our algorithms to more data sets (i.e., more patients). Because our code is extremely modular and has limited "magic numbers", we do not anticipate any complications when expanding to a new data set. We would also like to further cross-check and validate our results using the outputs of both of the algorithms in order to decrease false positive rates. In addition, we would like to explore better dimensionality reduction algorithms to better represent the features in our data.

We hope that our work provides an interesting and unique approach towards image analysis of brain scans through MRI. We invite you to view all of our code, input data, output data, and images at our Github repository at the following URL: `https://github.com/bksim/images`.

7 References

1. MATLAB - image processing toolbox
2. http://www.mathworks.com/products/image/examples.html?file=/products/demos/shipping/images/i
3. http://www.medicinenet.com/mri_scan/article.htm
4. http://www.magnetic-resonance.org/MagRes%20Chapters/21_03.htm
5. http://12000.org/my_courses/UCI_COURSES/CREDIT_COURSES/fall_2004/EECS_207A_UCI_FALL_
6. http://www.csse.uwa.edu.au/ pk/research/matlabfns/

7. Salman SD, Bahrani AA. Segmentation of tumor tissue in gray medical images using watershed transformation method.

8. Roy S, Bandyopadhyay SK. Detection and Quantification of Brain Tumor from MRI of Brain and its Symmetric Analysis.

8 Appendix A: Watershed Segmentation Images

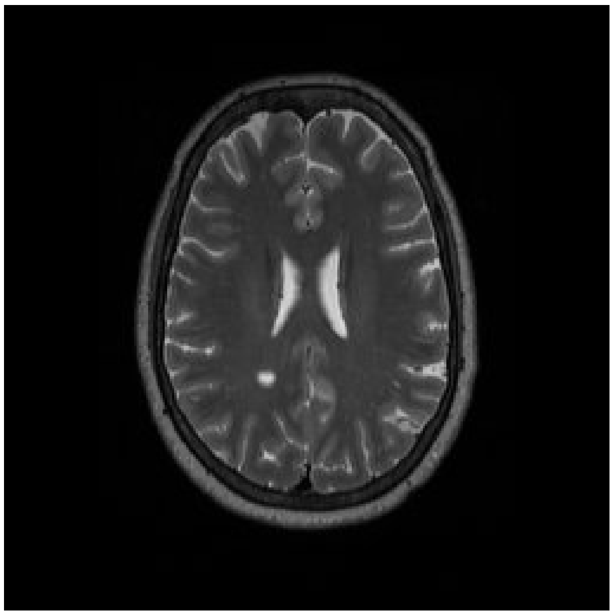
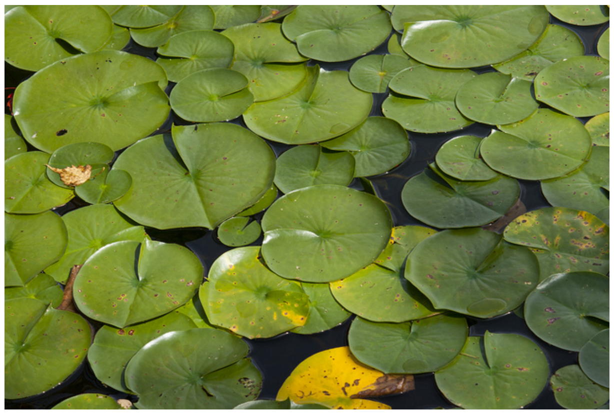
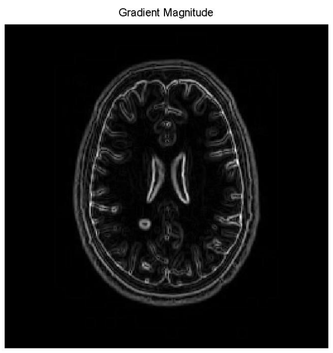



Figure 8: Recall the original MRI image as well as a 3D-subject lilypad landscape for comparison



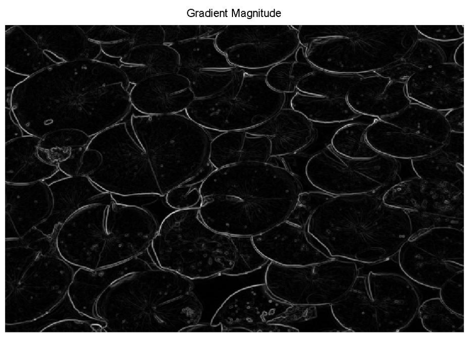
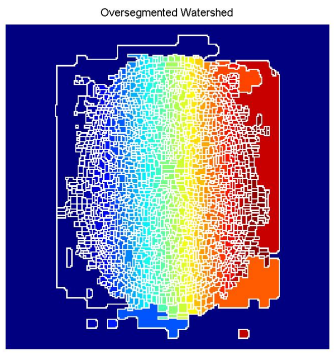


Figure 9: A logical first stepwould be to calculate gradients directly from the grayscale



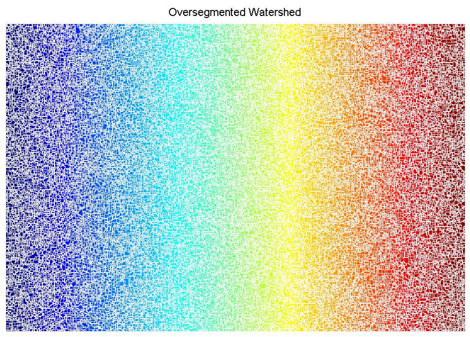


Figure 10: Apply the watershed transform to the gradient results in oversgementation where there is too much foreground detail

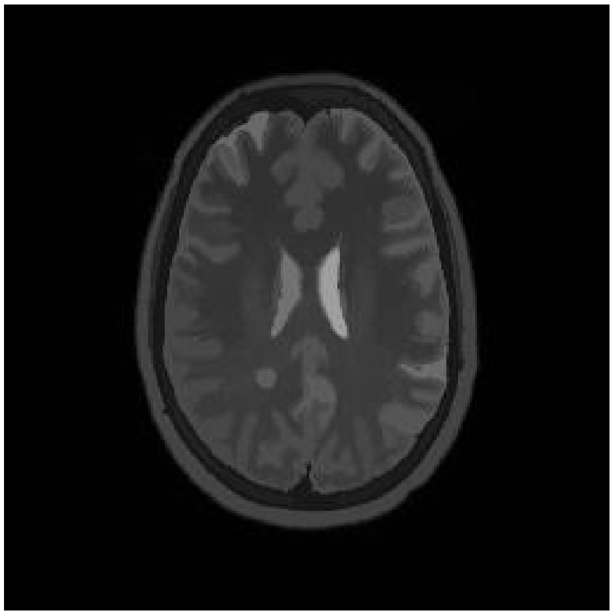


Figure 11: Apply image erosion and dilation functions to remove foreground detail. Notice the disappearnce of texture and blemishes best seen from the lilypads

9 Appendix B: Symmetry Analysis Images

Please see github.

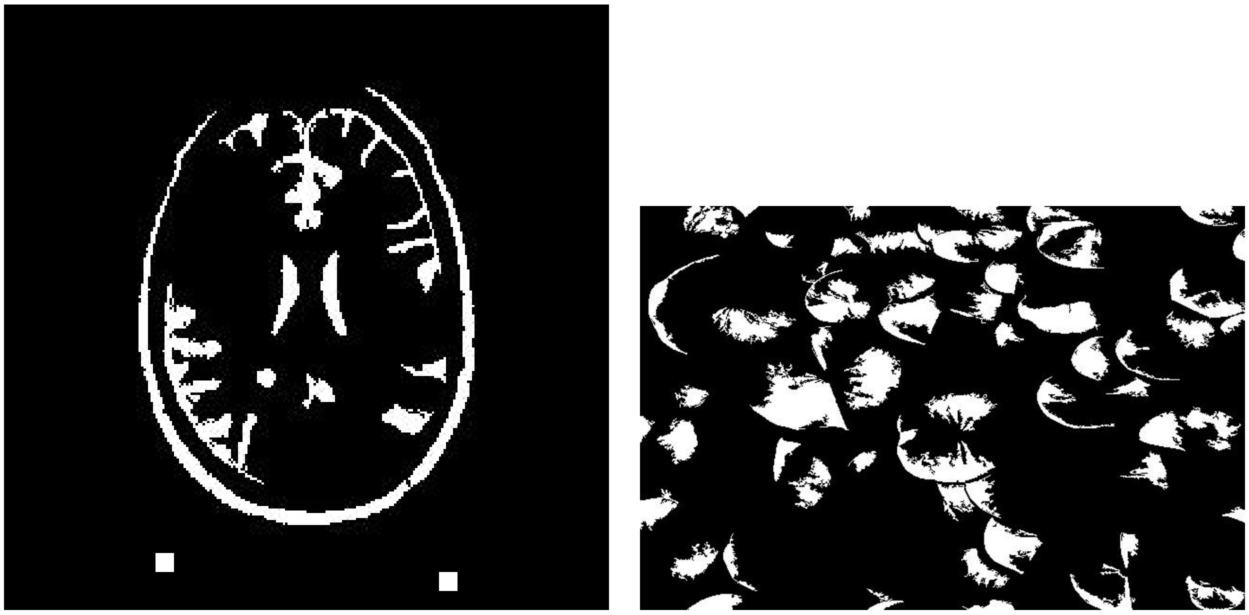


Figure 12: Calculate maxima which marks the foreground area

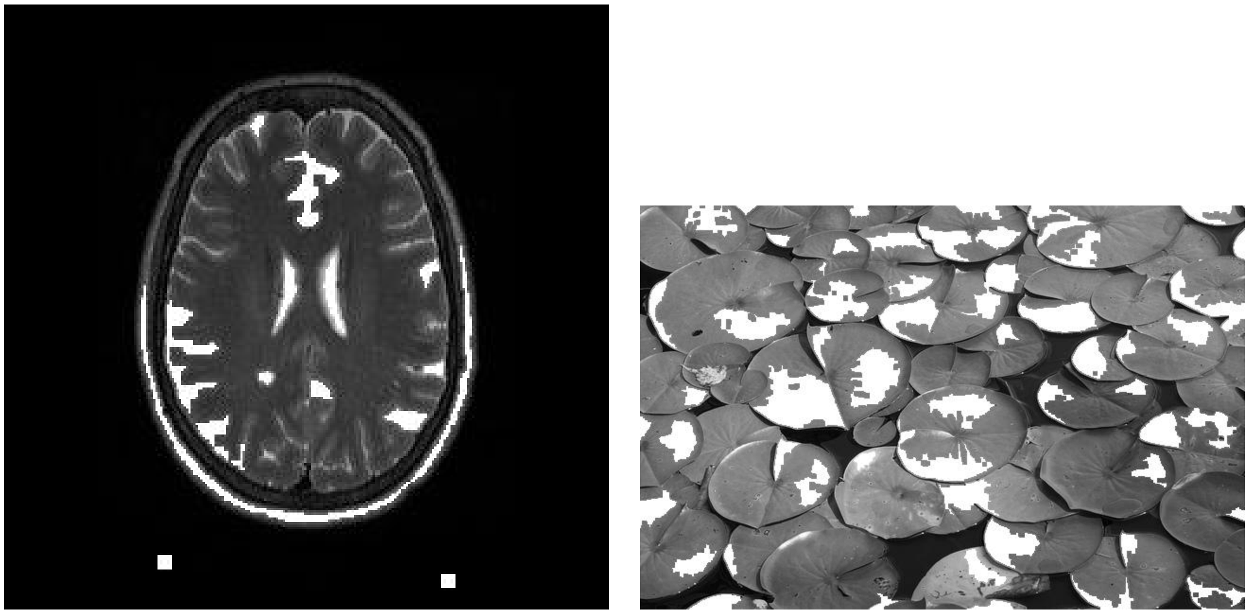


Figure 13: Superimpose maxima with original image and run erosion and dilation function pair to smooth out edges

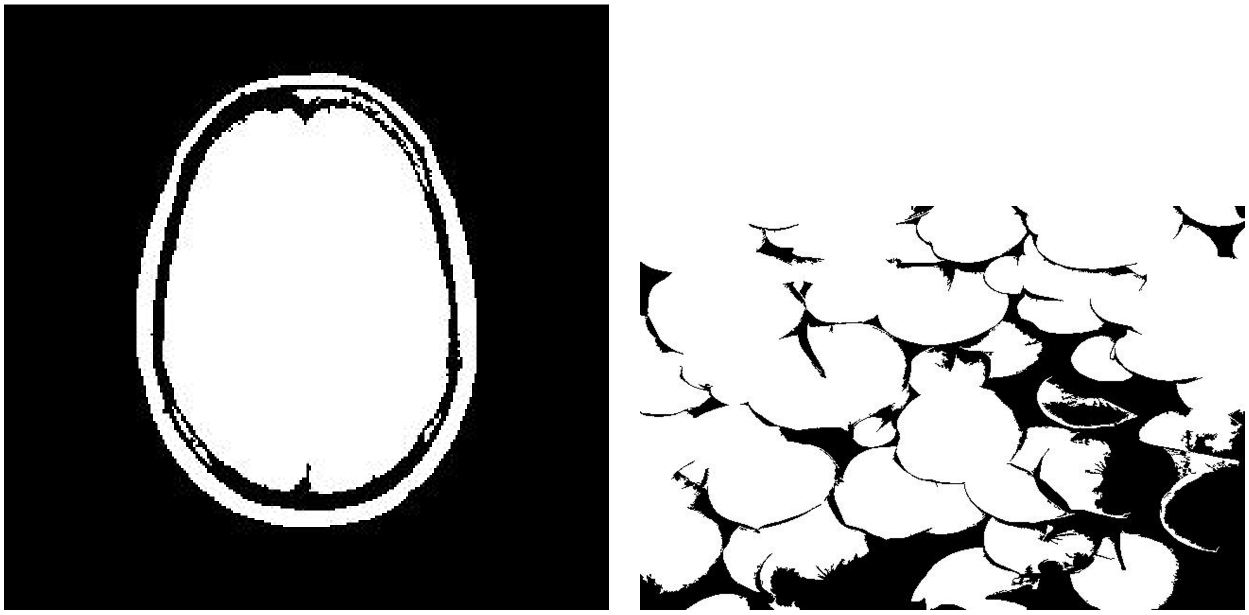


Figure 14: Detect background with simple thresholding

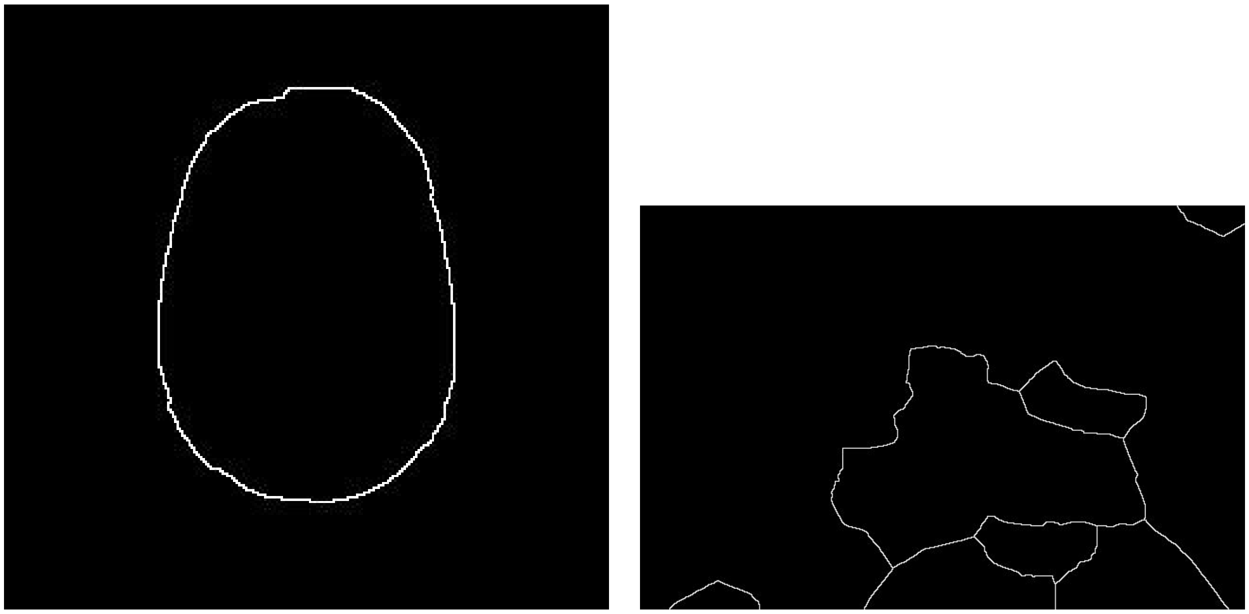


Figure 15: Mark watershed ridgelines, the background

10 Appendix C: MATLAB code

10.1 Watershed Segmentation Code

Treats the input image across various filters to display the final watershed segmented color overlay.

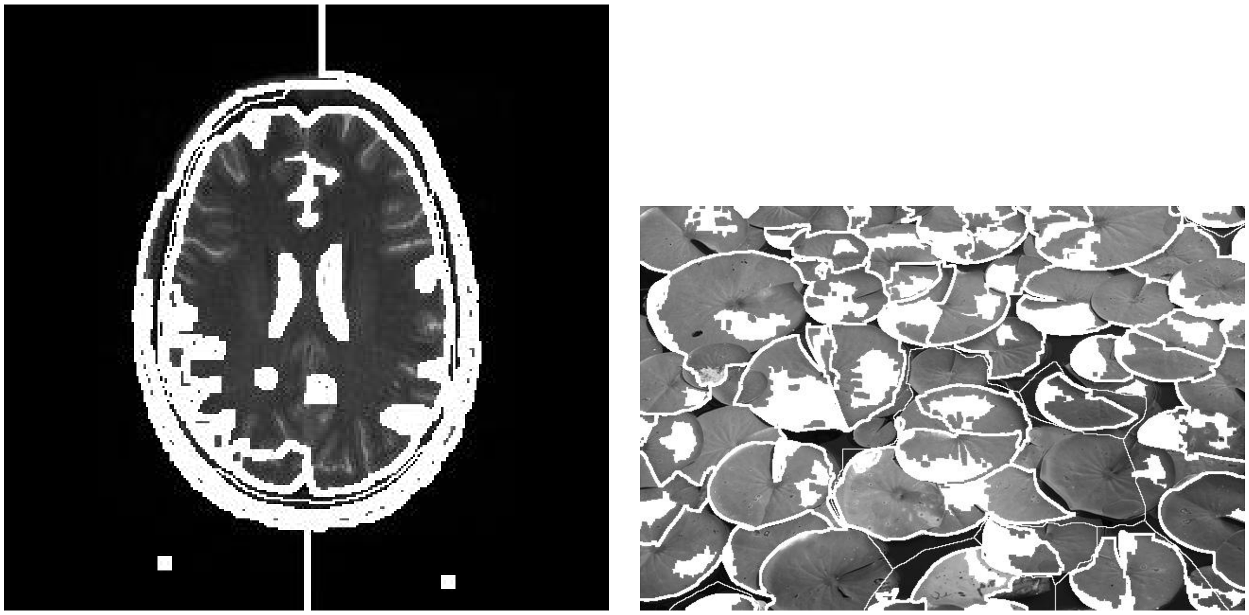


Figure 16: Superimpose all foreground and background markers

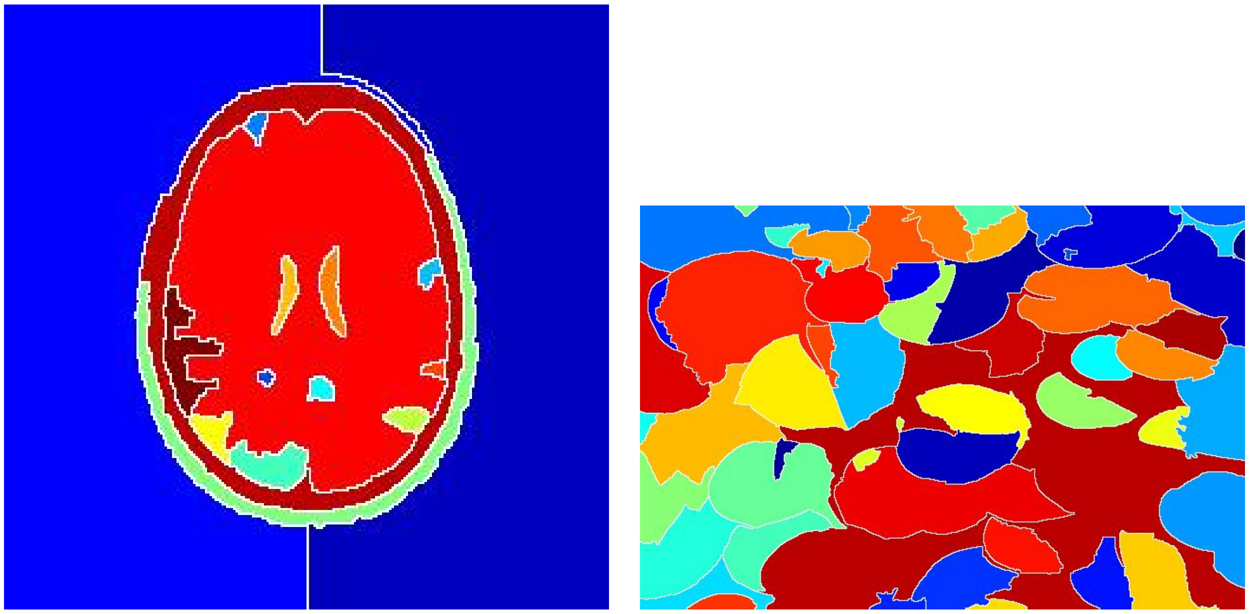
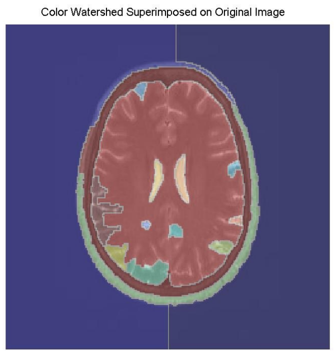


Figure 17: Get watershed transform. This result is no longer oversegmented

10.1.1 watershedMethod.m

```
%generalized watershed method starting with colored image

%read image - using lilypad.jpg, a colored, 3D landscape of lilypads
I = imread('lilypad.jpg');
```



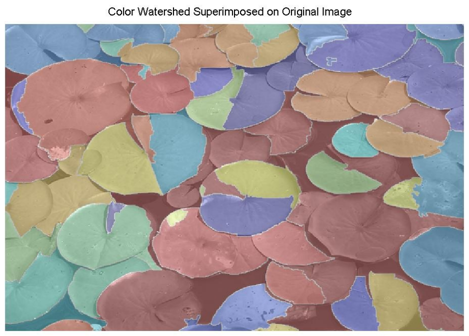


Figure 18: Superimpose colored watershed result with 50% transparency over original image

```
%convert RGB into grayscale
I = rgb2gray(I);

%horizontal edge filter
hy = fspecial('sobel');
%vertical component
hx = hy';

%use image filter, needs "double" input, filter, option
Iy = imfilter(double(I), hy, 'replicate');
Ix = imfilter(double(I), hx, 'replicate');

%calculate gradient
gradmag = sqrt(Ix.^2 + Iy.^2);
figure(1), imshow(gradmag, []), title ('Gradient Magnitude')

%export this figure, need to scale range so coudln't just imwrite
print (1, '-djpeg', 'Gradient_Mag')

%demonstrate oversegmentation of direct watershed of gradient
L = watershed(gradmag);
Lrgb = label2rgb(L);
figure(2), imshow(Lrgb, []), title('Oversegmented Watershed')

%export this figure (couldn't export image due to transparency)
print (2, '-djpeg', 'Overseg_Watershed')

%create disk shaped morphological structuring element of size 6 - picked so
%not to over or under segment, disk shaped as target structure is round
se = strel('disk', 6);
```

```
%first method to open: (imopen)
Io = imopen(I, se);
imwrite(Io, 'Open_Disk6.jpg', 'jpg');
%figure, imshow(Io), title('Opening (Io)')

%second method to open: reconstructed (imerode)
Ie = imerode(I, se);
Iobr = imreconstruct(Ie, I);
imwrite(Iobr, 'OpenReconstuct_Disk6.jpg', 'jpg');
%figure, imshow(Iobr), title('Opening-by-reconstruction (Iobr)')

%first method of closing: (imclose)
Ioc = imclose(Io, se);
imwrite(Ioc, 'Close_Disk6.jpg', 'jpg');
%figure, imshow(Ioc), title('Opening-closing (Ioc)')

%second method to close: reconstructed (imdilate)
Iobrd = imdilate(Iobr, se);
Iobrcbr = imreconstruct(imcomplement(Iobrd), imcomplement(Iobr));
Iobrcbr = imcomplement(Iobrcbr);
imwrite(Iobrcbr, 'CloseReconstruct_Disk6.jpg', 'jpg');
%figure, imshow(Iobrcbr), title('Opening-closing by reconstruction
%(Iobrcbr)')

%use function to store maxima (foreground marker)
fgm = imregionalmax(Iobrcbr);
imwrite(fgm, 'Maxima.jpg', 'jpg');
%figure, imshow(fgm), title('Regional maxima of opening-closing by
%reconstruction (fgm)')

%copy image and superimpose to form new working image
I2 = I;
I2(fgm) = 255;
imwrite(I2, 'MaximaSuperimposed.jpg', 'jpg');
%figure, imshow(I2), title('Regional maxima superimposed on original image
%(I2)')

%create a new morphological element to smooth out the foreground markers
se2 = strel(ones(3,3));

%smooth out the foreground markers superimposed
fgm2 = imclose(fgm, se2);
fgm3 = imerode(fgm2, se2);
```

```
fgm4 = bwareaopen(fgm3, 15);
I3 = I;
I3(fgm4) = 255;
imwrite(I3, 'MaxmimaSuperSmooth.jpg', 'jpg');
%figure, imshow(I3), title('Modified regional maxima superimposed on
%original image (fgm4)')

%identify background pixels - anything below a certain threshold is
%"background"
bw = im2bw(Iobrcbr, graythresh(Iobrcbr));
imwrite(bw, 'Background_Threshold.jpg', 'jpg');
%figure, imshow(bw), title('Thresholded opening-closing by reconstruction
%(bw)')

%adapt background to form watershed ridge lines (where there is little
%foreground, mark off)
D = bwdist(bw);
DL = watershed(D);
bgm = DL == 0;
imwrite(bgm, 'Watershed_Ridge_Lines.jpg', 'jpg');
%figure, imshow(bgm), title('Watershed ridge lines (bgm)')

%calculate watershed segmentation as before
gradmag2 = imimposemin(gradmag, bgm | fgm4);
L = watershed(gradmag2);
I4 = I;
I4(imdilate(L == 0, ones(3, 3)) | bgm | fgm4) = 255;
imwrite(I4, 'All_Markers_Super.jpg', 'jpg');
%figure, imshow(I4), title('Markers and object boundaries superimposed on
%original image (I4)')

%color image visualization
Lrgb = label2rgb(L, 'jet', 'w', 'shuffle');
imwrite(Lrgb, 'Color_Watershed.jpg', 'jpg');
%figure, imshow(Lrgb), title('Colored watershed label matrix (Lrgb)')

%superimpose color on original image to see what the shading represents
figure(3), imshow(Lrgb)
title('Color Watershed Superimposed on Original Image')
hold on;
handle = imshow(I);
alpha(0.5);
hold off;
```

```
%export this figure (couldn't export image due to transparency)
print (3, '-djpeg', 'result')
```

10.2 Symmetry Analysis Code

10.2.1 filename.m

```
Please see Github.
```

10.3 Anomaly Detection Code

10.3.1 filename.m

```
Please see Github.
```

10.4 Miscellaneous Code

Converts stack of MRI images stored as a GIF into a sequence of JPEG files

10.4.1 python.py

```
Please see Github.
```