Detection of Alzheimer's Disease Plaques in a Transgenic Mouse Brain Using Image Analysis of SPION-Enhanced Magnetic Resonance Images.

New Mexico Supercomputing Challenge
Final Report

April 4, 2012

Team 82
Manzano High School

Team Member: Jordan E. Medlock

Mentor: Laurel Sillerud, PhD, University of New Mexico, Department of Biochemistry and Molecular Biology

Correspondence: jordanemedlock@gmail.com

Running title: Image Analysis of Alzheimer's Plaques in MRIs of a Tg Mouse

TABLE OF CONTENTS

1. ABSTRACT

Alzheimer's disease is a debilitating and fatal brain disorder, which impacts millions of older adults. Currently, it is only definitively diagnosed using histological analysis of plaques and neurofibrillary tangles in the brain, post mortem. Magnetic resonance imaging techniques can be used to identify Alzheimer's plaques, but the identifiable plaques are very small and difficult to see in the image. This makes diagnosing, monitoring, and treating the disease very difficult. Contrast agents are being developed to increase the conspicuity of the plaques in magnetic resonance images, although this still means days of counting the plaques by hand. This computerized algorithm automates the process of counting and analyzing the plaques in magnetic resonance images. Alzheimer's plaques can be counted in approximately five seconds with this program, rather than a month of counting plaques manually.

2. INTRODUCTION

2.1 Alzheimer's Disease

Alzheimer's Disease (AD) is a fatal, degenerative brain disease that causes a reduction in both cognitive functioning and memory. People with AD also experience progressive changes to their personality, behavior and judgment. AD primarily impacts adults over the age of 65. According to The Alzheimer's Association (2011), AD is the most common form of dementia, with a prevalence of approximately 5.4 million people with the diagnosis in the United States as of 2011. AD not only affects the people who suffer from the disease but millions of family members are impacted by the necessity to provide care for their loved ones. On average, one in eight Americans over age 65 has been diagnosed with AD. As the U.S. population continues to age, due to improvements in medical care and environmental conditions, The Alzheimer's Association projects that by 2050 there could be 11 to 16 million people in the U.S. with AD.

Alzheimer's Disease is characterized by a loss of neurons in the cerebral cortex. Evidence of the increased formation of plaques and neurofibrillary tangles in the brains of patients with AD has been found under microscopy. These plaques and neurofibrillary tangles are generally found in the hippocampus, the entorhinal cortex, the basal forebrain, and the amygdala (Bouras et al. 1994). The AD plaques are deposits of regularly ordered fibrillar aggregates composed of amyloid-beta peptides (Aβ) of 36-43 amino acids located in the brain around neurons. Aβ is a peptide from an amyloid precursor protein (APP), a critical element for neuronal growth and post-injury repair (Turner et al. 2003). Neurofibrillary tangles are composed of tau proteins in a hyper-phosphorylated state (Shin et al. 1991). AD is a protein mis-folding disease, produced by abnormally folded Aβ peptides and tau proteins; however, the cause of this process is unknown (Hashimoto et al. 2003). The following figures show a representation of Aβ plaques near neurons and neurofibrillary tangles within the nucleus of the neuron, followed by a histological image of the brain of a transgenic Alzheimer's mouse with the presence of visible plaques:

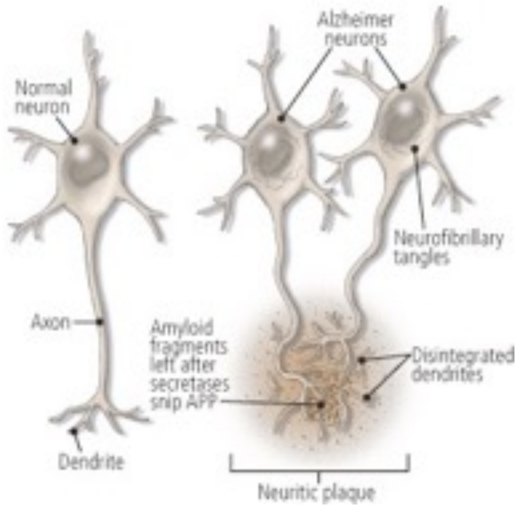

Figure 1. Representative image of Aβ plaques, from www.alz.org.

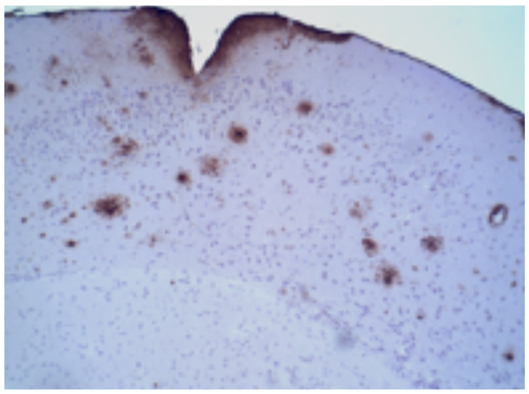

Figure 2. Histological image of Aβ plaques, from Sillerud et al.

Neuroinflammation is also present in the brains of patients with AD. Inflammation is found in many disease processes throughout the body and may be an indicator of tissue damage or an immunological response. Aβ plaques are encased in microglial cells; this is an indication that inflammation is present (Kreutzberg, 1995). The microglial cells are a type of macrophage found in brains; they are responsible for engulfing and digesting pathogens. The process by which these plaques and tangles cause atrophy and degeneration of the cerebral cortex may be related to neuroinflammation and microglial cells (Aloisi, 2001).

AD has been found to have a genetic cause in some individuals. Several genes have been found that produce proteins that enable AD development in humans. A gene has been located within chromosome 21 that is found to produce APP in different populations. A gene in chromosome 14 is transcoded into presenilin 1 (PS-1), and a gene in chromosome 1 is transcoded into presenilin 2 (PS-2). Mice were created that expressed APP, PS-1, PS-2, or tau protein genes. Most transgenic mice used in research are APP/PS-1 bigenic mice because they develop Aβ plaques and show symptoms similar to AD, such as progressive memory loss. However, these mice do not develop neurofibrillary tangles, which may impact the predictive ability of the research (Ashe, 2001). Transgenic mice that are created with tau protein genes exhibit neurofibrillary tangles but do not show signs of dementia.

2.2 Diagnosing Alzheimer's Disease

Currently, AD is diagnosed through clinical findings of neuropsycholgical symptoms, such as language and memory loss, using assessments of intellectual functioning. Other cerebral pathologies and possible causes for the dementia must be excluded to make a differential diagnosis of AD (Mendez, 2006). Neuroimaging, such as computed tomography, magnetic resonance imaging, single photon emission computed tomography, and positron emission tomography are used to exclude other causes for memory loss. In order to confirm the diagnosis, though, post-mortem brain tissue is analyzed histologically for signs of Aβ plaques. Neuroimaging techniques are not readily available to directly diagnose AD or to monitor the disease's progression, *in vivo*.

2.3 Magnetic Resonance Imaging

Magnetic Resonance Imaging (MRI) is a radiological technique used to visualize internal structures of the body. This technology employs a magnetic field to magnetize and align the magnetic poles of the nuclei within some cells in the direction of the MRI field. Tissues within the body return to their non-magnetized state at different rates, called their relaxivity rate, which allows the MRI to translate the information into images that reflect identifiable tissues such as bone, muscle, and structures within the brain (Squire, 1997). Most often, MRIs use 1-3 Teslas (T), which is a unit of measuring magnet strength. T_1-weighted, T_2-weighted, and T_2*-weighted MRIs are standards in use for clinical and research purposes. T_1-weighted scans provide images with darker water and brighter fat, which provides good gray matter and white matter contrast for images of the brain. T_2-weighted scans show dark fat and bright water, which is well suited to show inflammation. T_2* scans increase the contrast for certain types of tissues (Squire, 1997).

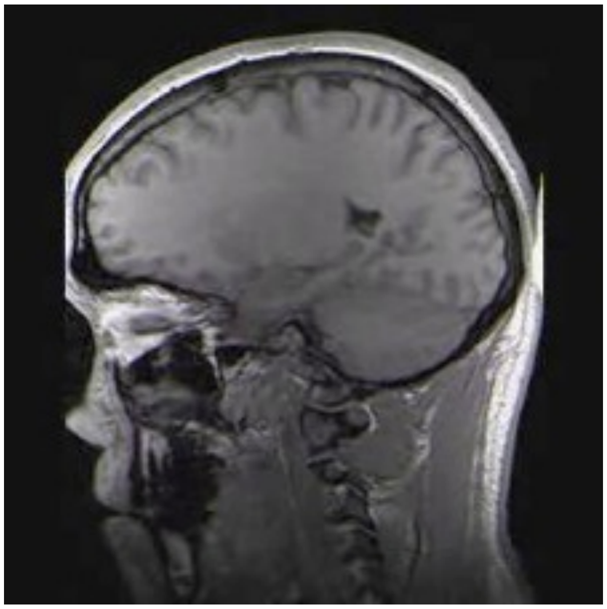

Figure 3. Para-sagittal MRI of the head, from Wikipedia Commons.

MRI has many practical uses in medicine, and it is frequently used to observe the structure of brains without harm to the patient. An MRI AD detection method would be greatly applicable for *in vivo* measurement of the progression of the disease. It has been possible to see the plaques in the brain *in vivo* with MRI;

however, they are so small, around 50 micrometers (μm), that many hours, very high resolution imaging (<100 μm), and high magnetic fields (>9T) are required to identify Aβ plaques (Dhenain et al., 2009).

2.4 SPIONs

Superparamagnetic iron oxide nanoparticles (SPIONs) have been studied as a means of increasing the magnetic susceptibility of the plaques in the brain in order to view them in lower-powered MRI fields (Sigurdsson, 2008). SPIONs have an iron oxide core that is coated with a variety of other organic or inorganic materials. The superparamagnetic quality of SPIONs is characterized by a high relaxivity time, the time it takes to reach zero magnetization when not exposed to a magnetic field. This is characterized by a relaxation time τ:

where τ is time (10^{-9}), K is the anisotropy energy (20,000J/m^3 for iron oxide), V is the volume of the particle, κ_B is the Boltzmann constant, and T is the temperature (Hofmann-Amtenbrink et al., 2009). The high relaxivity time of SPIONs compared to the brain's relatively low relaxivity time enables the MRI to readily visualize the differences in structures. When a SPION is coated with organic material, such as a peptide or protein, the median diameter of the nanoparticles is 50-160 nanometers (Hofmann-Amtenbrink et al., 2009). Anti-APP conjugated SPIONs and anti-Tau conjugated SPIONs selectively target Aβ plaques and bind to them in order to increase their conspicuity in MR images. These SPIONs have been shown to specifically recognize Aβ plaque in brains, which is consistent with histological studies (Sillerud et al.).

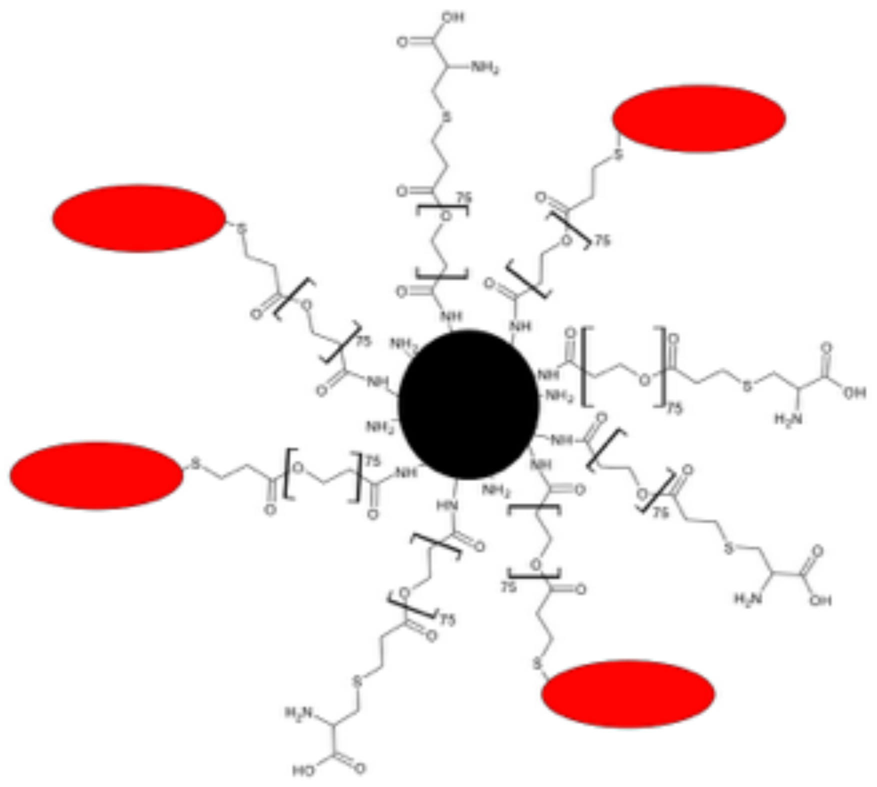

Figure 4. Structure of SPION particle coated with peptides, from Hofmann-Amtenbrink et al. 2009

Gadolinium contrast agents have been used successfully, *ex vivo*, to identify Aβ plaques; however, they leak toxic Gd^{3+} ions, which harm patients and cannot be used *in vivo* (Podulso et al., 2004). Highly invasive methods, such as injections of ozone, must also be introduced to allow the Gadolinium to cross the blood-brain barrier (BBB). Because neuroinflamation in the brains of patients with AD compromises the BBB, a contrast agent must be created that is small enough to cross the BBB without using these invasive techniques. SPIONs are non-toxic (Brambilla et al., 2011); the ferric iron in the SPION is a normally occurring element in the body and is not harmful when injected, and it is small enough to cross the BBB. This allows the SPIONs to be used to count plaques *in vivo,* in MR images.

SPION-treatment has allowed AD researchers to count plaques *in vivo*; however, counting the plaques is a long and labor-intensive process. In Sillerud et al., the researchers took days to count plaques in an MR image of a single mouse brain. It was necessary for the researchers to find the difference between the intensities of the plaque and the brain. They computed the difference in standard deviations between the intensity of the suspected plaque and its surrounding pixels. If the standard deviation was less than 2.5, the region could not be considered plaque. The process of determining whether each darker region of the brain could be considered took many hours for each plaque. A process to automate the analysis of the image would allow both researchers and physicians to more quickly gather necessary information. Researchers could gather the data for studies related to the treatment of AD faster; improving the speed treatments are available to the people who need them most. Physicians treating people with AD could use an automated process to diagnose AD earlier and help monitor the process of the disease in individuals.

2.5 Image Analysis

Computerized algorithms are created to extract meaningful data from an image. Automating the image analysis can produce quantifiable and replicable data and reduce the time needed when compared to manual analysis. Image analysis algorithms utilize different methods including machine learning, digital geometry, and signal processing. Previous studies have used machine learning, a subcategory of artificial intelligence, to analyze MR images in order to locate plaque-like areas within the body (Nattkemper, et al. 2005). Artificial intelligence and machine learning require many data sets of recognizable points to extrapolate data from an image. In machine learning, plaques are first marked and enumerated in an image where the locations of the plaques have already been established manually. Then, the program creates parameters through trial and error until all of the regions specified by the algorithm match with the established plaques. However, when analyzing an image with less than approximately 10,000 reference points the correlation between manual and automated results is low. Artificial intelligence requires large data sets to learn to unambiguously identify plaques (Bishop, 2006).

When analyzing a single brain with 668 plaques, machine learning will not provide accurate information.

Signal processing uses arithmetic operations on discrete signals to create a desired output: the identification of plaques in the brain (Lim, 1995). The entire MR image is considered the signal, which is measured by the intensity of the pixels. Specifically, the plaques need to be differentiated from the brain and the noise that is overlaid on the image during the MRI process. Signal processing is a clear, mathematical system that provides exact data from complicated images. However, when using a signal processing image analysis algorithm with MR images complications arise. The intensities of the pixels are uneven throughout the image, which makes it difficult to use the same plaque-finding parameters. Because of the magnetic field used in the MRI, the images vary from top to bottom and from center to edges. With signal processing, though, the image can be flattened and the plaques can be thresholded and identified.

3. PROBLEM STATEMENT

Problems still exist with the visualization of Aβ plaques in humans, *in vivo,* to allow for the diagnosis and treatment of AD in the early stages. A variety of neuroimaging techniques have been used to try to visualize the plaques. In MR images, the Aβ plaques cannot be readily seen without a contrast agent. When a contrast agent is introduced (SPIONs) the plaques are numerous and time consuming to count by hand. It can take up to a month for a researcher to count the plaques in a single transgenic mouse brain. If a quick computational solution can be found to count and analyze plaques *in vivo,* in MR images, it would save many hours of time, which instead could be spent monitoring and treating AD. With a computational solution, many brains can be analyzed in a single day. My hypothesis is that the number of Aβ plaques found with a signal processing computer algorithm will correlate with the data found by manually counting the plaques in SPION-enhanced MR images of transgenic mouse brains.

4. METHODS

4.1 Subjects

All animal procedures were approved by the University of New Mexico (UNM) Institutional Animal Care and Use Committee. The animal procedures were completed by the UNM Biochemistry and Molecular Biology Department. The transgenic mouse used in this research was received from Jackson Laboratory in Bay Harbor, Maine. The mouse was a six-week-old, double transgenic Alzheimer's Disease mouse (B6C3-Tg (APPswe,PSEN1dE9) 85Dbo/Mmjax). There were two transgenes in the mouse: one was the mouse/human chimeric Aβ (A4) precursor protein (APPswe; K595N/M596L), and the other was a deletion of exon 9, which corresponds with early onset Alzheimer's Disease. After 20 weeks of age, Aβ peptide and human presenilin were detected in the mouse. By 12 months, astrocytosis was measured in the mouse, with significant cognitive impairment by 13 months. For 14 months, the mouse was given *ad libitum* access to food and water. Then, the mouse was treated with 7.07 μg of anti-tau SPION and 1.52 μg of anti-APP SPIONs by vein injection in the tail and killed 24 hours later. The brain was quickly harvested and fixed in buffered formalin for three days. The brain was stored in 2% agarose gel and 3mM NAN_3, then held at 4°C until it was used for MRI analysis.

4.2 Magnetic Resonance Images

The MRI studies were conducted at the UNM Biomedical Research and Integrative Neuroimaging Center at 4.7 T. Dr. Laurel Sillerud provided the MR images to this team. The optimal sequence for plaque detection was a T_2-weighted image with slight T_2^*-weighting. A 192 x 1024 pixel (px) receive-only surface coil was used, and each pixel had a 60μm width. The MRI produced 32 coronal image-slices, each 120 μm wide.

4.3 Image Analysis

4.3.1 Programing Language

Many factors were considered in determining the programming language to use in processing the MR images for this study. MATLAB has a strong focus on matrix manipulations and a large set of image processing tools. Because the MR images are stored as a matrix of integer values, MATLAB has the ability to modify those images using its matrix manipulation tools. Established image-processing algorithms are available in MATLAB to locate object boundaries in binary images and extrapolate data from these boundaries, which would provide information regarding the edge and location of the Aβ plaques in the images. MATLAB, in contrast to C, C++, or Java, has boundary finding, matrix manipulation, and parallel processing tools without the need for a third party application-programming interface or the need to reinvent the processing tools by writing the entire code. In a field of multiple programming languages, MATLAB was determined to be the most applicable to this study.

4.3.2 Pre-Processing the Image

Thirty-two MR images were generated, each containing one MRI slice (1024x192 px), in the Digital Imaging and Communications in Medicine (DICOM) standard format. DICOM is a prominent means of formatting images in the scientific and medical communities. ImageJ, an image processing application, was used to convert the DICOM images into uncompressed Tagged Image File Format (TIFF), a recognizable MATLAB format, without any loss of resolution. After converting the images to TIFF, each file included 32 sub-files of MRI slices. MATLAB read each slice individually as a 16-bit unsigned integer matrix of gray scale pixels. This program then used a built-in MATLAB function to convert the 16-bit integer image into a 64-bit floating-point number matrix to enable operations that require precise outputs. A cell array was then created with the matrices from each MRI slice within it.

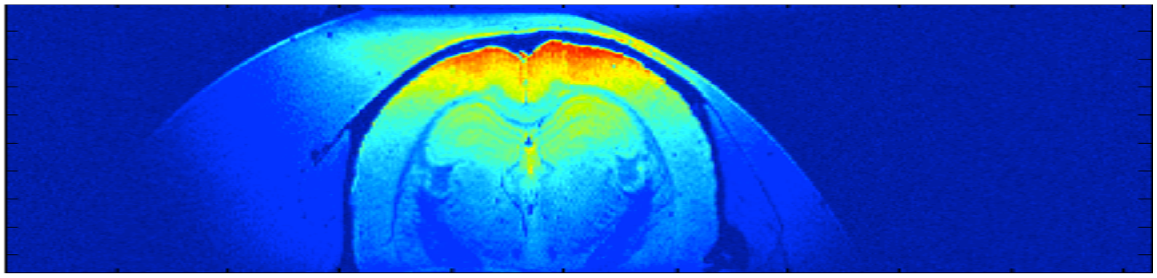

Figure 5. MR image of 10th slice in SPION treated Tg mouse brain

The first step in finding Aβ plaques within the images was to define the perimeter of the brain, thus ensuring any future findings are within the brain itself. A mask, which is a boolean image, was created to find the edge of the brain in order to differentiate the brain from the noise outside of the brain. To accomplish this, the image is displayed using the MATLAB image viewing and editing feature. Then, a freeform Region of Interest (ROI) tool was used to draw an outline around the brain. The selection is turned into an ROI object, then a mask. In this mask, the region outside of the ROI object is coded as zeros, while the region inside (the brain) is coded as ones. Later in the process this mask is used to filter out non-plaque findings.

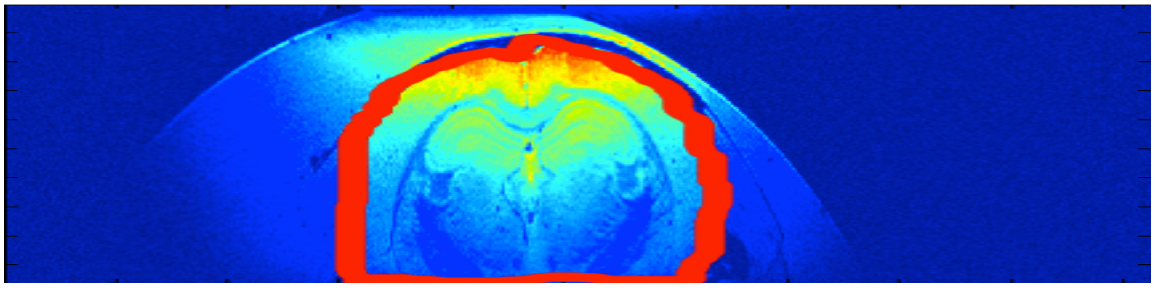

Figure 6. Freeform region of interest around brain in 10th slide

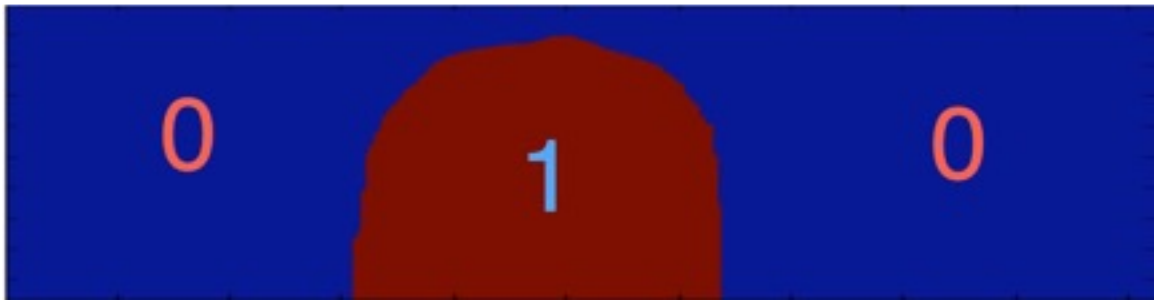

Figure 7. Boolean mask of brain in 10th slide

<u>4.3.3 Processing the Image</u>

On the MR image Aβ plaques appear as dark spots, or less intense regions, in the brain several pixels wide. However, there are numerous darker areas within the image that are not plaques. To differentiate plaques from non-plaque features or noise, a plaque-finding algorithm was created. Only plaques with a z-score greater than 5 standard deviations (σ) away from the mean are considered, because this ensures that there is no chance that a specific region is hypointense solely from noise. The z-score of each pixel is calculated by dividing the intensity of the pixel by the standard deviation of the noise. The standard deviation is the average distance from the mean pixel intensity, found using the formula below where n is the number of pixels, μ is the average intensity of the group of pixels, and x_i is the intensity of the pixel.

The standard deviation of the noise was found by defining a forty-by-forty grid of pixels on the upper left corner of the image, because they are the pixels furthest from the brain. The larger the standard deviation of the noise the less likely the pixel represents noise. Then, the image is divided by the standard deviation of the noise to create a z-score image, or an image where every pixel's intensity represents its difference from the average noise intensity. Finding the standard deviation allows the images to be standardized so that every brain can be analyzed using the same parameters.

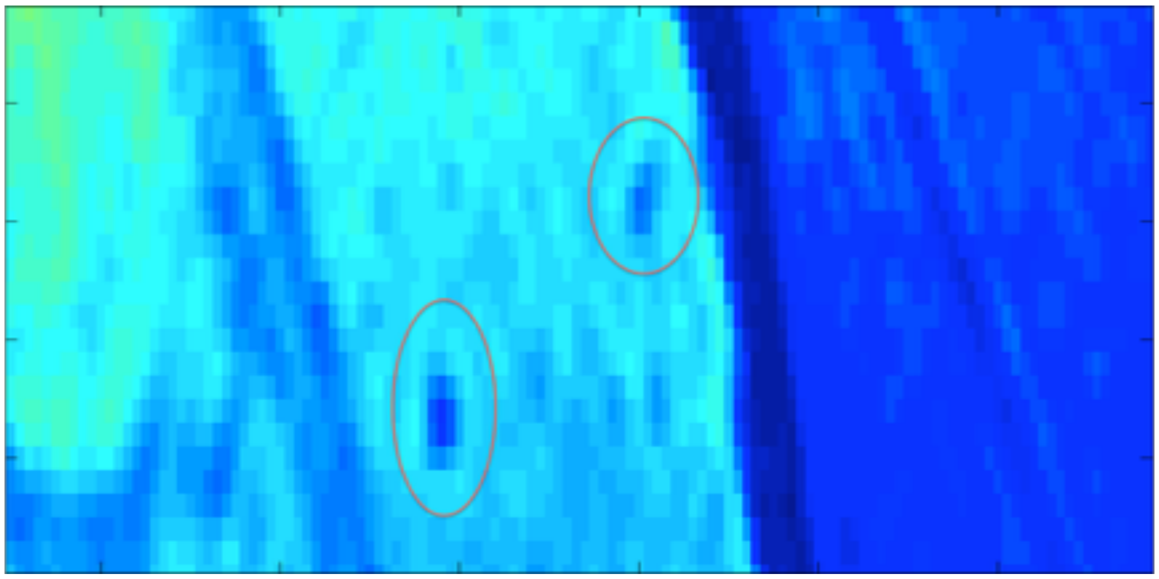

Figure 8. SPION-enhanced Aβ plaques in MR image of 10th slice

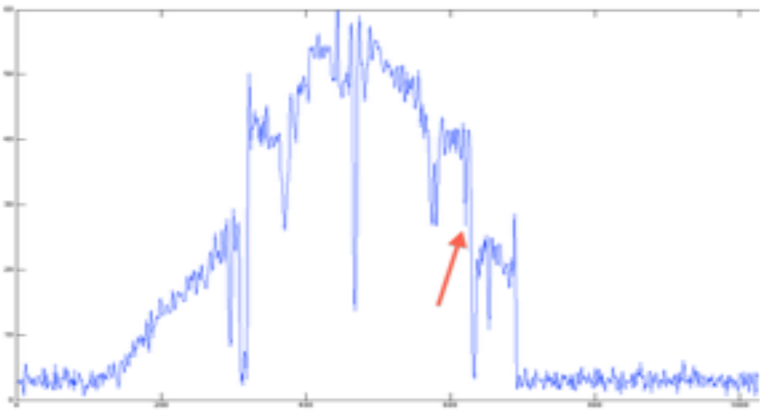

Figure 9. Z-score intensity graph of a row of pixels in 10th slice; red arrow points to small hypointense region/plaque (row runs through upper plaque in Figure 9)

The MR images have an uneven intensity distribution, so the image cannot be analyzed using the same parameters throughout the image without normalizing the intensities first. The intensity of the magnetization during the MRI is greatest at the top of the image and reduces with distance of the brain from the MRI machine. The standard deviation and intensity of the image at the top is different than it is on the bottom, as it is in the center versus the edges. Initial attempts to correct for this difference using a method of flattening the entire image increased the standard deviation of the noise and resulted in no findings. To account for the uneven intensities, the program divided the image into one-pixel-wide rows and treated each row as a separate image; however, the center of the image it is still more intense than at the edges. It was determined that the intensity of the signal of the brain itself could be flattened without impacting the noise outside of the brain. The program used the mask to allow only the part of the row that contains the brain to be flattened, without flattening the noise. Then, the row was recombined with the noise, leaving the brain flat with the same original standard deviation while the noise is at a different intensity level.

The image of the brain was flattened using a Gaussian filter, which blurs the image and then subtracts the blurred image from the original. A Gaussian blur uses a Gaussian distribution to modify the intensity of each pixel based on the pixels around it. The equation for the Gaussian distribution is:

where G is the pixel intensity at (x, y), x and y are the horizontal and vertical locations of the pixel respectively, and σ is 10 so that the program could have a sufficiently blurry image to use as a background shape. The Gaussian blur image is just the background shape of the image, meaning that when the Gaussian blur image is subtracted from the original image the background shape is removed, leaving only the intensities as they relate to the background shape.

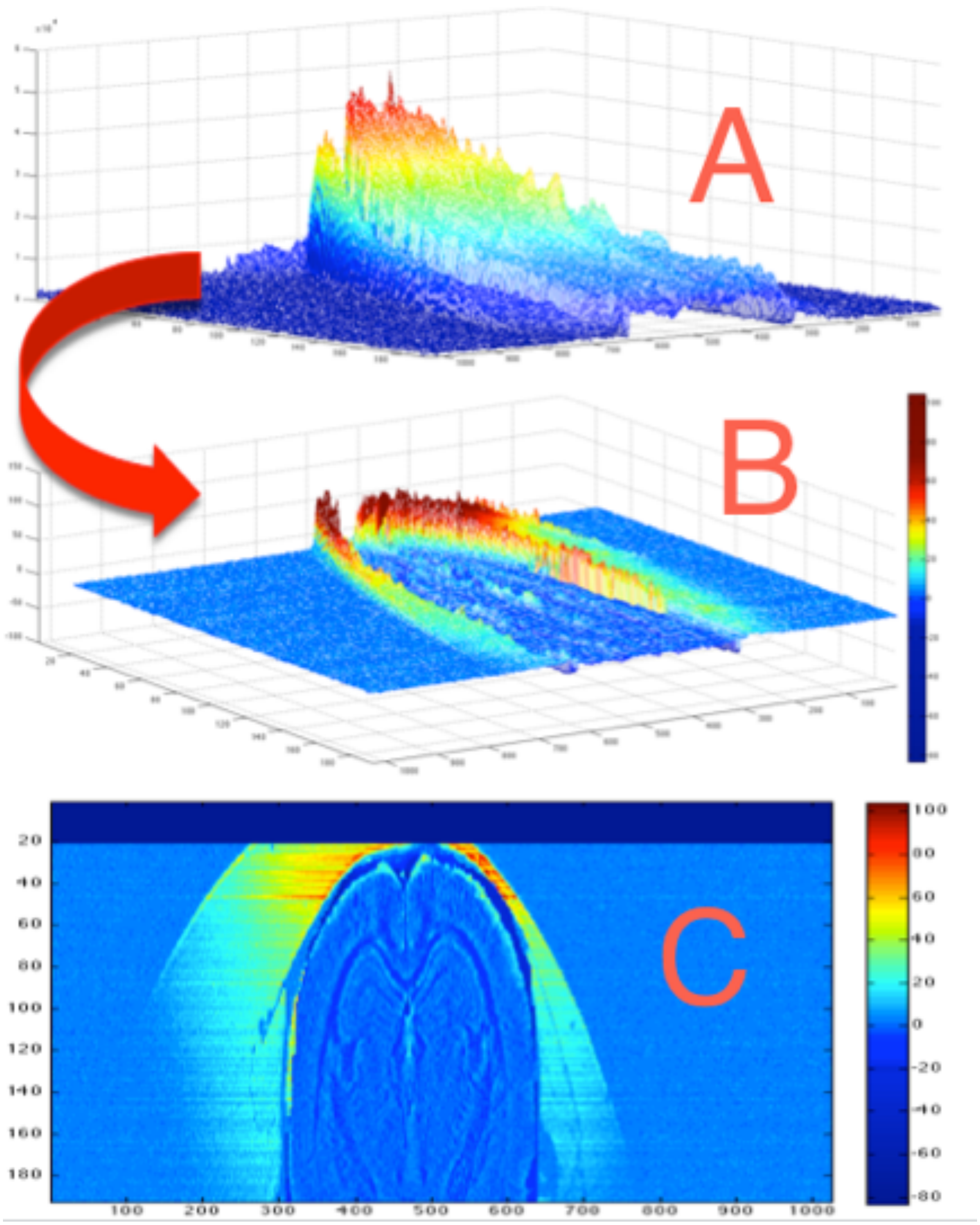

Figure 10. A: A mesh graph of intensity of the MR image before it is flattened

B: Mesh graph of the intensity of the MR image after flattening

C: Same as B but viewed as an image

The values of the intensities of the brain were then centered at zero, so the intensity of the pixels represents a difference from the average value of the brain. To accomplish this, the average value of the noise was found by using a built-in MATLAB mean-finding function. The resulting number is then subtracted from every value in the array, resulting in the noise being centered at zero. Using the mask, the program found the average intensity of only the brain, and not the noise.

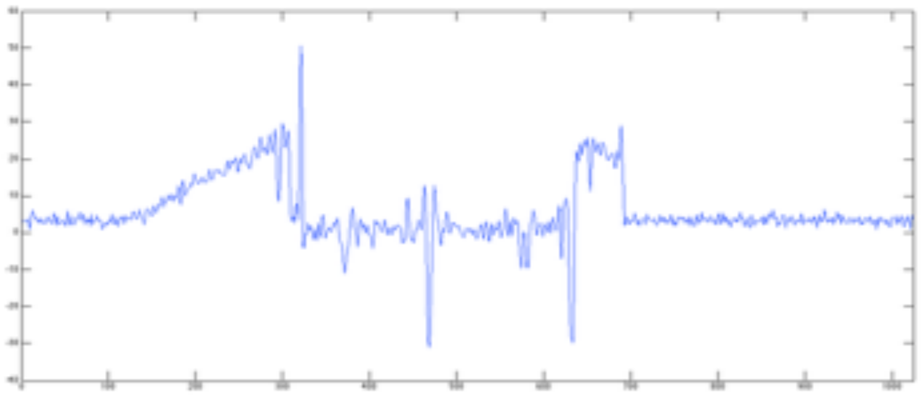

Figure 11. Z-score intensity graph of a row from 10th slice after the brain was centered at zero

Because the plaques have a lower intensity than the brain, the intensities were inverted so that the plaques will have greater values than the brain itself. The program then thresholded the image at 5, creating a binary image with values greater than 5 represented by 1 and values less than 5 are represented by 0. This binary image represents every pixel that has a distance of 5σ from the average value of the signal, based on whether the pixel value is 1 or 0. The areas that remain of this image are the plaques and the large areas in or outside the brain that are less intense than the brain itself.

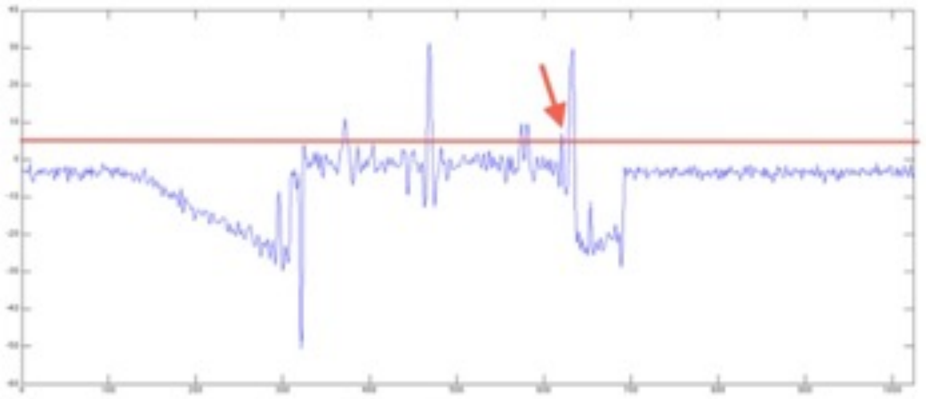

Figure 12. Inverted row with red line at 5 where the image will be thresholded; one plaque is shown above the threshold while there are multiple other spikes in the graph above 5

The binary image was then converted into a list of regions. The MATLAB function `bwboundaries` found the coordinates of every pixel inside a highlighted region and the area of that region. The program then used the coordinates found from `bwboundaries` to find the z-score of the corresponding area in the image. A list was then compiled containing each perspective plaque's coordinates in x, y, z, their z-score relative to the brain, their size in pixels, and a metric describing how close to circular they are, by calculating the ratio where P is the perimeter of the finding and A is the area in pixels (See sample data set in Appendix).

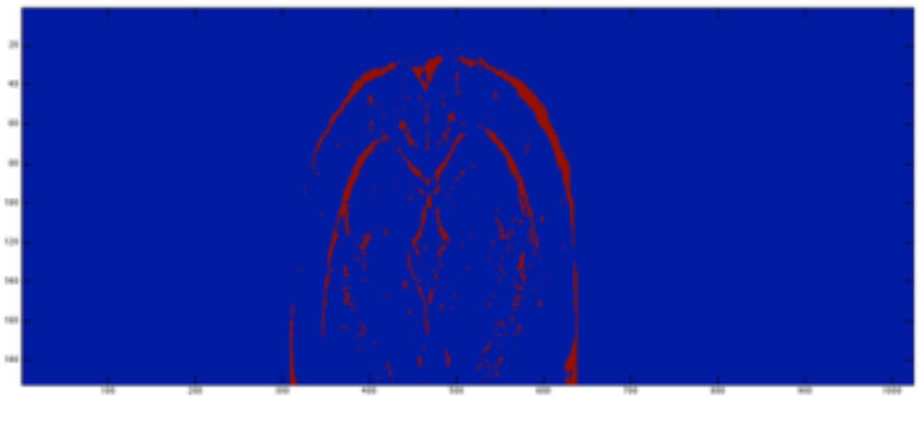

Figure 13. Binary image of findings in 10th slide before filtering non-plaque

4.3.4 Filtering Non-Plaque Findings

The compiled list of regions now contains many values that are not plaque, including components of the brain, which are dark relative to the rest of the brain. To distinguish plaques, the program filters the list of findings according to their size, z-score, and position. A parameter-sweeping script was used to find optimum upper and lower size thresholds as well as the minimum z-score. The parameter-sweeping program found that the optimum minimum size threshold is three pixels in area, the optimum maximum size threshold is ten pixels in area, and a minimum z-score threshold is 5σ. These parameters yielded the greatest possible correlation between the manual results and the automatic results created in this program.

4.3.5 Parallel Processing

In order to increase the speed and efficiency of the program, a built-in MATLAB structure `parfor` was used. This structure iterates over a set of values, running the contents simultaneously and independently on all processors and threads. The `parfor` structure allowed this program to run in around 5 seconds on a quad-core, 8-thread, Intel i7 processor.

5. RESULTS

To compare the manual results to the automated results, a MATLAB script was written that created a matrix of the number of plaques per slice for the manual results versus the number of plaques per slice by the program. From these arrays a scatter plot was made to compare the two data sets. The correlation coefficient that resulted from the comparison between the automatic results and the manual results is 0.68, which is a sufficiently large coefficient of correlation to be considered representative of the number of plaques in a brain. The automated results produced 1,723 possible plaques in the brain while the manual results contained 668 plaques, which is a difference of 1,055 plaques.

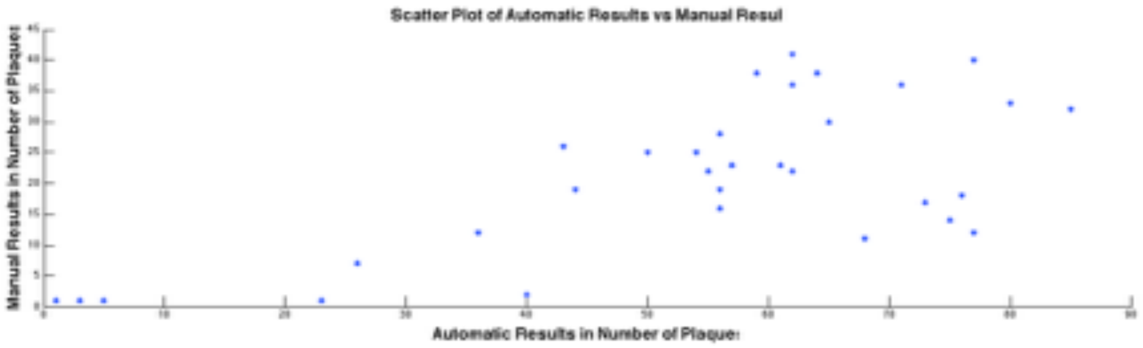

Figure 14. Scatter plot of Sillerud et al. manual results vs. automated plaques per slice

6. CONCLUSION

Alzheimer's disease is a deadly brain dysfunction that impacts millions of older adults. Currently, physicians diagnose AD by neuropsychological evaluation and the ruling out of other possible causes for dementia. A definitive diagnosis can only come after the patient's death with histological examination of the brain to identify plaques and neurofibrillary tangles. There are no diagnostic tools, in use, that can provide a diagnosis of AD while the patient is still alive. MRIs provide information regarding the volume and structure of the brain but the plaques are so small and difficult to identify that prohibitively strong MRI fields are necessary to view them. Recently, SPIONs have been introduced as a contrast agent that can be coated with organic material that will specifically bind to Aβ plaques to increase their conspicuity when viewing the MR images. When subjects are injected with the SPIONs, the plaques become visible on MRI and can be counted. Researchers at UNM are counting the plaques in MR images of transgenic mouse brains through a laborious manual process for each possible plaque. An automated means of locating and counting these plaques has not been available. Now, researchers and clinicians can use this signal processing image analysis algorithm to vastly decrease the time necessary to analyze each brain.

The correlation of 0.68 between the automated results and the manual results indicate a moderately high accuracy of the quantity, positions, intensities, sizes, and shapes of Aβ plaques in SPION-enhanced MR images of AD transgenic mouse brains. Because of this correlation, the relative amounts of plaques in the brain can be predicted from the number of plaques in the automated results. The results from this study can be used as a reasonable replacement for human labor when counting Aβ plaques in SPION-treated patients and comparing them between other automated results.

This signal processing algorithm found 1,723 plaques while the manual results found 668 plaques. The probable reason for an increase in plaques from the automated results is that the algorithm picked up more regions in the brain that have similar visual properties as plaques, but are not plaques, such as the ventricles and the corpus colossum in certain slices. The additional findings may also be related to improved accuracy with a computerized algorithm in finding plaques that are difficult to visualize. The correlation of findings, however, indicates an accurate representation of the differences in quantity between slices. This allows for a true comparison using this algorithm between different brains. It would not provide an accurate comparison between the data gathered manually on a brain and the data gathered by this program on another brain. It is critical that the algorithm provide an accurate representation of the difference in the number of plaques per brain, while the absolute number is less important.

With the accuracy of these results, this program also ran in approximately five to ten seconds for a brain. When comparing this time to analyze an MR image to the days it is currently taking, it becomes more practical to use this computer algorithm. The program is a more viable tool for measuring plaque density than the previous method of marking and counting the plaques by hand. A program that can be run concurrently, in 10 seconds, as the MRI data is being collected would make the diagnosis and treatment of AD a quick and seamless process. This automated process can also give more quantitative information of not only the number of plaque and their intensities but also values that cannot be found by hand, such as size and roundness. These will give researchers and clinicians new data points to reference, more data that they can use to determine the efficacy of drugs or the progression of the disease in patients.

This program is the culmination of many researchers looking for faster and more efficient means of diagnosing, monitoring, and treating AD. Not only will faster results allow physicians to diagnose AD after an MRI, it will also allow for the physician to monitor the progress of the disease and efficacy of the treatment modalities. This program will also save many researchers the time and money that it would take to count plaques by hand, allowing for the expedited research of future drugs and treatments for AD. Giving researchers and physicians a tool to expedite this process is the most significant original achievement of this project.

7. DISCUSSION

Developing this program has been a trying and rewarding experience. I was able to use the knowledge I have gathered from learning a variety of programming languages including C++, Java, MATLAB, and others to develop a strong knowledge of what it takes to make a scientific application. Since beginning to participate in the Supercomputing Challenge, during my freshman year, I have learned that I enjoy coding and do it at every opportunity. At this time, I am working on several computationally intense coding projects to develop iOS and web applications. I am grateful that this challenge has opened up the opportunities to pursue coding and application development.

This particular program has failed many times. I tried over twenty different ideas for how the program should work; every time something went wrong, and I decided to make it simpler. I started in the middle of last summer with an extremely complex 300+ line algorithm that did not work. This experience showed me that a huge, complex algorithm was the wrong approach to take. Then, once my program seemed to be running perfectly I found the plaques per slice correlation compared to Dr. Sillerud's manually gathered results was negative, and I had to completely redesign the algorithm again. This project has taught me to be patient and start simply and work my way toward more complexity.

Next, I will work on applying this program to the MRI data that Dr. Sillerud is using to conduct his research on the inhibition NF-κB and its effects on AD as well as pancreatic cancer. I am also looking to acquire a larger data set to apply this program to. The number of brains I have available to analyze may increase my ability to use an element of machine learning in the algorithm to decrease the number of false positives. When this program is applied to multiple brains to show the benefits of drug protocols in decreasing plaque density, Dr. Sillerud is planning on helping me publish the results in a peer-reviewed journal.

I am also looking for other problems to apply my program to; medical processes that need a large number of objects to be counted and analyzed. I am consulting with Dr. Sillerud at the UNM Department of Biochemistry and Molecular Biology to discuss possible other applications for this program. Possible applications could be identifying cancer cells, tumors, or other pathogens.

Working on this project has stimulated the idea of studying computer science and, more specifically, image processing and artificial intelligence because it is such a fascinating emerging field. Biochemistry was never a field I considered for my future but I have found the idea that my program may help alleviate personal suffering through research exciting. Completing this program in a medical field has opened up the possibility of pursuing a field like computational biology. There are so many applications of image analysis and artificial intelligence in the medical field, which could help solve the problems that are so important to people.

8. ACKNOWLEDGEMENTS

First, I would like to thank Dr. Laurel Sillerud, my mentor, who worked continuously with me developing the algorithm and teaching me the science behind what he, and myself, were working on. It has been an amazing learning experience for me. I understand what a huge opportunity it is for me to work in such a field this early in my school career. It has been an unparalleled privilege for me to be a part of his team.

Then, I would like to thank the entire research team at UNM Department of Biochemistry and Molecular Biology for their tremendous help and support throughout the project.

Also, I would like to thank my mom, Alyx Medlock, for her support and help as a sounding board in problem solving my program. Not to mention, the work she put into editing this paper to help me make it perfect. She was invaluable in helping me manage my time and the deadlines to get this project finished.

Lastly, I would like to thank my sponsor, Mr. Shanley, for helping with the grammar and editing of this paper.

9. BIBLIOGRAPHY

Aloisi, F. (2001). Immune function of microglia. *Glia, 36*, 165-179.

Alzheimer's Association. (2011). *Alzheimer's Disease Facts and Figures*. Alzheimer's Association, Washington, D.C.: Available at www.alz.org.

Ashe, K.H. & Zahs, K.R. (2010). Probing the biology of Alzheimer's disease in mice. *Neuron. 6*, 631-645.

Ashe, K.H. (2001). Learning and memory in trasngenic mice modeling Alzheimer's disease. *Learning and Memory. 8*, 301-308.

Bachstetter, A.D., Xing, B., De Almeida, L., Dimayuga, E.R., Watterson, D.M., & Van Eldik, L.J. (2011). Microglial p38α MAPK is a key regulator of proinflammatory cytokine up-regulation induced by toll-like receptor (TLR) ligands or beta-amyloid (Aβ). *Journal of Neuroinflammation. 8*, 79.

Benveniste, H. et al. (2007). Anatomical and functional phenotyping of mice models of Alzheimer's disease by microscopy. *Annals of the New York Academy of Sciences. 1097,* 12-29.

Bishop, C.M. (2006). *Pattern Recognition and Machine Learning.* New York, NY: Springer.

Bouras, C., Hof, P.R., Giannakopoulos, P., Michel, J.P., & Morrison, J.H. (1994). Regional distribution of neurofibrillary tangles and senile plaques in the cerebral cortex of elderly patients: A quantitative evaluation of a one-year autopsy population from a geriatric hospital. *Cerebral Cortex. 4*, 138-150.

Brambilla, D. et al. (2011). Nanotechnologies for Alzheimer's disease: Diagnosis, therapy, and safety issues. *Nanomedicine. 7*, 521-540.

Chamberlain, R. et al. (2009). Comparison of amyloid plaque contrast generated by T2-weighted, T2*-weighted, and susceptibility-weighted imaging methods in transgenic mouse models of Alzheimer's disease. *Magnetic Resonance Medicine. 61*, 1158-1164.

Dhenain, M. et al. (2009). Characterization of in vivo MRI detectable thalamic amyloid plaques from APP/PS1 mice. *Neurobiology of Aging. 30*, 41-53.

Fodero-Tavoletti, A.T., Villemagne, V.L., Rowe, C.C., Masters, C.L., Barnham, K.J., & Cappai, R. (2011). Amyloid-β: The seeds of darkness. *International Journal of Biochemical Cell Biology. 43*, 1247-1251.

Games, D. (1995). Alzheimer-type neuropathology in transgenic mice overexpressing V717F beta-amyloid precursor protein. *Nature. 373*, 523-527.

Granic, I., Dolga, A.M., Nijholt, I.M., Van Dijk, G., & Eisel, U.L. (2009). Inflammation and NF-κB in Alzheimer's disease and diabetes. *Journal of Alzheimer's Disease. 16*, 809-821.

Hashimoto, M., Rockenstein, E., Crews, L., & Masliah, E. (2003). Role of protein aggregation in mitochondrial dysfunction and neurodegeneration in Alzheimer's and Parkinson's diseases. *Neuromolecular Medicine. 4*, 21-36.

Herbert, L.E., Scherr, P.A., Bienias, J.L., Bennett, D.A., & Evans, D.A., (2003). Alzheimer's disease in the U.S. population: Prevalence estimates using the 2000 census. *Archives of Neurology. 60*, 1119-1122.

Hofmann-Amtenbrink, M., Von Rechenberg, B., & Hofmann, H. (2009). Superparamagnetic nanoparticles for biomedical applications. In M.C. Tan (Ed.), *Nanostructured Materials for Biomedical Applications* (pp. 120-149). Kerala, India: Hindawai Publishing Corporation.

Holmes, C. (2008). Long-term effects of Abeta-42 immunisation in Alzheimer's disease: Follow-up of a randomized, placebo-controlled phase I trial. *Lancet. 372*, 216-223.

Jack, C.R. et al. (2004). In vivo visualization of Alzheimer's amyloid plaques by MRI in transgenic mice without a contrast agent. *Magnetic Resonance Medicine. 52*, 1263-1271.

Kreutzberg, G.W. (1995). The first line of defense in brain pathologies. *Drug Research. 45*, 357-360.

Lim, J.S. (1990). *Two-dimensional signal and image processing*. Englewood Cliffs, NJ: Princtice Hall.

Lin, M.M., Kim, D.K., El Haj, A.J., & Dobson, J. (2008). Development of superparamagnetic iron oxide nanoparticles (SPIONs) for translation to clinical applications. *IEEE Transactions on Nanobioscience. 7*, 298-305.

Math Works (2012). www.mathworks.com

McKhann, G. et al. (1984). Clinical diagnosis of Alzheimer's disease. *Neurology. 34,* 939-944.

Mendez, M.F. (2006). The accurate diagnosis of early-onset dementia. *International Journal of Psychiatry Medicine. 36*, 401-412.

Mukherjee, S., Dudley, J.I., & Das, D.K. (2010). Dose-dependency of resveratrol in providing health benefits. *Dose-Response. 8*, 478-500.

Mundt, A.P et al. (2009). Targeting activated microglia in Alzheimer's pathology by intraventricular delivery of a phagocytosable MRI contrast agent in APP23 transgenic mice. *Neuroimage. 46*, 367-372.

Nattkemper, T.W. et al. (2005). Evaluation of radiological features for breast tumor classification in clinical screening with machine learning methods. *Artificial Intelligence in Medicine. 34,* 129-139.

Niolaev, A., McLaughlin, T., O'Leary, D., & Tessier-Lavigne, M. (2009). N-APP binds DR6 to cause axon pruning and neuron death via distinct caspases. *Nature. 457*, 981-989.

Podulso, J.F. et al. (2011). Targeting vascular amyloid in artierioles of Alzheimer disease transgenic mice with amyloid β protein antibody-coated nanoparticles. *Journal of Neuropathology & Experimental Neurology. 70*, 653-661.

Poduslo, J.F. et al. (2002). Molecular targeting of Alzheimer's amyloid plaques for contrast-enhanced magnetic resonance imaging. *Neurobiology of Disease. 11*, 315-329.

Rivere, C., Richard, T., Quentin, L., Krisa, S., Merillon, J.M., & Monti, J.P. (2007). Inhibitory activity of stilbenes on Alzheimer's beta-amyloid fibrils in vitro. *Bioorganic & Medicinal Chemistry. 15*, 1160-1167.

Selkoe, D.J. & Schenk, D. (2003). Alzheimer's disease: Molecular understanding predicts amyloid-based therapeutics. *Annual Review of Pharmacology & Toxicology. 43*, 545-584.

Shin, R.W., Iwaki, T., Kitamoto, T., & Tateishi, J. (1991). Hydrated autoclave pretreatment enhance tau immunoreactivity in formalin-fixed normal and Alzheimer's disease brain tissues. *Laboratory Investigation. 64*, 693-702.

Sigurdsson, E.M. et al. (2008). A non-toxic ligand for voxel-based MRI analysis of plaques in AD transgenic mice. *Neurobiology of Aging. 29*, 836-847.

Sillerud, L.O. et al. (unpublished). SPION-enhanced MRI shows that inhibition of NF-**κ**B concomitantly lowers Alzheimer's plaque formation and microglial activation in transgenic mouse brain.

Sutcliffe, J.G., Hedlund, P.B., Thomas, E.A., Bloom, F.E., & Hilbush, B.S. (2011). Peripheral reduction of β-amyloid is sufficient to reduce brain β-amyloid:

Implications for Alzheimer's disease. *Journal of Neuroscience Research. 89*, 808-814.

Taylor, R.M., Huber, D.L., Monson, T.C., Ali, A.S., Bisoffi, M., & Sillerud, L.L. (2011). Multifunctional iron platinum stealth immunomicelles: Targeted detection of human prostate cancer cells using both fluorescence and magnetic resonance imaging. *Journal of Nanoparticle Research. 13*, 4717-4729.

Tiraboschi, P., Hansen, L.A., Thal, L.J., & Corey-Bloom, J. (2004). The importance of neuritic plaques and tangles to the development and evolution of AD. *Neurology. 62*, 1984-1989.

Turner, P.R., O'Connor, K., Tate, W.P., & Abraham, W.C. (2003). Roles of amyloid precursor protein and its fragments in regulating neural activity, plasticity and memory. *Progress in Neurobiology. 70*, 1-32.

Wang, A., Das, P., Switzer, R.C., Golde, T.E., & Jankowsky, J.L. (2011). Robust amyloid clearance in a mouse model of Alzheimer's disease provides novel insights into the mechanism of amyloid-beta immunotherapy. *Journal of Neuroscience. 31*, 4124-4136.

Waring, S.C. & Rosenberg, R.N. (2008). Genome-wide association studies in Alzheimer disease. *Archives of Neurology. 65*, 329-334.

Zhang, F., Liu, J., & Shi J.S. (2010). Anti-inflammatory activities of resveratrol in the brain: Role of resveratrol in microglial activation. *European Journal of Pharmacology. 636*, 1-7.

10. APPENDIX

10.1 MATLAB Code

```
%Main file PlaqueProgram120301.m


function [list3, img6] = PlaqueProgram120301(zThreshold, sizeThreshold,
                                             suffix, file, folder,
                                             numImg, lowerThreshold)

%Author: Jordan Medlock
%University of New Mexico
%email: jordanemedlock@gmail.com
%cell: (505)264-5163

% Testing for inputs/asking for inputs
if file==0
    fileIn = uigetfile('*.tif');% the input string which represents the
                                path to the image file
else
    fileIn = file;
end

if folder==0
    folder = uigetdir;  % the string which represents the path to the
                        directory which holds the masks
end                     % if nil asks user to create masks
if isempty(suffix)
    date = input('file suffix:', 's'); %suffix to use to create output
                                       folder
else
    date = suffix;
end
if isempty(numImg)
    numImg = input('number of images in file:'); %number of images in
                                                 composite
end
if isempty(sizeThreshold)
    sizeThreshold = input('upper threshold for plaque size in
pixels:'); %just as it says
end
if isempty(lowerThreshold)
    lowerThreshold = input('lower threshold for plaque size in
pixels:'); %just as it says
end
if isempty(zThreshold)
    zThreshold = input('lower threshold for plaques z-score:');%just as
                                                              it says
end
```

```
%getting main function to run application
[list3, img6] = plaqueProgram(numImg, fileIn, date, folder,
                              sizeThreshold, zThreshold,
                              lowerThreshold);
end




function [list3, imgbinarized] = plaqueProgram(numImg, fileIn, date,
                                               folder, sizeThreshold,
                                               zThreshold,
                                               lowerThreshold)
%Author: Jordan Medlock
%University of New Mexico
%Email: jordanemedlock@gmail.com
%Cell: (505)264-5163



outputFile = 'finalImg'; %constants to output names
outputfolder = 'outputFolder'; %constants to output names

mkdir(sprintf('%-s_%-s', outputfolder, date)); %making the directory to
                                               hold the outputs

threshold = sizeThreshold;
dash = filesep; %this line makes the program not only work on mac but
                 also windows and linux

tic;                     %start timer
img = cell(numImg);      %initialize variables
mask = cell(numImg);
for i=1:numImg                                      %for each image
    img{i} = double(imread(fileIn, i));             %read in the file at
                                                    index i and assign it
                                                    to the cell array 'img'
    if folder==0                                    % if there is no input
                                                    folder
        [mask{i}, ~] = cropImg(img{i}, i);          %create the masks and
                                                    assign them to the cell
                                                    array 'mask'
        imwrite(mask{i},sprintf(
            'outputFolder_%-s%-smask%-d.tif',date,dash,i)); %write
            'mask' to the output folder
    else
        mask{i} = double(imread(sprintf(
            '%-s%-smask%-d.tif',folder,dash,i))); %import the image and
```

```
                                                            convert it to a
                                                            double so that
                                                            mathematical
                                                            operations can be
                                                            performed on it

    end
end

imgfinal = cell(numImg); %initialize some more variables
stats = cell(numImg);
list2 = cell(numImg);
parfor i=1:numImg %parallel process these lines of code accross each
                slice in the brain

    %old code :-) failed experement

    %img1 = double(flattenImgMinus(img, 10));
    %[img2, ~] = makeZScoreImg(img1, noiseSTD{i});
    %img3 = double(findAreaSubtractedImg(img2, outerWidth, innerWidth,
     numImg, i));
    %                                imgbinarized = cell(numImg);
    %img4 = binarize(img3, 2.5);
    %[img5, stats{i}] = findXY(img4, threshold);
    %img6{i} = double(findPlaqueZScore(img3, img5));


    [m,n] = size(img{i}); %find the size of the image so that the
program can reference it and split it line by line
    for j=1:m
        %j=x(2); %line for debugging
        imgsplit = img{i}(j,:); %splits image row by row
        masksplit = mask{i}(j,:);                            %splits
                            mask row by row
        [imgstand,~,noiseavg] = makeZScoreImg(imgsplit);
                            %standardizes the image and finds the
                            average value of the noise
        brainavg = brainAverage(masksplit,imgstand);         %finds the
                            average value only of the brain using the
                            split mask
        brainavgmask = (-(masksplit-1))*brainavg;            %creates a
                            non-binary mask to apply to the image so
                            that the brain does not curve off at the
                            edges when flattened
        tobeflattened = brainavgmask + imgstand.*masksplit; %masks the
                            standardized image to be flattened
        flattenedonce = flattenImgMinus(tobeflattened,10).*masksplit;
                            %flattens the image with a standard
                            deviation of 10

        puttogether = flattenedonce+(imgstand.*(-masksplit+1));
                            %puts the image back together so that the
                            brain is flattened and the noise remains the
                            same
        minusnoise = puttogether-noiseavg;                  %subtracts
                               the noise puts the noise at zero
        minusbrain = minusnoise-brainAverage(masksplit,minusnoise);
                               %centers brain at zeros by subtracting the
                               average of the brain
        imgnegative(j,:) = -minusbrain;                     %inverts
                               the image around the brain
        imgbinarized(j,:) = binarize(imgnegative(j,:),zThreshold);
                               %binarizes the image at 'zScore' and the
                               indeces create the entire out of the rows
    end

    [imgbinarized, stats{i}] = findXY(imgbinarized, threshold); %takes
                               the binarized image and creates a list of
                               the boundaries and statistics about those
                               boundaries
    imgfinal{i} = double(findPlaqueZScore(imgnegative,imgbinarized));
                               %uses the binarized image as a mask over the
                               z-score image so that only plaque show up
    imwrite(uint16(imgfinal{i}), sprintf(
                               'outputFolder_%-s%-s%-s_%-s_%02.0f.tif',
                               date, dash, outputFile, date, i)); %outputs
                               the final image
    list2{i} = zeros(size([length(stats{i}),1:6]));     %initializes
                               'list2' note: 'list' is a built in function
                               which cannot be used as a variable name
    for j=1:length(stats{i})                 %for each boundary in the
                               'stats' list

        if length(stats{i}(j).PixelList)<1  %if there are no pixels in
                               'stats{i}(j)' skip it
            continue;
        end
                               %create the list from each instance variable
                               in the 'stats' object
        list2{i}(j,1:6) = [stats{i}(j).PixelList(1,1), stats{i}
                               (j).PixelList(1,2),
                               i,
                               imgnegative(stats{i}(j).PixelList(1,2),
                               stats{i}(j).PixelList(1,1)),  stats{i}
                               (j).Area, stats{i}(j).Perimeter/
                               (2*sqrt(pi*stats{i}(j).Area))];
    end
end


list3 = []; %initializes 'list3' as a blank array to hold each piece of
               'list2' in it
for i=1:numImg % for each slice
    list3 = [list3; list2{i}]; %add the last line of 'list2' to the
                                   last line in 'list3'
end
i = 1; %initialize the index
```

```
while i<length(list3)   %for all lines in 'list3', 'list3' changes
                        sizes often therefore it cannot use a 'for'
                        loop
    length(list3)       %output just to know its working :-)
    if list3(i,1)==0 ||
       list3(i,4) < zThreshold || mask{list3(i,3)}(list3(i,2),list3(i,
       1))==0 ||
       list3(i,5) > threshold ||
       list3(i,5) <= lowerThreshold %things that it'll cut out of
                                    'list3'


       list3(i,:) = [];%deleting that line
    else
        i = i + 1;      %one problem with matlab is that you cannot use
                         easy i++ incrementation
    end                 %lol going from c++ and not having '++' in
                         matlab was anoying
end
csvwrite(sprintf('outputFolder_%-s%-sfinalList_%-s.csv', date, dash,
date), list3);%write the output list as a csv
beep; %BEEP! done :-)
toc; % end timer
end % WOOHOO! Its done!







function img = flattenImg(imgIn, sigma) %uses a gaussian blur to divide
                                         the image and flatten it
%Author: Jordan Medlock
%University of New Mexico
%email: jordanemedlock@gmail.com
%cell: (505)264-5163



filter = fspecial('gaussian', size(imgIn), sigma);  %create the
                        gaussian filter for the images

img2 = imfilter(imgIn, filter)+0.000000000001;  %filter the image

img = double(imgIn)./double(img2); %./ means that it divides each cell
                        in the matrix by each the corresponding cell in
                        the other matrix


end

function img = flattenImgMinus(imgIn, sigma) %same thing as the other
                        one although it subtracts instead of divides
```

```
%Author: Jordan Medlock
%University of New Mexico
%email: jordanemedlock@gmail.com
%cell: (505)264-5163



filter = fspecial('gaussian', size(imgIn), sigma);  %create the
                        gaussian filter for the images

img2 = imfilter(imgIn, filter);  %filter the image

img = imgIn-img2;

end





function [img, noiseSTD, noiseHeight] = makeZScoreImg (imgIn, noiseSTD)
%makes a standardized image finds tha standard deviation of the noise,
and the average intenbsity of it

%Author: Jordan Medlock
%University of New Mexico
%Email: jordanemedlock@gmail.com
%Cell: (505)264-5163

[m,n] = size(imgIn);
if m > 40
    m=40;
end                     %deciding where to put the noise
if n > 40
    n=40;
end
if exist('noiseSTD', 'var') %if noiseSTD exists use it

    img = imgIn/noiseSTD;
else                        %if it doesnt create it
    noiseSTD = std2(imgIn(1:m,1:n));
    noiseHeight = mean2(imgIn(1:m,1:n));
    img = imgIn/noiseSTD;
end
end

function [imgOut] = findAreaSubtractedImg(imgIn, w, r, numImg, imgNum)
      %failed experement :( subtracts the surrounding pixels from each
      pixel in the image and creates an image from the output
      %very unnatural

%Author: Jordan Medlock
%University Of New Mexico
```

```
%Email: jordanemedlock@gmail.com
%Cell: (505)264-5163


[m,n] = size(imgIn);

imgOut = zeros(size(imgIn));

for i=1:m

    for j=1:n
        %waitbar(((((j-1)/n+i-1)/m+imgNum-1)/numImg), waitBar,
sprintf('%-.2f%%, image %-.0f of %-.0f, line %-.0f of %-.0f', ((((j-1)/
n+i-1)/m+imgNum-1)/numImg)*100,  imgNum, numImg, i, m));
        %piece of code to find time remaining: 32/60*cputime/((j-1)/n
+i-1)
        %didnt quite work...



        if i<=r
            if j<=r

                sumA = sum(imgIn(1:(i+w), 1:(j+w)));
                sumA = sum(sumA);
                areaA = (i+w)*(j+w);

                sumB = sum(imgIn(1:(i+r), 1:(j+r)));
                sumB = sum(sumB);
                areaB = (i+r)*(j+r);

                mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
                imgOut(i,j) = mm;
            elseif j<=w

                sumA = sum(imgIn(1:(i+w), 1:(j+w)));
                sumA = sum(sumA);
                areaA = (i+w)*(j+w);

                sumB = sum(imgIn(1:(i+r), (j-r):(j+r)));
                sumB = sum(sumB);
                areaB = (i+r)*(2*r+1);

                mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
                imgOut(i,j) = mm;
            elseif (n-w)>j>(w)

                sumA = sum(imgIn(1:(i+w), (j-w):(j+w)));
                sumA = sum(sumA);
                areaA = (i+w)*(2*w+1);

                sumB = sum(imgIn(1:(i+r), (j-r):(j+r)));
                sumB = sum(sumB);
                areaB = (i+r)*(2*r+1);
                mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
                imgOut(i,j) = mm;
            elseif (n-r)>=j>=(n-w)

                sumA = sum(imgIn(1:(i+w), (j-w):n));
                sumA = sum(sumA);
                areaA = (i+w)*(n-j+w+1);

                sumB = sum(imgIn(1:(i+r),(j-r):(j+r)));
                sumB = sum(sumB);
                areaB = (i+r)*(2*r+1);

                mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
                imgOut(i,j) = mm;
            else

                sumA = sum(imgIn(1:(i+w), (j-w):n));
                sumA = sum(sumA);
                areaA = (i+w)*(n-j+w+1);

                sumB = sum(imgIn(1:(i+r),(j-r):n));
                sumB = sum(sumB);
                areaB= (i+r)*(n-j+r+1);

                mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
                imgOut(i,j) = mm;
            end


        elseif i<=w
            if j<=r

                sumA = sum(imgIn(1:(i+w), 1:(j+w)));
                sumA = sum(sumA);
                areaA = (i+w)*(j+w);

                sumB = sum(imgIn((i-r):(i+r), 1:(j+r)));
                sumB = sum(sumB);
                areaB = (2*r+1)*(j+r);

                mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
                imgOut(i,j) = mm;
            elseif j<=w

                sumA = sum(imgIn(1:(i+w), 1:(j+w)));
                sumA = sum(sumA);
                areaA = (i+w)*(j+w);

                sumB = sum(imgIn((i-r):(i+r), (j-r):(j+r)));
                sumB = sum(sumB);
                areaB = (2*r+1)*(2*r+1);

                mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
                imgOut(i,j) = mm;
            elseif (n-w)>j>(w)

                sumA = sum(imgIn(1:(i+w), (j-w):(j+w)));
                sumA = sum(sumA);
                areaA = (i+w)*(2*w+1);

                sumB = sum(imgIn((i-r):(i+r), (j-r):(j+r)));
                sumB = sum(sumB);
                areaB = (2*r+1)*(2*r+1);

                mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
                imgOut(i,j) = mm;
            elseif (n-r)>=j>=(n-w)

                sumA = sum(imgIn(1:(i+w), (j-w):n));
                sumA = sum(sumA);
                areaA = (i+w)*(n-j+w+1);

                sumB = sum(imgIn((i-r):(i+r), (j-r):(j+r)));
                sumB = sum(sumB);
                areaB = (2*r+1)*(2*r+1);

                mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
                imgOut(i,j) = mm;
            else

                sumA = sum(imgIn(1:(i+w), (j-w):n));
                sumA = sum(sumA);
                areaA = (i+w)*(n-j+w+1);

                sumB = sum(imgIn((i-r):(i+r),(j-r):n));
                sumB = sum(sumB);
                areaB= (2*r+1)*(n-j+r+1);

                mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
                imgOut(i,j) = mm;
            end


        elseif (m-w)>i>w
            if j<=r

                sumA = sum(imgIn((i-w):(i+w), 1:(j+w)));
                sumA = sum(sumA);
                areaA = (2*w+1)*(j+w);

                sumB = sum(imgIn((i-r):(i+r), 1:(j+r)));
                sumB = sum(sumB);
                areaB = (2*r+1)*(j+r);

                mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
                imgOut(i,j) = mm;
            elseif j<=w
```

```
            sumA = sum(imgIn((i-w):(i+w), 1:(j+w)));
            sumA = sum(sumA);
            areaA = (2*w+1)*(j+w);

            sumB = sum(imgIn((i-r):(i+r), (j-r):(j+r)));
            sumB = sum(sumB);
            areaB = (2*r+1)*(2*r+1);

            mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
            imgOut(i,j) = mm;
        elseif (n-w)>j>(w)

            sumA = sum(imgIn((i-w):(i+w), (j-w):(j+w)));
            sumA = sum(sumA);
            areaA = (2*w+1)*(2*w+1);

            sumB = sum(imgIn((i-r):(i+r), (j-r):(j+r)));
            sumB = sum(sumB);
            areaB = (2*r+1)*(2*r+1);

            mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
            imgOut(i,j) = mm;
        elseif (n-r)>=j>=(n-w)

            sumA = sum(imgIn((i-w):(i+w), (j-w):n));
            sumA = sum(sumA);
            areaA = (2*w+1)*(n-j+w+1);

            sumB = sum(imgIn((i-r):(i+r), (j-r):(j+r)));
            sumB = sum(sumB);
            areaB = (2*r+1)*(2*r+1);

            mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
            imgOut(i,j) = mm;
        else

            sumA = sum(imgIn((i-w):(i+w), (j-w):n));
            sumA = sum(sumA);
            areaA = (2*w+1)*(n-j+w+1);

            sumB = sum(ingIn((i-r):(i+r),(j-r):n));
            sumB = sum(sumB);
            areaB= (2*r+1)*(n-j+r+1);

            mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
            imgOut(i,j) = mm;
        end



    elseif (m-r)>i>=(m-w)
        if j<=r
            sumA = sum(imgIn((i-w):m, 1:(j+w)));
            sumA = sum(sumA);
            areaA = (m-i+w+1)*(j+w);

            sumB = sum(imgIn((i-r):(i+r), 1:(j+r)));
            sumB = sum(sumB);
            areaB = (2*r+1)*(j+r);

            mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
            imgOut(i,j) = mm;
        elseif j<=w

            sumA = sum(imgIn((i-w):m, 1:(j+w)));
            sumA = sum(sumA);
            areaA = (m-i+w+1)*(j+w);

            sumB = sum(imgIn((i-r):(i+r), (j-r):(j+r)));
            sumB = sum(sumB);
            areaB = (2*r+1)*(2*r+1);

            mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
            imgOut(i,j) = mm;
        elseif (n-w)>j>(w)

            sumA = sum(imgIn((i-w):m, (j-w):(j+w)));
            sumA = sum(sumA);
            areaA = (m-i+w+1)*(2*w+1);

            sumB = sum(imgIn((i-r):(i+r), (j-r):(j+r)));
            sumB = sum(sumB);
            areaB = (2*r+1)*(2*r+1);

            mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
            imgOut(i,j) = mm;
        elseif (n-r)>=j>=(n-w)

            sumA = sum(imgIn((i-w):m, (j-w):n));
            sumA = sum(sumA);
            areaA = (m-i+w+1)*(n-j+w+1);

            sumB = sum(imgIn((i-r):(i+r), (j-r):(j+r)));
            sumB = sum(sumB);
            areaB = (2*r+1)*(2*r+1);

            mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
            imgOut(i,j) = mm;
        else

            sumA = sum(imgIn((i-w):m, (j-w):n));
            sumA = sum(sumA);
            areaA = (m-i+w+1)*(n-j+w+1);

            sumB = sum(ingIn((i-r):(i+r),(j-r):n));
```

```
            sumB = sum(sumB);
            areaB= (2*r+1)*(n-j+r+1);

            mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
            imgOut(i,j) = mm;
        end

    else
        if j<=r

            sumA = sum(imgIn((i-w):m, 1:(j+w)));
            sumA = sum(sumA);
            areaA = (m-i+w+1)*(j+w);

            sumB = sum(imgIn((i-r):m, 1:(j+r)));
            sumB = sum(sumB);
            areaB = (m-i+r+1)*(j+r);

            mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
            imgOut(i,j) = mm;
        elseif j<=w

            sumA = sum(imgIn((i-w):m, 1:(j+w)));
            sumA = sum(sumA);
            areaA = (m-i+w+1)*(j+w);

            sumB = sum(imgIn((i-r):m, (j-r):(j+r)));
            sumB = sum(sumB);
            areaB = (m-i+r+1)*(2*r+1);

            mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
            imgOut(i,j) = mm;
        elseif (n-w)>j>(w)

            sumA = sum(imgIn((i-w):m, (j-w):(j+w)));
            sumA = sum(sumA);
            areaA = (m-i+w+1)*(2*w+1);

            sumB = sum(imgIn((i-r):m, (j-r):(j+r)));
            sumB = sum(sumB);
            areaB = (m-i+r+1)*(2*r+1);

            mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
            imgOut(i,j) = mm;
        elseif (n-r)>=j>=(n-w)

            sumA = sum(imgIn((i-w):m, (j-w):n));
            sumA = sum(sumA);
            areaA = (m-i+w+1)*(n-j+w+1);

            sumB = sum(imgIn((i-r):m, (j-r):(j+r)));
            sumB = sum(sumB);
            areaB = (m-i+r+1)*(2*r+1);
```

```
                mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
                imgOut(i,j) = mm;
            else

                sumA = sum(imgIn((i-w):m, (j-w):n));
                sumA = sum(sumA);
                areaA = (m-i+w+1)*(n-j+w+1);

                sumB = sum(imgIn((i-r):m,(j-r):n));
                sumB = sum(sumB);
                areaB= (m-i+r+1)*(n-j+r+1);

                mm = (sumA- sumB)/(areaA- areaB)-imgIn(i,j);
                imgOut(i,j) = mm;
            end
        end
    end
end
end




function newImg = binarize(img, threshold)
%Author: Jordan Medlock
%Email: jordanemedlock@gmial.com
%Cell: (505)264-5163
%last edited 110815




[m,n]=size(img);

newImg = zeros(size(img));

for i=1:m
    for j=1:n
        if img(i,j)>threshold
            newImg(i,j)=1;
        else
            newImg(i,j)=0;
        end
    end
end

end
```

```
function [L, stats] = findXY(img, threshold) % finds the boundaries of
                        a binary image and outputs them in a list
%Author: Jordan Medlock
%Email: jordanemedlock@gmail.com
%Cell: (505)264-5163
%last edited: 110816


[~,L] = bwboundaries(img);
stats = regionprops(L,'Area','PixelList','Perimeter');


end



function imgOut = findPlaqueZScore(zScoreImg, plaqueOnlyImg) %finds the
                        z-score of only the plaque regions
%Author: Jordan Medlock
%email: jordanemedlock@gmail.com
%cell: (505)264-5163
%last edited 110809



img = binarize(plaqueOnlyImg, 1);

imgOut = img.*zScoreImg;
end





function [mask, outImg] = cropImg(imgIn, i) %asks the user to crop the
                        image and create a mask

img = imgIn;

imagesc(img); %display the image

title(sprintf('image %.0f', i)); % put its title so the user knows
                        whats going on
h = imfreehand('closed', true);  %ask the user to create a freehand
                        region of interest around the brain
h.wait();                        %wait till they are done
mask = h.createMask();           %create a binary mask from the ROI
                        object
outImg = img.*mask;              %mask the image
end
```

```
function plaques = subtractAreaFromCandidates(img, candidates,
                        threshold, w, r) %failed exprement :-( supposed
                        to do the job of the other failed experement
                        but that didnt do a good job either
plaques = candidates;
for i=1:length(candidates)
    sum = zeros([length(candidates),1]);
    num = zeros([length(candidates),1]);
    for j=-r-1:r+1
        for k=-r-1:r+1
            if sqrt(j^2 + k^2) <  w || sqrt(j^2 + k^2) > r
                continue;
            end
            if candidates(i,1) + j <=0 || candidates(i,2) + k <=0
                continue;
            end
            [m,n] = size(img);
            if candidates(i,1) + j > m || candidates(i,2) + k > n
                continue;
            end
            num(i) = num(i) + 1;
            sum(i) = sum(i) + img(candidates(i,2) + j, candidates(i,1)
+ k);
        end
    end

end
a = 1;
while a < length(plaques)
    if img(plaques(a,2),plaques(a,1))-(sum(a)/num(a)) < threshold
        plaques(a,:) = [];
        sum(a) = [];
        num(a) = [];
    else
        plaques(a,3) = img(plaques(a,2),plaques(a,1))-(sum(a)/num(a));
        a = a + 1;
    end
end



end





function brainAvg = brainAverage(mask, imgIn) %find the average value
                        of the brain rather than the whole image
img   = imgIn;
[m,n] = size(imgIn);
```

```
sum = 0;
num = 0;
for i=1:m
    for j=1:n
        if mask(i,j) == 1
            sum = sum + img(i,j);
            num = num + 1;
        end
    end
end
brainAvg = (sum/num);


end



%Finally!!!!
```

```
%%%%%%%%ParameterSweeping.m script

readS5('s5coords.txt');
s5=s5coords;
s5(:,3)=s5(:,3)./10;
output = zeros(10,20);
a=1;
compS5 = zeros(32);
for l=1:32
    [m,n] = size(s5);
    o=1;
    while o
        compS5(l)=compS5(l)+1;
        a=a+1;
        if a < m
            o = l==s5(a,3);
        else
            o = 0;
        end
    end

end
```

```
for i=1:10
    for j=i:20

        list3=PlaqueProgram120301(2.5, j, 120229, 'mase_s5.tif',
'outputFolder_S5_120315', 32, i);

        [m,n] = size(list3);
        compAuto = zeros(32);
        a=1;
        for l=1:32
            o=1;
            while o
                compAuto(l) = compAuto(l)+1;
                a=a+1;
                if a < m
                    o = l==list3(a,3);
                else
                    o=0;
                end
            end

        end
        [i,j]
        output(i,j)=corr(compAuto(3:29,1),compS5(3:29,1))
    end
end
beep;
beep;
beep;




%%%%%%%%%Compare.m script

readS5('s5coords.txt');
s5=s5coords;
s5(:,3)=s5(:,3)./10;
a=1;
compS5 = zeros(32);
for l=1:32
    [m,~] = size(s5);
    o=1;
    while o
        compS5(l)=compS5(l)+1;
        a=a+1;
        if a < m
            o = l==s5(a,3);
        else
            o = 0;
        end
    end
```

```
end

i=1;
list2_5_7_17=PlaqueProgram120301(2.5, 17, '120229', 'mase_s5.tif',
'outputFolder_120208_6', 32, 7);
list2_5_2_10=PlaqueProgram120301(2.5, 10, '120229', 'mase_s5.tif',
'outputFolder_120208_6', 32, 2);

[m,n] = size(list2_5_7_17);
compAuto = zeros(32);
a=1;
for l=1:32

    o=1;
    while o
        compAuto(l) = compAuto(l)+1;
        a=a+1;
        if a < m
            o = l==list3(a,3);
        else
            o=0;
        end
    end

end

output(1)=corr(compAuto(:,1),compS5(:,1));



beep;
beep;
beep;





%%%%%%%demo.m
i=1;
img = cell(2,1);
img{i} = double(imread('mase_s5.tif',10));

zThreshold = 2.5;
mask{i} = double(imread('outputFolder_120208_6/mask10.tif'));
imagesc(img{i});
h = impoint();
h.wait();
x = uint16(h.getPosition());

[m,n] = size(img{i});
%for j=1:m
j=x(2);
```

```
    imgsplit = img{i}(j,:);
    masksplit = mask{i}(j,:);
    [imgstand,~,noiseavg] = makeZScoreImg(imgsplit);
    brainavg = brainAverage(masksplit,imgstand);
    brainavgmask = (-(masksplit-1))*brainavg;
    tobeflattened = brainavgmask + imgstand.*masksplit;
    flattenedonce = flattenImgMinus(tobeflattened,10).*masksplit;

    puttogether = flattenedonce+(imgstand.*(-masksplit+1));
    minusnoise = puttogether-noiseavg;
    minusbrain = minusnoise-brainAverage(masksplit,minusnoise);
    negative(j,:) = -minusbrain;
    binary(j,:) = binarize(negative(j,:),5);
%end


[m,n] = size(img{i});
for j=1:m

    imgsplit2d(j,:) = img{i}(j,:);
    masksplit2d(j,:) = mask{i}(j,:);
    [imgstand2d(j,:),~,noiseavg2d(j,:)] =
makeZScoreImg(imgsplit2d(j,:));
    brainavg2d(j,:) = brainAverage(masksplit2d(j,:),imgstand2d(j,:));
    brainavgmask2d(j,:) = (-(masksplit2d(j,:)-1))*brainavg2d(j,:);
    tobeflattened2d(j,:) = brainavgmask2d(j,:) +
imgstand2d(j,:).*masksplit2d(j,:);
    flattenedonce2d(j,:) = flattenImgMinus(tobeflattened2d(j,:),
10).*masksplit2d(j,:);

    puttogether2d(j,:) = flattenedonce2d(j,:)+(imgstand2d(j,:).*(-
masksplit2d(j,:)+1));
    minusnoise2d(j,:) = puttogether2d(j,:)-noiseavg2d(j,:);
    minusbrain2d(j,:) = minusnoise2d(j,:)-
brainAverage(masksplit2d(j,:),minusnoise2d(j,:));
    negative2d(j,:) = -minusbrain2d(j,:);
    binary2d(j,:) = binarize(negative2d(j,:),5);
end
```

10.2 Sample List of Findings for S5

x (px)	y (px)	z (slices)	z-score (SD)	area (px)	roundness closer to 1 is more round
383	57	1	5.31389986650220	6	0.786393873897061
455	49	1	6.81606202892931	8	0.822084468062250
461	52	1	5.06738755789502	6	0.786393873897061
469	48	1	5.71665769003585	8	1.07997935257674
572	57	1	5.19151973385078	4	0.963131863949189
297	132	3	6.07815770859543	5	1.03986373924986
382	110	3	6.60138723169894	4	0.564189583547756
403	77	3	5.86092255911581	5	0.683038916019309
425	39	3	6.42988844909019	7	1.18041728111098
436	55	3	6.26445082693567	5	0.713649646461109
437	78	3	5.96030171031703	6	0.786393873897061
450	125	3	5.43649489290419	6	0.949261377864701
451	33	3	5.48736528646123	4	0.681037072175311
454	63	3	6.09605547162596	4	0.681037072175311
454	104	3	5.65846609186770	5	0.683038916019309
455	132	3	5.88334571371092	5	0.683038916019309
464	56	3	6.09242838619616	5	0.935352168221325
469	158	3	7.80993959130433	5	0.861451327634586
474	100	3	5.12034897160334	10	0.861451327634586
480	130	3	5.02855781206426	5	0.861451327634586
484	123	3	5.26835651328861	9	0.830151103815378
488	134	3	5.29249146546113	4	0.763660723748473
498	93	3	5.59786663072441	7	0.728059254791900
500	38	3	9.93846804846567	6	0.690988298942671
521	76	3	6.53233718500042	6	0.853855802910311
523	81	3	5.32778465580986	9	0.752252778063675
540	79	3	7.40368838692621	6	0.690988298942671
546	132	3	6.34491304893277	4	0.564189583547756
550	77	3	7.88945889671868	9	0.775068669433270
558	65	3	5.73948176091358	4	0.564189583547756
632	158	3	5.10640855133784	7	1.24287489096088
633	154	3	5.41060722495354	7	1.02963127233795
299	144	4	5.62718925101854	9	0.830151103815378
302	139	4	6.44515202611410	5	1.11376457983660
359	134	4	6.95453599282405	7	0.728059254791900
369	80	4	7.28835422399129	5	0.787550487047847
379	112	4	5.20951074709501	4	0.681037072175311
382	107	4	5.58040768990441	4	0.564189583547756
390	128	4	5.90966092233452	6	0.718931944883810
391	98	4	5.05272313940031	4	0.763660723748473
392	95	4	6.18092736700214	8	1.07997935257674
398	140	4	6.20602942795813	5	0.861451327634586
399	90	4	6.04826820193784	10	1.21827615086514
405	86	4	5.11949613540628	4	0.564189583547756
413	124	4	5.16237463594234	8	1.16260300414991
417	73	4	6.69432168843741	4	0.763660723748473
417	84	4	5.15232213882268	5	0.787550487047847
423	61	4	5.54748850161786	6	0.690988298942671
464	63	4	5.22160602597933	4	0.763660723748473
465	67	4	5.99968936640585	7	0.790516864641796
482	50	4	8.64695810966032	4	0.564189583547756
482	80	4	5.29513436569209	4	0.880508212376027
486	80	4	5.22034514037046	4	0.880508212376027
490	86	4	7.22232203566978	4	0.564189583547756
493	86	4	5.74475241763457	8	0.763660723748473
496	119	4	5.52539430191041	7	1.33120328988401
505	72	4	5.59815407899168	5	0.683038916019309
519	76	4	6.49483563576837	4	0.880508212376027
519	112	4	5.16329328124243	5	0.861451327634586
534	95	4	6.40742053761642	6	0.853855802910311
551	123	4	5.06501690470077	5	0.965962898663125
553	74	4	5.50901293039601	9	0.852966995184973
554	91	4	5.59582268090094	9	1.17401094983470
559	97	4	5.44337383508423	9	1.04103018970089
566	101	4	5.08084853240030	5	0.683038916019309
569	104	4	5.38956617780332	10	1.03986373924986
580	119	4	5.27398944314700	6	0.949261377864701
628	180	4	5.59205499974951	5	0.683038916019309
356	78	5	8.67609492972065	4	0.564189583547756
361	127	5	5.90950612548181	9	0.963131863949189
363	62	5	7.09218006078481	4	0.763660723748473
378	105	5	5.36690783493555	10	1.16602036535087
378	111	5	5.85110102915837	4	0.681037072175311
387	50	5	10.1005765522327	8	1.16260300414991
397	140	5	6.03478478637453	6	0.690988298942671
403	86	5	5.52215035842937	7	0.728059254791900
408	127	5	5.85343606631386	5	1.00925300880806
410	66	5	6.72375862051081	4	0.564189583547756
411	135	5	7.81294984926535	8	0.963131863949189
412	165	5	5.15136734258408	4	0.846284375321635
414	124	5	5.33999780447493	4	0.564189583547756
435	134	5	5.82624312810094	9	1.01821429833130
440	188	5	5.20652019320323	4	0.564189583547756

Continues for 1723 rows