Iowa Research Online
The University of Iowa's Institutional Repository

University of Iowa
Iowa Research Online

Theses and Dissertations

Spring 2017

Machine learning improves automated cortical surface reconstruction in human MRI studies

David G. Ellis
University of Iowa

Copyright © 2017 David G. Ellis

This thesis is available at Iowa Research Online: http://ir.uiowa.edu/etd/5465

Recommended Citation
Ellis, David G.. "Machine learning improves automated cortical surface reconstruction in human MRI studies." MS (Master of Science) thesis, University of Iowa, 2017.
http://ir.uiowa.edu/etd/5465.

Follow this and additional works at: http://ir.uiowa.edu/etd


Part of the Biomedical Engineering and Bioengineering Commons

MACHINE LEARNING IMPROVES AUTOMATED CORTICAL SURFACE RECONSTRUCTION IN HUMAN MRI STUDIES

by

David G. Ellis

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

May 2017

Thesis Supervisor: Associate Professor Hans J. Johnson

Copyright by

DAVID G. ELLIS

2017

All Rights Reserved

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

MASTER'S THESIS

This is to certify that the Master's thesis of

David G. Ellis

has been approved by the Examining Committee for the thesis requirement for the Master of Science degree in Biomedical Engineering at the May 2017 graduation.

Thesis Committee: _____
Hans J. Johnson, Thesis Supervisor

_____
Edwin L. Dove

_____
Jatin Vaidya

ACKNOWLEDGEMENTS

To my wife, Harvest, for encouraging me and being an example of how to finish strong. You're simply the best. To Hans J. Johnson and Ipek Oguz for their guidance through the project. To Regina E.Y. Kim and David Welch for taking time out of their days to help answer my many questions. To my friends and family who prayed for me and most importantly to God who enabled me to learn and grow throughout. Thank you.

ABSTRACT

Analysis of surface models reconstructed from human MR images gives researchers the ability to quantify the shape and size of the cerebral cortex. Increasing the reliability of automatic reconstructions would increase the precision and, therefore, power of studies utilizing cortical surface models. We looked at four different workflows for reconstructing cortical surfaces: 1) BAW + LOGIMSOS-B; 2) FreeSurfer + LOGISMOS-B; 3) BAW + FreeSurfer + Machine Learning + LOGISMOS-B; 4) Standard FreeSurfer[4]. Workflows 1-3 were developed in this project. Workflow 1 utilized both BRAINSAutoWorkup(BAW)[16][17][18] and a surface reconstruction tool called LOGISMOS-B[26]. Workflow 2 added LOGISMOS-B to a custom built FreeSurfer workflow that was highly optimized for parallel processing. Workflow 3 combined workflows 1 and 2 and added random forest classifiers for predicting the edges of the cerebral cortex. These predictions were then fed into LOGISMOS-B as the cost function for graph segmentation. To compare these workflows, a dataset of 578 simulated cortical volume changes was created from 20 different sets of MR scans. The workflow utilizing machine learning (workflow 3) produced cortical volume changes with the least amount of error when compared to the known volume changes from the simulations. Machine learning can be effectively used to help reconstruct cortical surfaces that more precisely track changes in the cerebral cortex. This research could be used to increase the power of future projects studying correlations between cortical morphometrics and neurological health.

PUBLIC ABSTRACT

For decades, evaluation of quantified metrics derived from human magnetic resonance imaging (MRI) studies have allowed researchers to advance the understanding of the brain. Software innovation has allowed researchers to reconstruct three dimensional models of the surfaces that separate different types of tissue in the brain. Analysis of these models provides quantifiable features describing the shape and size of the brain. The reliability of these features is crucial for many research studies involving MRI scans of the brain.

The commonly used software for creating the three dimensional models representing the inner and outer edges of the cerebral cortex is FreeSurfer[4]. Although FreeSurfer is state-of-the-art, its reliability leaves much room for improvement. Furthermore, few, if any, alternatives to FreeSurfer exist.

Three methods for reconstructing the surfaces representing the edges of the cerebral cortex were created in this thesis. The methods were tested against FreeSurfer using a set of scans with simulated changes. The method that utilized machine learning to predict the locations of the edges of the cerebral cortex from the MRI scans proved to best detect the simulated changes.

Therefore, the machine learning method detailed in this work proved to be a promising alternative to FreeSurfer for reconstructing the surfaces of the cerebral cortex. Future research involving surface models of the cerebral cortex could be enhanced by using the machine learning method outlined in this thesis.

TABLE OF CONTENTS

LIST OF TABLES vii

LIST OF FIGURES viii

CHAPTER

1 BACKGROUND AND SIGNIFICANCE 1

1.1 Introduction 1
1.2 Cortical Surfaces and Morphometrics 4
1.3 Image Processing Pipelines 5
1.4 Specific Aims 7

2 SIMULATING CORTICAL VOLUME CHANGES 9

2.1 Introduction 9
2.2 Software to Simulate Brain Atrophy and Growth in Human MR Scans 10
2.3 Creating the Simulated Cortical Volume Change Data Set 13
2.4 Simulation Data Set Summary 17
2.5 Using the Simulated Data Set 18

3 LOGISMOS-B PIPELINE 21

3.1 Introduction to LOGISMOS-B 21
3.2 Implementation 21

4 NIPYPE IMPROVES THE PERFORMANCE TIME AND CUSTOMIZABILITY OF FREESURFER'S RECON-ALL 28

4.1 Creating the Nipype Recon-All Pipeline 29
4.2 Test Results and Performance Improvements 33
4.3 Incorporating LOGISMOS-B into Recon-All 35

5 CORTICAL EDGE DETECTION USING MACHINE LEARNING . 36

5.1 Replacing the Gradient Magnitude Cost Function 36
5.2 Ground Truth 37
5.3 Features 37
5.4 Classifier Selection 39

5.5 Classifier Training 39
5.6 Edge Prediction Pipeline 41

6 RESULTS 43

6.1 Evaluating Algorithm Performance 43
6.2 Evaluating the Significance of Differences Between Methods . . . 44

7 CONCLUSIONS 49

7.1 Summary 49
7.2 Discussion 51
7.3 Recommendations for Future Work 53

REFERENCES 56

LIST OF TABLES

Table

3.1 Inputs required by LOGISMOS-B[25]. Note that I did not create any part of LOGISMOS-B. 24

5.1 List of features used for edge classification. 40

6.1 Root mean square error in mm^3 of each of the workflows utilized to measure the simulated volume changes. 44

6.2 P-values of linear mixed effects analysis comparing the means of the volume changes under the general linear hypothesis that the means are the same. The pipeline utilizing machine learning (RF+LOGB) produced volume changes that were significantly different from the other three pipelines. 48

LIST OF FIGURES

Figure

1.1 Illustration showing the layers of the cerebral cortex (i.e. gray matter) and the subcortical tissue (i.e. white matter). The edge between the white matter and gray matter (i.e. white matter surface) is shown in blue while the outer edge of the brain (i.e. gray matter surface) is shown in green. The cortical thickness is defined as the distance between the white matter surface and the gray matter surface. 5

2.1 Example of simulated atrophy on a T1 scan. (Top Left) Original T1 scan with intensity units scaled from 0 to 5000. (Bottom Left) T1 scan with simulated atrophy in the region encompassed by the red box. (Right) Difference between original image and atrophied image. Scale is intensity units. Area shown in red has seen a decrease in pixel intensity. 11

2.2 Correcting the simulation images. After the simulation is performed the modified images have a voxel lattice shift that needs to be corrected. To correct the shift, the entire image is shifted one unit of spacing in all directions. The image is then cropped to the smallest possible image that contains all of the positive values of the label map. The resulting image is a cropped T1 image containing the atrophy/growth in the correct voxel space. The original T1 image that does not contain the atrophy/growth is then used to fill in the information outside the cropped area. 14

2.3 Visualization showing an affected region containing an error due to the atrophy simulation and the same region after correcting the error. The red ovals highlight an area of the skull before and after the correction. Before the correction, the surface of the skull appears to be jagged. After the correction, the jagged part of the skull has been removed. This jagged edge that appeared before the correction is due to the voxels being misaligned. The image on the right does not show the jagged edge because the voxels have been aligned correctly. 15

2.4 Selecting atrophy region location. (Left) The atlas labels are shown from Multi-Atlas Label Fusion (MALF) done as a part of BRAINSAutoWorkup. The label mask of the desired region is created from the MALF label atlas. To find the thickest regions of the label mask the inner label distance is computed using SimpleITK[22]. The skeleton of the label mask is then multiplied with the inner distances. A random pixel above the 95th percentile for inner label distance is selected from the skeleton pixels. This pixel is used as the center for the atrophy region. So that the atrophy region will be proportional to the size of the brain, the radius of the atrophy region is computed to equal 1/1000th of the total brain volume as defined by the MALF labels. (Bottom right) The region that will be affected by the simulated atrophy is shown in green. 17

2.5 Distribution of simulated cortical volume changes. Shown are the histograms of the simulated cortical volume changes with the atrophy simulation rate set to 0.5 (top left), 0.7 (top right), and 1.5 (bottom left). Also shown is the combined distribution of all the simulations. The volume decreases were calculated by taking the cortical volume of the atrophied region from the label map input into the simulator as well as the atrophied label map produced by the simulator. 19

2.6 Method for measuring the cortical volume contained within the atrophy region from white and gray matter surfaces. Given the surfaces (e.g. from FreeSurfer, LOGISMOS-B, etc.) a mask is created defined as the pixels contained within the gray matter surface but not contained within the white matter surface. The resulting mask is the cortical mask (top middle). This mask can be combined with the atrophy region mask to give a mask of the cortical volume contained within the atrophy region. 20

3.1 Illustration of the LOGISMOS-B preprocessing pipeline. The first step in the pipeline is to run BRAINSAutoWorkup. The MALF labels and cerebrospinal fluid(CSF) posterior probabilities are used to remove from the BRAINSABC label map areas that include the hemisphere not being processed and voxels that likely contain CSF. The resulting label map is then used to create a white matter mask and a genus zero white matter surface. After this preprocessing, LOGISMOS-B is run. 23

4.1 Visualization of the cortical thickness projected onto the left hemisphere's inflated surface for both the FreeSurfer(left) and `Nipype`(middle) pipelines as well as the difference of the two thickness maps(right). The thickness measurements produced by the `Nipype` pipeline showed no differences to those produced by FreeSurfer's `recon-all` script. 34

5.1 Example of determining the ground truth from Neuromorphometrics data set. (Left) ADNI 3 Tesla T1w scan and (center) manually edited label maps from the Neuromorphometrics data set corresponding to the T1w scan. (Right) Ground truth masks derived from the label maps used for training the classifier. The gray matter edge is shown in blue, and the white matter edge is shown in green. 38

5.2 The Machine Learning Pipeline. This combines pipeline FreeSurfer and BRAINSAutoWorkup. 42

6.1 The simulated volume change (x-axis) compared to the measured volume changes (y-axis) in the cortex for each pipeline. The black line (y=x) represents the target of the measured volume changes matching the simulated volume changes. 45

6.2 Scatter plot of the absolute error (y-axis) of the pipelines to measure the simulated volume change (x-axis) in the cortex. The black line (y=0) represents the target absolute error of the measured volume changes compared to the simulated volume changes. 46

6.3 Box plot showing the errors of the workflows when measuring the volume changes for the atrophy simulations. 47

6.4 Box plot showing the errors of the workflows when measuring the volume changes for the growth simulations. 47

CHAPTER 1
BACKGROUND AND SIGNIFICANCE

1.1 Introduction

For decades, evaluation of quantified metrics derived from human magnetic resonance imaging (MRI) studies have allowed researchers to advance the understanding of the brain by establishing relationships between brain structure and neurological health. Furthermore, surface-based modeling has offered new possibilities to study distinct features that describe the shape and size of the cerebral cortex. Software innovation has allowed these features to be quantified by analyzing reconstructed surface meshes representing the inner and outer edges of the cerebral cortex. Using software processing tools, researchers are able to segment the MR images and reconstruct surfaces that represent the inner and outer boundaries of the cerebral cortex.

Metrics describing form, also known as morphometrics, give researchers powerful tools to quantify differences and changes in the cortex and find correlations to differences and changes in neurological health. For example, one such morphometric, cortical thickness, has been linked to disease progression in many high profile diseases such as Alzheimers disease[21], Parkinsons disease[42], and Huntingtons disease[30]. Related morphometrics, cortical surface area and cortical folding, have been shown to decrease with age in specific regions of adult brains[10]. Furthermore, in older adults, decreased cortical surface area in specific brain regions correlated to poor per-

formance on tasks designed to assess working memory[24]. A combination of cortical morphometrics and other data in infants between 6 and 12 months of age was used to accurately predict the diagnosis of autism in those high-risk infants at 24 months of age[9].

Many, if not most, researchers analyzing cortical morphometrics rely on automated surface reconstructions from the publicly available tool, FreeSurfer[4][5]. However, authors of a recent validation study examining FreeSurfer's automated cortical thickness measures strongly suggested that the automated results be used with caution[12]. This study acquired two T1-weighted (T1w) structural MRIs from 40 healthy controls[12]. The T1w scans were acquired one week apart[12]. This allowed the researchers to compare the automated FreeSurfer's cortical thickness and volume measurements between the subject's first and second scan[12]. While the correlation of automatic measurements between the two scans was reasonable, 15 out of 40 of the sets of automated cortical surface reconstructions failed visual inspection due to inaccuracies. When the failed sets of scans were excluded from the analysis, the correlation of the test-retest measurements significantly improved[12]. The authors concluded, "FreeSurfer performs well for cortical thickness...but its performance is significantly improved by the addition of visual screening approval, which enhances precision and therefore power."[12] The researchers also mention in this study that, "significant differences observed across imaging sites, between visually approved/disapproved subjects, and across regions with different sizes suggest that these measures should be used with caution."[12]

Therefore, while the state of the art software for automated cortical reconstructions, FreeSurfer, is reasonably accurate, it still requires manual intervention to achieve sufficiently precise measurements[12]. If the reliability of the automatic cortical surface segmentation could be enhanced, this would increase the precision and, therefore, power of studies without the need for manual correction.

Machine learning may present a way to increase the reliability of automatic cortical surface reconstructions. Generally speaking, machine learning is a method for teaching a computer to perform a specific task without explicitly programming how it should accomplish said task. Machine learning has been used in many different applications for automatic medical image segmentation including brain tumor segmentation from MR scans[23]. By feeding into an algorithm a set of brain MR images and the corresponding manual tumor segmentations, a computer algorithm can learn how to reproduce the manual segmentations on its own. This algorithm, known as a classifier, can then be used to identify tumors in new MR images that have no manual segmentations.

In this thesis, machine learning will be utilized to automatically predict locations in a MR image that have a high probability of being located on either the inner or outer edge of the cerebral cortex. A novel cortical surface segmentation algorithm, LOGISMOS-B, will be implemented with and without the use of machine learning. These experimental image processing methods containing LOGISMOS-B will be tested, and the results will be compared against each other and against FreeSurfer. Furthermore, a custom implementation of FreeSurfer will be developed that is well

suited for parallel processing in high performance computing environments.

1.2 Cortical Surfaces and Morphometrics

The two primary surfaces on the cerebral cortex are the surface separating the cerebral cortex from the subcortical tissue and the surface on the outermost edge of cerebral cortex. The surface separating the subcortical tissue and the cerebral cortex is often referred to as the white matter surface. The outer most surface of the cortex corresponds to the pia mater [37] and can be referred to as the pial surface. For the sake of simplicity, the surfaces of the inner and outer edges of the cerebral cortex will be referred to as the white matter surface and the gray matter surface, respectively. Figure 1.1 visualizes these surfaces on a magnetic resonance (MR) scan of a human brain.

Common measures, that is morphometrics, for estimating the form of the cerebral cortex include cortical volume[11], cortical thickness[11], cortical surface area[11], and cortical folding (gyrification)[31]. Cortical volume measures the volume of the brain that contains cortical tissue. Cortical volume can be approximated by estimating the volume that is in-between the white matter and gray matter surfaces. Cortical thickness describes the local thickness of the layers of the cerebral cortex. As illustrated in Figure 1.1, cortical thickness can be quantified as the distance from white matter surface to the gray matter surface. Cortical surface area is defined as either the surface area of the white matter surface or the surface area of the surface midway between the white matter surface and the gray matter surface[37]. Cortical folding,

or gyrification, attempts to quantify the amount of cortex buried within the sulcal folds and compare the amount of buried cortex to the amount of visible cortex[31].

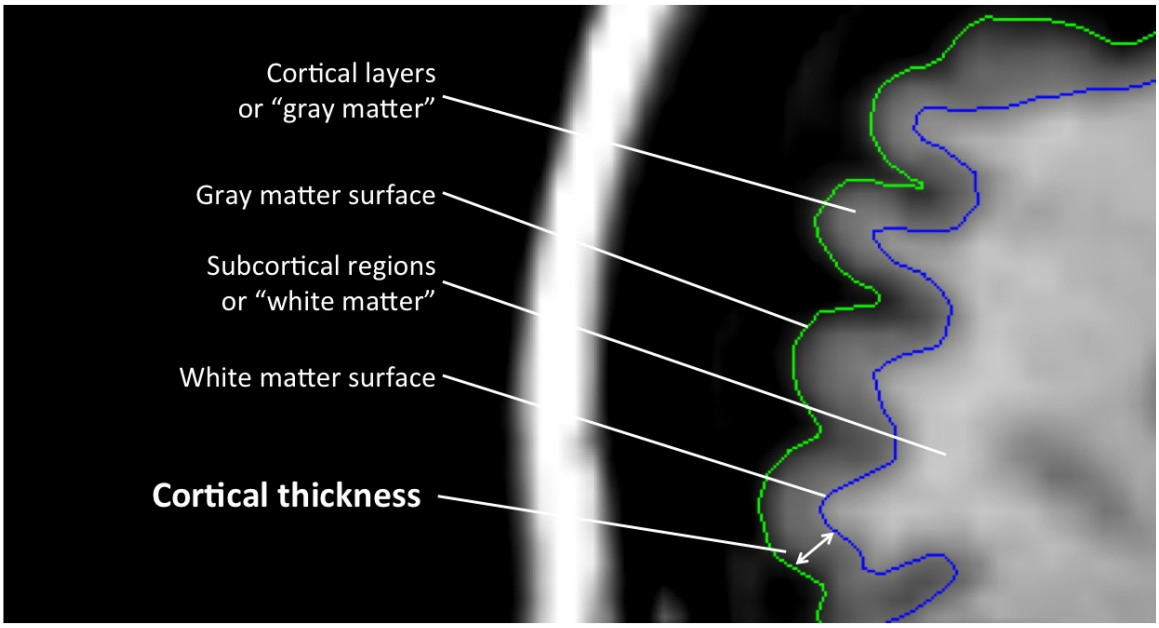

Figure 1.1: Illustration showing the layers of the cerebral cortex (i.e. gray matter) and the subcortical tissue (i.e. white matter). The edge between the white matter and gray matter (i.e. white matter surface) is shown in blue while the outer edge of the brain (i.e. gray matter surface) is shown in green. The cortical thickness is defined as the distance between the white matter surface and the gray matter surface.

1.3 Image Processing Pipelines

A pipeline is a series data processing steps where the outputs of one processing step act as the inputs to other processing steps down the line. Pipelines allows

for many processing steps to be seamlessly interlocked into a single workflow. In order to produce highly efficient pipelines that could be easily used and shared in the neuroimaging community, all of the pipelines for image processing described in this research were created in `Nipype`[7]. `Nipype` is an open source framework designed for nueroimaging interfaces and pipelines. `Nipype` offers improvements over standard shell scripting such as easier rerunning of the pipeline, improved parallel processing, and support for high performance computing. `Nipype` also increases the modularity of the pipeline code. This increased modularity allows for algorithms from other packages (e.g., ANTS[1], FSL[38], BRAINSTools[16][17][18]) to be explored, added into the pipeline, or take the place of existing processing steps. `Nipype` makes rerunning pipelines easier by checking the hash values for a given step in the pipeline before starting that step. Hash values are used to uniquely identify data. If a set of data changes, then the hash value identifier for that data will also change. Therefore, if the hash value for a particular input file to a processing step has not changed since a previous run of the pipeline, then that file is identical and does not need to be reproduced. If the hash values for all of the inputs for a particular processing step have not changed, then `Nipype` will save time by not rerunning that step of the pipeline. Therefore, if a pipeline initially fails, `Nipype` will save time when rerunning the pipeline by only running the parts of the pipeline that have not yet run.

Plugins in `Nipype` allow pipelines to submit jobs to available computing clusters in a high performance computing environment as well as run jobs in parallel on machines with multiple cores. This can dramatically reduce the amount of time

required to complete the processing of a data set by performing steps in the pipeline as soon as all the necessary inputs are collected. This is more efficient than running pipelines in shell scripts that typically only run one command at a time. In conclusion, since the benefits of `Nipype` offer many advantages to traditional shell scripting, all of the pipelines for segmenting the white and gray matter surfaces in this research were created as `Nipype` pipelines and with the goal of making them all available for use and modification by the neuroimaging community.

1.4 Specific Aims

The first aim of this project is to develop a data set that can be used to compare methods for reconstructing cortical surfaces. For this purpose, I explain how I created a data set with simulated volume changes in specific regions of the cerebral cortex, in Chapter 2. Since the volume changes in the cortex are known for this data set, it can be used to measure the effectiveness of methods for detecting cortical volume changes. In this thesis, the cortical volume changes will be calculated from the resulting cortical surface reconstructions. Therefore, methods that segment the cortical surfaces with greater precision will better reflect the known changes in cortical volume.

The second aim of this project is to implement LOGISMOS-B in a robust, automatic data processing pipeline capable of utilizing high performance computing resources. Chapter 3 describes the processing steps of the LOGISMOS-B `Nipype` pipeline that I created.

The third aim of this project is to propose potential improvements to the LOGISMOS-B pipeline. Chapter 4 details the conversion of FreeSurfer's `recon-all` into `Nipype` and it's subsequent integration with the LOGISMOS-B pipeline. Chapter 5 implements machine learning to provide better cortical edge probability initializations that aim to enhance the accuracy of the surfaces segmented by LOGISMOS-B. The artificially atrophied data set will be used to generate metrics for comparing the effectiveness of these changes in the preprocessing methods for LOGISMOS-B.

The final aim of this project is to compare the various preprocessing methods for LOGISMOS-B against each other and against the current state of the art, FreeSurfer. Chapters 6 & 7 describe and discuss the results of running the described methods on the simulated data set.

CHAPTER 2
SIMULATING CORTICAL VOLUME CHANGES

2.1 Introduction

Without quantified benchmarks for assessing the accuracy of cortical surface reconstructions, comparing the effectiveness of these methods is impossible. Researchers have previously attempted to validate image based cortical thickness measurements resulting from cortical surface reconstructions by scanning and autopsying two postmortem brains[29], histological examination on resected brain tissue[3], as well as manual image analysis[19]. However, these methods were performed on a very limited number of subjects and cortical regions. Furthermore, as mentioned by Tustison et al.[35], the methods for assuring the quality of most of these manual measurements is sorely lacking and the measurements themselves could be subject to human bias.

Due to a lack of well-validated ground truth cortical thickness measurements, comparative inferences have been proposed for the validation of image based cortical thickness measurements[35]. This method of validation proposes that due to the high correlation between cortical thickness and phenotypic data such as age and gender, more accurate cortical thickness measurements will be better predictors of such phenotypic data when analyzed on a large number of subjects[35]. While such evaluation of cortical thickness trends may be useful, this method does not validate against a ground truth and requires a large amount of data to be collected, processed,

and analyzed.

In order to validate cortical surface reconstructions, I propose the creation of a set of scans containing known cortical volume changes. By simulating volume changes in the cortical anatomy of brain scans, an artificial ground truth for cortical volume change can be attained. With this artificial ground truth, the precision with which a method accurately detects cortical volume changes can be quantified. This chapter describes how I created a data set with known, albeit artificial, changes in cortical volume.

2.2 Software to Simulate Brain Atrophy and Growth in Human MR Scans

Due to its public availability, growth and atrophy simulations were performed using software provided by Karacali and Davatzikos[15]. This software simulates either growth or atrophy on a set of images in a user-specified region while still preserving the topology outside of that region. The region of on which cortical volume changes are simulated is a sphere defined by a user-specified center and radius. The software then simulates brain tissue atrophy or brain tissue growth in the user-defined sphere shaped region of the image. Figure 2.1 shows an example of a scan before and after the simulation has been applied along with the voxel intensity decrease shown in red. The software also is given a desired rate of atrophy/growth. A rate below 1 tells the software to simulate atrophy while a rate above 1 tells the software to simulate tissue growth. In order to create a set of data with a wide range of simulated volume

changes, I used atrophies rates of 1.5, 0.7, and 0.5.

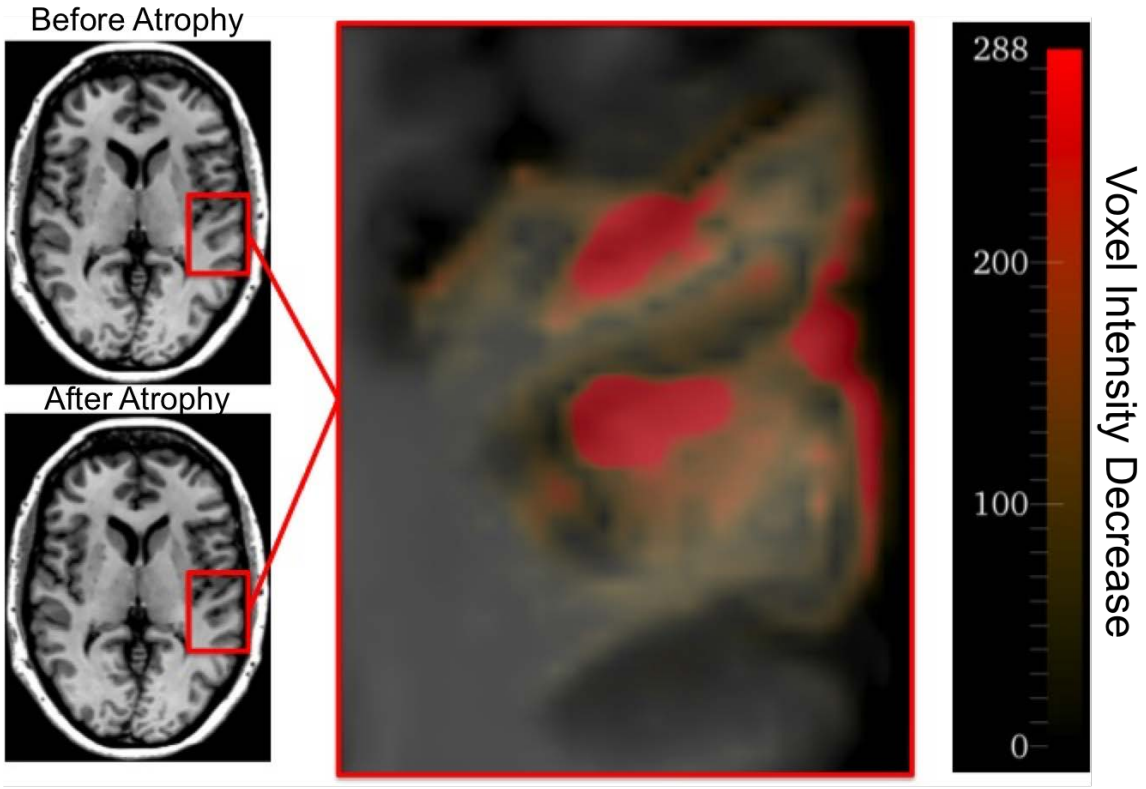

Figure 2.1: Example of simulated atrophy on a T1 scan. (Top Left) Original T1 scan with intensity units scaled from 0 to 5000. (Bottom Left) T1 scan with simulated atrophy in the region encompassed by the red box. (Right) Difference between original image and atrophied image. Scale is intensity units. Area shown in red has seen a decrease in pixel intensity.

Unfortunately, due to the way the atrophy/growth simulator was designed, the simulator shifts certain voxels contained within the scan. This shift causes the

resulting atrophied scans to be misaligned to the original scan. In what presumably is an attempt to save memory usage, the simulator crops the T1 image to the smallest volume that contains all of the labeled values from the label map containing the labels for cerebrospinal fluid (CSF), gray matter(GM) and white matter(WM). The atrophy/growth simulation is then run on the smaller image. The shift occurs when the simulator attempts to put the atrophied version of the smaller image back into the original image. Instead of lining the image up correctly with the original image, the smaller image is shifted by one index in all directions.

Fortunately, the shift that the simulator imposes is consistent. I was able to create a processing pipeline in `Nipype` that performs the atrophy/growth simulation and then automatically corrects the outputs by compensating for the shift. The process used to correct for the shift is visualized in Figure 2.2. The first step used to correct the output image with simulated atrophy/growth was to shift the image up by one voxel in all directions. Next, the shifted atrophied image was cropped using the CSF, GM, WM label map in an identical fashion to the cropping performed by the atrophy simulator. Finally, the original non-atrophied image and the cropped atrophied image were combined so that the voxels outside the boundaries of the cropped image were filled in with the voxel values of the original image. In order to test that the atrophy simulation workflow with the method for correcting the outputs did not unintentionally distort images, the workflow was run with the atrophy rate set to perform no changes to the image (atrophy rate = 1). Under these conditions, the output images after being corrected by the simulation pipeline were identical to

the image input into the pipeline. This indicates that all unintended distortions that were applied to the images had been corrected. Figure 2.3 shows a part of a scan with visual defects before correction by the simulation pipeline.

2.3 Creating the Simulated Cortical Volume Change Data Set

In order to detect a method's ability to track changes in cortical thickness, a new data set was created where simulated changes were introduced into the anatomy of specific areas of a MRI brain scan. In these simulations of brain atrophy and tissue growth, the volume changes in the cortical region are known.

20 scanning sessions were selected from 20 unique subjects. All scanning sessions data came from a 3 Tesla scanner at the University of Iowa and contained both T1w and T2w data. These sessions had previously been processed with BRAINSAutoWorkup [16, 17]. The results were manually evaluated to ensure no obvious errors existed in the automatic segmentation from BRAINSAutoWorkup. To add variation in the simulation location, the 20 subjects were then randomly split into a group of 10 subjects on whose images cortical volume change would be simulated in the left hemisphere and 10 subjects on whose images the simulation would be done in the right hemisphere. Due to its large size and thick cortex, the middle temporal region was selected as the location to simulate atrophy or growth in the left hemisphere. The rostral middle frontal region was selected as the location of atrophy/growth in the right hemisphere, as it had a similar size and thickness when qualitatively compared to the middle temporal region.

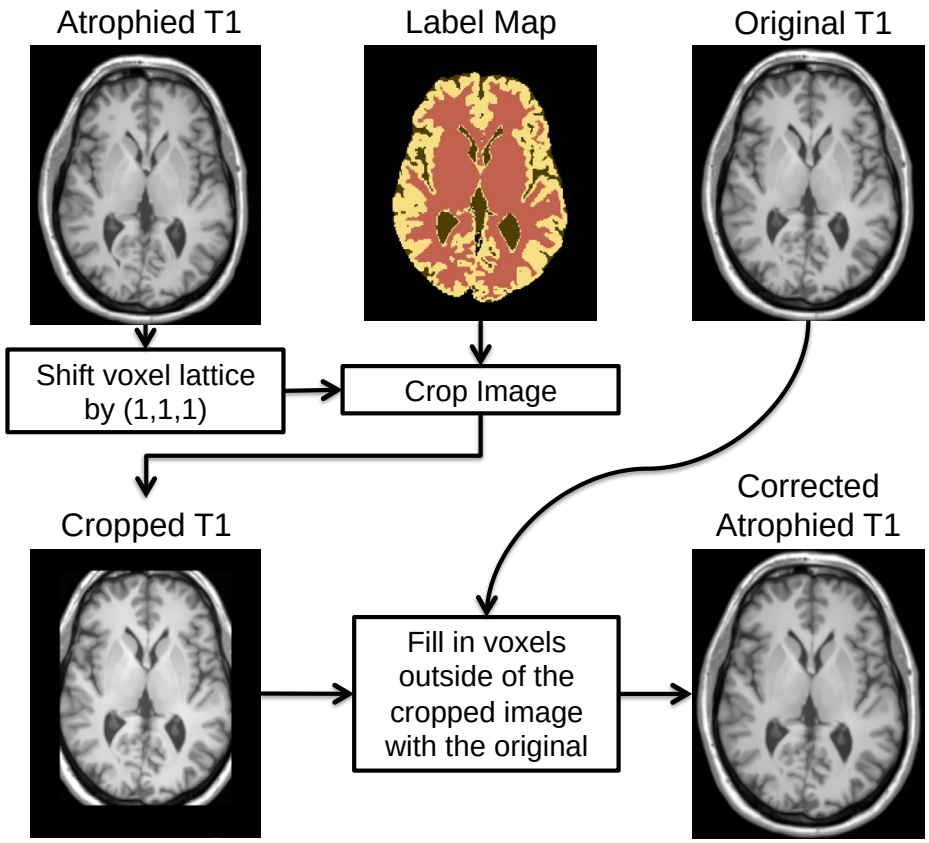

Figure 2.2: Correcting the simulation images. After the simulation is performed the modified images have a voxel lattice shift that needs to be corrected. To correct the shift, the entire image is shifted one unit of spacing in all directions. The image is then cropped to the smallest possible image that contains all of the positive values of the label map. The resulting image is a cropped T1 image containing the atrophy/-growth in the correct voxel space. The original T1 image that does not contain the atrophy/growth is then used to fill in the information outside the cropped area.

The process for selecting the atrophy regions started with a labeled region in the cerebral cortex as defined by Multi-Label Atlas Fusion(MALF)[41][36] which was

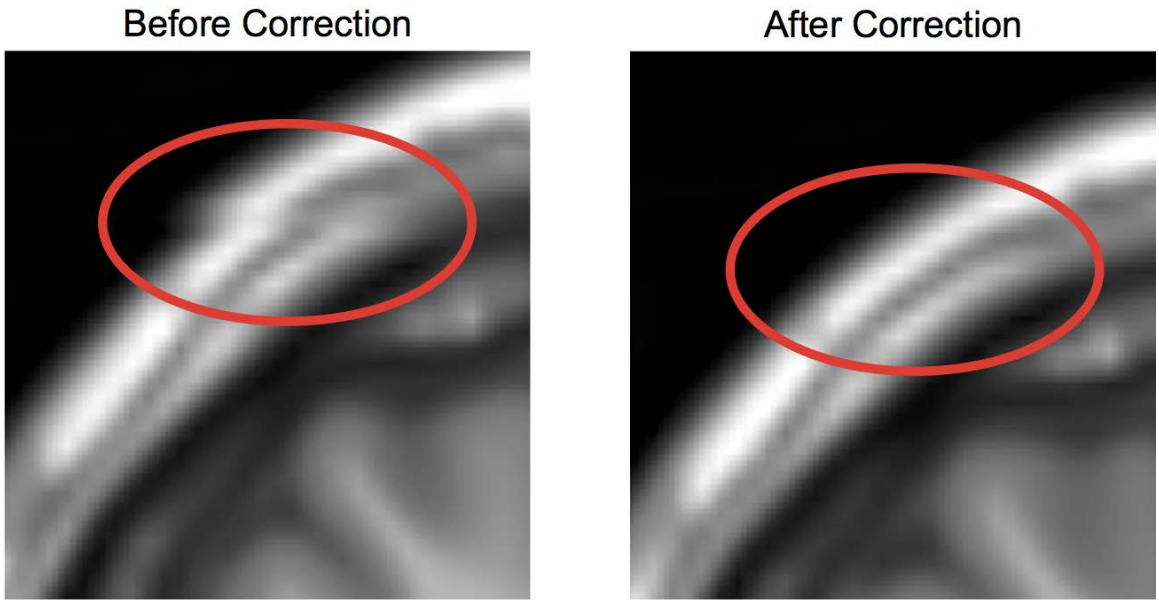

Figure 2.3: Visualization showing an affected region containing an error due to the atrophy simulation and the same region after correcting the error. The red ovals highlight an area of the skull before and after the correction. Before the correction, the surface of the skull appears to be jagged. After the correction, the jagged part of the skull has been removed. This jagged edge that appeared before the correction is due to the voxels being misaligned. The image on the right does not show the jagged edge because the voxels have been aligned correctly.

performed as a part of BRAINSAutoWorkup. As mentioned earlier, the left middle temporal region and the right rostral middle frontal were chosen as the general locations in which to run the atrophy/growth simulations. In order to randomly choose a region in these general locations an automated process was employed. Figure 2.4 visualizes the automated process used to select the regions for growth/atrophy sim-

ulations. To avoid performing atrophy or growth in areas that did not contain much cortical tissue, only locations within thick areas of the cortex were selected. This was accomplished by calculating the inner label distance of the labeled cortical region as defined by MALF. The voxels(3 dimensional pixels) with the highest inner label distance correspond to thick cortical locations within the labeled region. The voxels that lay along the skeleton of the labeled region were then selected as candidate atrophy centers. So that only the thickest parts of the labeled region were selected, a threshold was applied so that only the voxels that were above the 95th percentile for inner label distance remained. The voxels were then selected randomly from those that remained. The selected voxels were then used as the centers for the atrophy/growth simulation.

In order to have atrophy/growth regions that were consistent in size relative to the brain, the radius of atrophy was selected so that the atrophy region would be approximately one one-thousandth of the total brain volume. This size was chosen based on qualitative analysis of atrophy regions of various sizes. The atrophy regions approximately one one-thousandth the size of the brain appeared to produce noticeable amounts of atrophy without significantly disturbing the topology of the brain. The process of selecting a region center and radius was repeated on each scan set until the desired number of centers and radii had been obtained. These centers and radii were then input into the simulation pipeline to perform the growth/atrophy simulations in those regions.

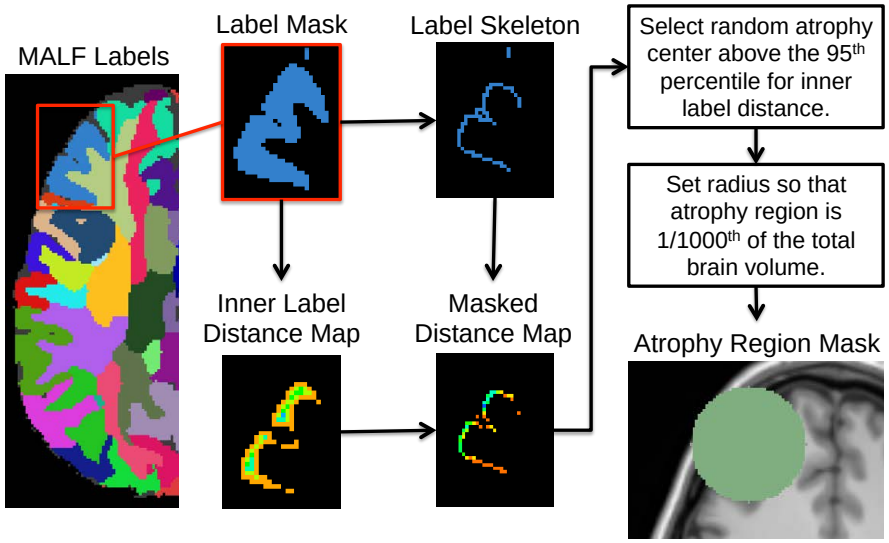

Figure 2.4: Selecting atrophy region location. (Left) The atlas labels are shown from Multi-Atlas Label Fusion (MALF) done as a part of BRAINSAutoWorkup. The label mask of the desired region is created from the MALF label atlas. To find the thickest regions of the label mask the inner label distance is computed using SimpleITK[22]. The skeleton of the label mask is then multiplied with the inner distances. A random pixel above the 95th percentile for inner label distance is selected from the skeleton pixels. This pixel is used as the center for the atrophy region. So that the atrophy region will be proportional to the size of the brain, the radius of the atrophy region is computed to equal 1/1000th of the total brain volume as defined by the MALF labels. (Bottom right) The region that will be affected by the simulated atrophy is shown in green.

2.4 Simulation Data Set Summary

In total, 400 atrophy simulations and 200 growth simulations were performed. 200 simulations were run with each of the atrophy rates of 0.5, 0.7, and 1.5. 300 of

the simulations were performed in the location of left middle temporal region and the other 300 in the right rostral middle frontal region. 22 of the atrophy simulations were duplicates, having the same atrophy rate at the same location, and were not included in the final data set. Therefore, the final a data set contained 578 unique simulations. Figure 2.5 shows the resulting distribution of simulated cortical volume in the data set.

2.5 Using the Simulated Data Set

With this set of atrophy simulations and their known changes in cortical tissue volume within the region of atrophy, different methods of detecting the white and gray matter surfaces can be compared.

Since the volume change is known, the method for detecting surfaces that has volume changes closest to the known volume change will be the more precise method for measuring cortical volume changes. Figure 2.6 shows how the cortical volume within the specified atrophy region was calculated from the white and gray matter surfaces. The comparisons were done by converting the output gray matter and white matter mesh surfaces to a mask files representing the volume inside each surface. The cortical volume mask was found by taking the volume inside the gray matter surface that was not inside the white matter surface. The cortical volume contained within the atrophied region was then calculated.

The cortical volume inside the atrophy regions was measured with each method for detecting surfaces for all atrophy simulations both at baseline and with the sim-

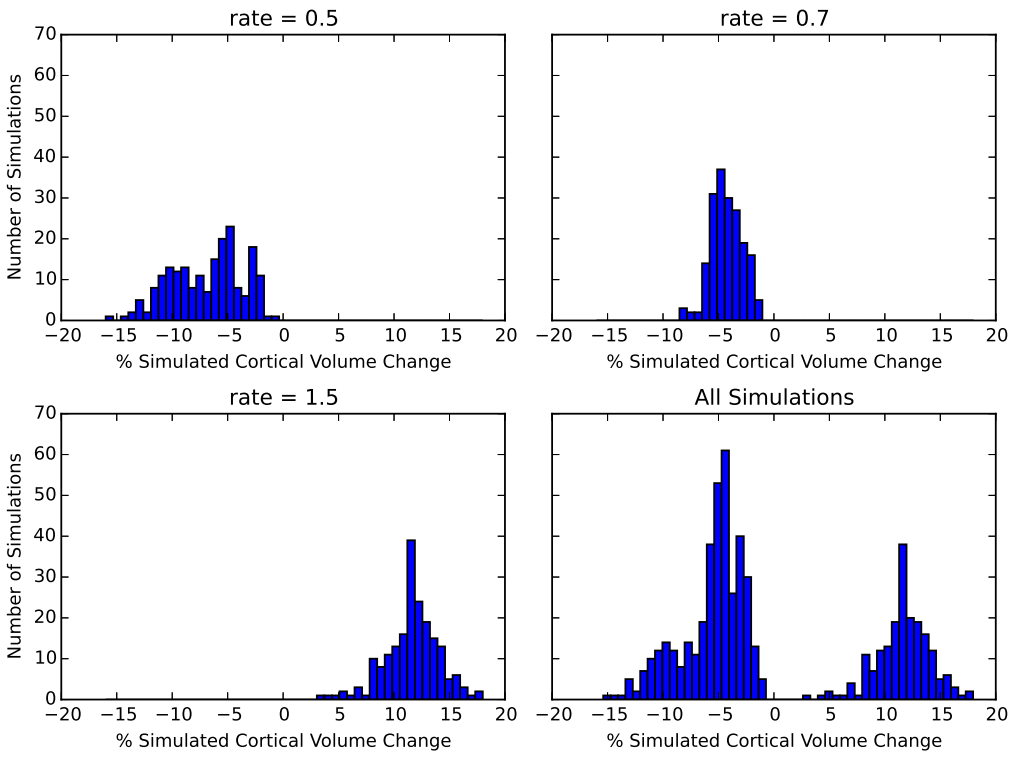

Figure 2.5: Distribution of simulated cortical volume changes. Shown are the histograms of the simulated cortical volume changes with the atrophy simulation rate set to 0.5 (top left), 0.7 (top right), and 1.5 (bottom left). Also shown is the combined distribution of all the simulations. The volume decreases were calculated by taking the cortical volume of the atrophied region from the label map input into the simulator as well as the atrophied label map produced by the simulator.

ulated atrophy. The change in volume was then compared to the known change in cortical volume from the atrophy simulator.

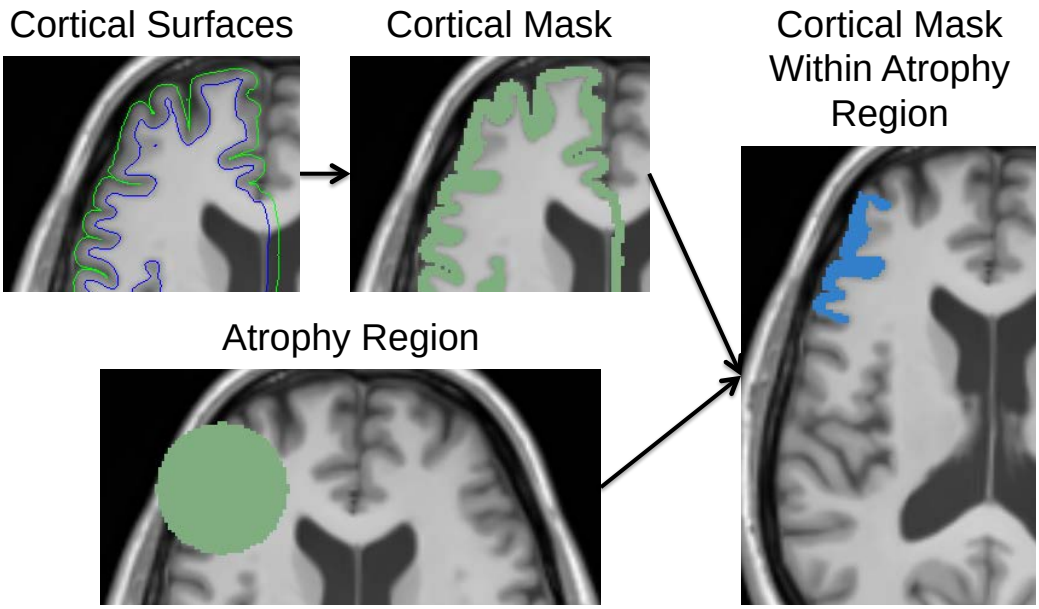

Figure 2.6: Method for measuring the cortical volume contained within the atrophy region from white and gray matter surfaces. Given the surfaces (e.g. from FreeSurfer, LOGISMOS-B, etc.) a mask is created defined as the pixels contained within the gray matter surface but not contained within the white matter surface. The resulting mask is the cortical mask (top middle). This mask can be combined with the atrophy region mask to give a mask of the cortical volume contained within the atrophy region.

CHAPTER 3
LOGISMOS-B PIPELINE

3.1 Introduction to LOGISMOS-B

One approach to improving the accuracy of cortical surface reconstructions is to utilize a different surface segmentation framework. The LOGISMOS[14] graph segmentation framework is promising, as it has been shown to accurately segment multiple surfaces simultaneously in various medical imaging modalities and applications [33][40][43]. Developments by Oguz et. al.[26] have utilized the LOGISMOS framework to segment the white and gray matter surfaces of the brain from anatomical MR images. Oguz et. al.[26] have demonstrated that the Layered Optimal Graph Image Segmentation of Multiple Objects and Surfaces for the Brain (LOGISMOS-B) may have promise for improving upon the current state of the art reconstructions.

3.2 Implementation

I implemented LOGISMOS-B in a robust pipeline for reconstructing the cortical surfaces. This implementation allowed for a comparison to FreeSurfer's surface segmentation. Furthermore, the creation of the pipeline in `Nipype` allowed for easy modification and comparison between preprocessing methods.

As LOGISMOS-B requires prior information in addition to the T1w and T2w scans, the preprocessing methods used have a substantial impact on the final results. Table 3.1 describes the inputs required by LOGISMOS-B and Figure 3.1 gives a broad overview of the LOGISMOS-B preprocessing pipeline that I developed as a part of

this thesis.

The first step of the LOGISMOS-B pipeline was to normalize and segment the input T1w and T2w images using BRAINSAutoWorkup(BAW) [16][17][18]. BRAINSAutoWorkup outputs the following images that were used for later steps in the LOGISMOS-B pipeline:

- Averaged and normalized T1w and T2w scans aligned in ACPC space.
- Labeled atlas from a 42 year old male (i.e. HNCMA atlas) [8] aligned to the T1w and T2w images.
- Posterior probabilities for the cerebrospinal fluid (CSF). These probabilities range from 0 (low probability) to 1 (high probability).
- Label atlas produced by multi-atlas label fusion (MALF)[16].
- Atlas of brain labels as defined by BRAINSABC.

A crucial step in LOGISMOS-B pipeline is the generation of the label map produced by BRAINSABC. LOGISMOS-B uses the BRAINSABC label map to determine the regions that LOGISMOS-B considers to be inside the brain or outside the brain. To void excluding brain tissue that may have been inaccurately excluded from the label map, LOGISMOS-B dilates the label map upon execution. Only voxel locations inside the dilated label map are considered for the graph segmentation by LOGISMOS-B.

In practice, it was observed that with the default label map output from BRAINSABC the segmentation produced by LOGISMOS-B would sometimes place the gray matter surface in the middle of area that is likely to be CSF. In such cir-

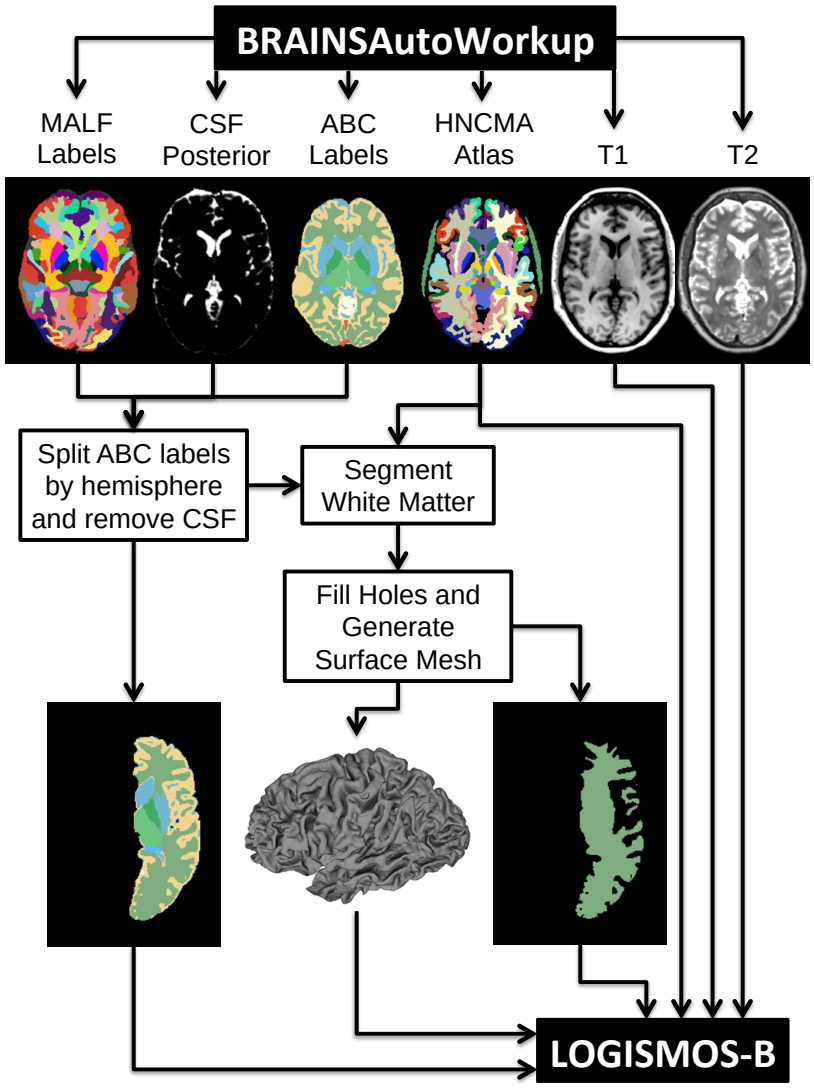

Figure 3.1: Illustration of the LOGISMOS-B preprocessing pipeline. The first step in the pipeline is to run BRAINSAutoWorkup. The MALF labels and cerebrospinal fluid(CSF) posterior probabilities are used to remove from the BRAINSABC label map areas that include the hemisphere not being processed and voxels that likely contain CSF. The resulting label map is then used to create a white matter mask and a genus zero white matter surface. After this preprocessing, LOGISMOS-B is run.

Table 3.1: Inputs required by LOGISMOS-B[25]. Note that I did not create any part of LOGISMOS-B.

T1w and T2w scans	LOGISMOS-B uses gradient vector field of these scans to create the graph. Furthermore, LOGISMOS-B uses a combination of the gradient magnitudes of these scans as a cost function for the graph.
BRAINSABC label map	LOGISMOS-B uses the label map provided by BRAINSABC to determine the areas of the graph that are outside the brain.
White matter mask	A mask containing the white matter area that is used for creation of the graph.
White matter surface	An initial estimate for the white matter surface.
HNCMA labels	The HNCMA atlas[8] morphed to the subject space. LOGISMOS-B uses these labels to segment with different parameters the regions of the cortex that should be thicker or thinner.

cumstances, since LOGISMOS-B solves for both surfaces simultaneously, the white matter surface would be inaccurate as well.

To remedy this problem of LOGISMOS-B misplacing the cortical surfaces, the pipeline I created modifies the BRAINSABC label map to only include the cortical and subcortical voxels for the hemisphere being processed. When the left hemisphere is being processed, the pipeline only retains the voxels belonging to the left hemisphere in the label map, and the same is true for the right hemisphere. Automatically segmenting the hemispheres in the brain is complicated due to the fact the that the brain hemispheres are asymmetrical [34]. Despite the asymmetry, the label map produced by multi-atlas label fusion (MALF) appeared to provide robust and accurate segmentation of the hemispheres.

The LOGISMOS-B pipeline combines the MALF label regions belonging to the hemisphere being processed into a single hemisphere mask. First, a dictionary was created of all the label names and the corresponding integer that represents that label in the label map image. Next, a mask was created of all the labels in the left hemisphere and another mask for all the labels in the right hemisphere. These masks were then applied to the ABC labels to generate a label map for the left hemisphere and a separate label map for the right hemisphere. The CSF voxels, the third and fourth ventricles, the brain stem, and the cerebellum as defined by the MALF label map are all removed from the hemisphere mask. In order to keep LOGISMOS-B from reconstructing the gray matter surface in an area that is likely to be CSF, the pipeline removed from the label map any voxel with a posterior CSF probability

greater than 90%. Furthermore, to avoid having holes in the subcortical regions, the lateral ventricles were preserved and not removed along with the rest of the CSF.

Qualitatively, the exclusion of non-cortical and non-subcortical regions along with the voxels from the opposite hemisphere appeared to improve the surface reconstructions resulting from the pipeline. However, even with CSF voxels removed, it was observed that LOGISMOS-B would still place the gray matter surface in a CSF region. This is due to the fact that the BRAINSABC label map is dilated upon the execution of LOGISMOS-B. Therefore, modifying the BRAINSABC label map was not enough to prevent LOGISMOS-B from making obvious errors when reconstructing the gray matter surface.

LOGISMOS-B also requires a white matter mask as well as an initial white matter surface. I observed that LOGISMOS-B appeared to produce more accurate results when given a white matter segmentation that erred on the side of falsely labelling gray matter as white matter (false positive) rather than incorrectly classifying white matter as a different tissue type (false negative). In an early version of the LOGISMOS-B pipeline, the MALF label map was used to generate the white matter segmentation. This segmentation appeared to be relatively accurate but erred on the side of being an under ambitious segmentation that left out small areas of white matter. LOGISMOS-B was not able to recover the missing white matter segments during execution. Therefore, the latest version of the pipeline uses the BRAINSABC label map to define the white matter segmentation. This label map tends to include parts of the gray matter in in its segmentation. However, the final reconstructions output

from LOGISMOS-B appeared to adjust and correct the areas where gray matter was inaccurately labeled as white matter in the white matter mask.

After the white matter mask was created, the pipeline used the GenusZeroImageFilter (included in BRAINSTools) on the mask to fill in any holes and handles. Then, the pipeline generated a smooth initial estimate of the white matter surface by converting the white matter mask into a surface with BRAINSSurfaceGeneration (also included in BRAINSTools).

The final step in the pipeline was to use LOGISMOS-B to reconstruct the gray and white matter surfaces for each hemisphere. LOGISMOS-B takes as inputs the filled white matter mask, the white matter surface mesh, and the brain labels the hemisphere being processed. LOGISMOS-B also uses the warped HNCMA to set graph constraints differently depending on the labeled region. To set the graph node costs, LOGISMOS-B uses a combination of gradient magnitudes from the T1 and T2 images. Using the gradient magnitude as a cost function allows for the graph segmentation to detect the edges of features in the image.

CHAPTER 4
NIPYPE IMPROVES THE PERFORMANCE TIME AND CUSTOMIZABILITY OF FREESURFER'S RECON-ALL

A prominent tool for processing scans and reconstructing the cortical surfaces is FreeSurfer's `recon-all` script. FreeSurfer's `recon-all` script sequentially calls approximately 170 commands on FreeSurfer executables. The `recon-all` script's specific combination of commands take the input MR head images and process them to produce numerous meaningful neuromorphological measurements, label maps, and surfaces. Among the output from this script are the white and gray matter surfaces that are then used to generate cortical thickness measurements and other cortical morphometrics.

Due to the large number of commands that are run by `recon-all`, the processing time for a single MR session is rather large. When tested on a 16 core CentOS machine with 64 GB of memory, the processing time for the `recon-all` script took nearly 9 hours. Converting the `recon-all` script into a `Nipype` pipeline could reduce the amount of processing time required by effectively using available computing resources. `Nipype` pipelines are able to run commands simultaneously on multiple computing cores. This feature makes `Nipype` pipelines more efficient than scripts when run on computers with multiple cores and in high performance computing environments.

In addition to quicker processing times, converting the `recon-all` shell script to a `Nipype` pipeline would allow for easy modification and experimentation. The

Nipype pipeline could be easily modified to create a hybrid pipeline that incorporates both FreeSurfer and LOGISMOS-B to segment the cortical surfaces. For more details on the advantages of Nipype pipelines see Section 1.3.

4.1 Creating the Nipype Recon-All Pipeline

The first step to creating the Nipype equivalent of the recon-all script was to determine what commands the script runs and in what order. This information was ascertained by running the recon-all script on a set of test scans and then inspecting the recon-all.cmd output log. This log lists the commands that were run by the script and the time that they were run.

Once the list of commands had been obtained, the next step was to wrap each of the necessary FreeSurfer executables in a Nipype interface. The Nipype interface allows command line executables to be called in Python. Each interface calls a specific executable and translates Python coded inputs to inputs on the command line.

In order to wrap an executable in a Nipype interface, it is necessary to know the executable input files, input parameters, and output files. Sometimes the input files, parameters, and output files are all listed on the command line or in the FreeSurfer documentation. However, many of the FreeSurfer executables assume the location of input files. For example, assume that ImplicitExampleEXE is an executable that is run on the command line. The user runs the executable as follows:

```
bash$ ImplicitExampleEXE
'some example text'
bash$
```

From the documentation we might know that what ImplicitExampleEXE

prints to the screen is the contents of a file. However, this file is not specified on the command line. Instead, the file location is hard coded within the program. As with many FreeSurfer executables, the hard coded location of the file may not be given in the documentation. Figuring out what file the executable is calling requires some investigating. FreeSurfer commands often make use of a custom environmental variable `SUBJECTS_DIR` and then read in contained at specific locations with in the directory specified by `SUBJECTS_DIR`. We can purposefully cause the executable to fail by setting `SUBJECTS_DIR` to an empty directory.

```
bash$ export SUBJECTS_DIR=/path/to/empty/dir
bash$ ImplicitExampleEXE
IOERROR: Failed to read '/path/to/empty/dir/ExampleFile.txt'
bash$
```

When the `ImplicitExampleEXE` failed, the error message printed the location of the input file that it was trying to read. Now we know that the executable is reading the file `$SUBJECTS_DIR/ExampleFile.txt`. By purposefully causing the executables to fail in this way, we were able to determine the location of the hard coded input files for all the executables used in the `recon-all` script.

However, when the input files are hardcoded into the executable rather than explicitly defined by the user on the command line at execution, there is no way to tell the executable to read in a user-specified file from different location in the file system. `Nipype` interfaces depend on being able to dictate the input and output files for command line executables. This control allows over the input and output files allows `Nipype` to link together interfaces into fully automatic processing pipelines. Without this control, `Nipype` would not be able to run processing steps in parallel as

commands could potentially modify shared directories in unpredictable ways.

The lack of control over the hardcoded input files for FreeSurfer executables was solved by creating an empty directory, modifying the `SUBJECTS_DIR` environmental variable to point to that empty directory, and then by copying the necessary input files into their respective hardcoded locations within the directory. In the case of `ImplicitExampleEXE` we know that the executable reads in the file `$SUBJECTS_DIR\ExampleFile.txt`. However, we want to create an interface that allows us to use `ImplicitExampleEXE` to read in any file specified by the user. In the `Nipype` interfaces for `ImplicitExampleEXE` we can create an input called 'inputFile'. Next we can write code to have the interface make an empty directory, change `SUBJECTS_DIR` to point to that directory, and then copy the file specified by 'inputFile' to the location `$SUBJECTS_DIR\ExampleFile.txt`. Every time the interface is called, it will run this code prior to running the executable. So, if we want `ImplicitExampleEXE` to read `SomeOtherFile.txt` we can pass `SomeOtherFile.txt` as the input to the interface. The interface will then copy that file in the method above and `ImplicitExampleEXE` will read `SomeOtherFile.txt`. In this fashion, `Nipype` interfaces were created for all of the FreeSurfer executables used in the FreeSurfer's `recon-all` script. Furthermore, documentation, examples, and tests were written for each of the interfaces. The interfaces and code were then made freely available as part of the `Nipype` source code at `https://github.com/nipy/nipype/tree/master/nipype/interfaces/freesurfer`.

With the interfaces available, the `Nipype` pipeline that replicates the process-

ing steps of FreeSurfer's `recon-all` was assembled by carefully connecting the input and output files of the interfaces together in the proper order. Most of the information needed to connect the interfaces in the pipeline could be ascertained from the `recon-all.cmd` output log or FreeSurfer's helpful, yet non-exhaustive, developer's table `https://surfer.nmr.mgh.harvard.edu/fswiki/ReconAllDevTable`. The information that could not be found was inferred by testing different configurations of interfaces in `Nipype` against the outputs from the `recon-all` script. Separate `Nipype` pipelines were created for AutoRecon1, AutoRecon2, and AutoRecon3, the three main processing steps for FreeSurfer's `recon-all` script. These pipelines were connected together to create the high-level `Nipype` pipeline that replicates the processing steps of FreeSurfer's `recon-all` script. Furthermore, an output node was created with the all files produced by the pipeline. This allows for additional steps to be seamlessly added onto the end of the pipeline. After processing, the output files are copied to a user-specified location with traditional FreeSurfer result layout. The pipeline is designed to detect the version of FreeSurfer being used and run either 5.3.0 and 6.0beta versions of the `recon-all` script. The pipelines was also integrated into the source code for `Nipype` and is freely available at `https://github.com/nipy/nipype/tree/master/nipype/workflows/smri/freesurfer`.

The `Nipype` pipeline can be easily run in Python with `Nipype` release 0.12.1 or later:

```
from nipype.workflows.smri.freesurfer import
   create_reconall_workflow
recon_all = create_reconall_workflow()
recon_all.inputs.inputspec.subjects_dir = "/path/to/
   example_directory"
```

```
recon_all.inputs.inputspec.T1_files = "t1_file.nii"
recon_all.inputs.inputspec.subject_id = "example"
recon_all.run()
```

The above code will run the `Nipype` pipeline on the specified T1w file and output processed files into the directory specified by input "subjects_dir" variable.

Finally, a tutorial was created to demonstrate how to use the pipeline. This tutorial was added to the `Nipype` source code as well at `https://github.com/nipy/nipype/blob/master/examples/smri_fsreconall.py`.

4.2 Test Results and Performance Improvements

A thorough comparison between the newly created pipeline in `Nipype` and the `recon-all` script proved that the `Nipype` pipeline produced identical images and surfaces to those of FreeSurfer's `recon-all` script. Both the pipeline and the script were run on identical sets of inputs and a folder was created with the set of outputs from each. All the images from both output folders were converted into NIfTI format[20] and the surfaces were converted into VTK's ASCII format[32]. Next each of the converted output images and surfaces from FreeSurfer's `recon-all` script was compared to the identically named file in the `Nipype` pipeline's output folder. The images were compared by loading each image into Python using SimpleITK[22] and testing if the SimpleITK hash values of each image were identical. Identical hash values between two images indicate that the information contained in those images is identical. The surfaces were compared using the 'diff' tool that can be found on UNIX operating systems. The 'diff' tool prints to the screen text differences in files.

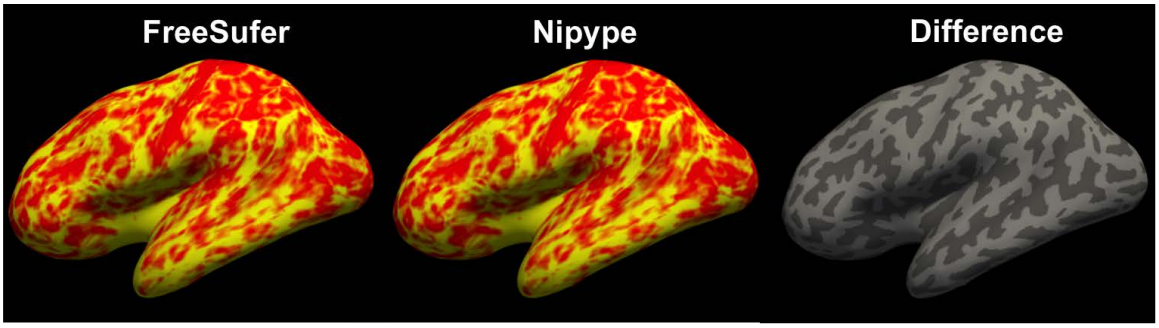

Figure 4.1: Visualization of the cortical thickness projected onto the left hemisphere's inflated surface for both the FreeSurfer(left) and `Nipype`(middle) pipelines as well as the difference of the two thickness maps(right). The thickness measurements produced by the `Nipype` pipeline showed no differences to those produced by FreeSurfer's `recon-all` script.

If 'diff' does not print any differences to the screen, then the text files are identical. 'diff' was able to compare the VTK files because they were stored in ASCII format. If 'diff' does not print any differneces to the screen, then the two VTK files must contain identical information. Using these tests, all of the images and surfaces output by the `Nipype` pipeline and the FreeSurfer script were identical.

The cortical thickness measurements were also compared between the `Nipype` pipeline and FreeSurfer's `recon-all` script. The results were visualized as can be seen in Figure 4.1. The thickness measurements produced by the `Nipype` pipeline showed no differences to those produced by FreeSurfer's `recon-all` script.

The pipeline running times were compared by running FreeSurfer's `recon-all` and the equivalent `Nipype` pipeline. The `Nipype` pipeline was run using multiprocess-

ing to utilize multiple cores on the machine. Though both pipelines allow for multi-threading of certain pipeline steps using OpenMP, this feature was not utilized in this test. Under these conditions, FreeSurfer's `recon-all` pipeline completed processing in over 8.9 hours. The `Nipype` pipeline completed processing in under 6 hours.

The near 3 hour performance enhancement is likely due to the fact that `Nipype` runs in parallel as many commands as possible while FreeSurfer runs commands sequentially.

4.3 Incorporating LOGISMOS-B into Recon-All

After the creation of the `Nipype recon-all` pipeline, it was then possible to create a FreeSurfer modified pipeline that incorporated both LOGISMOS-B and FreeSurfer. This allowed for better comparisons between the pipelines. The white surface file was used as the initial white surface file for LOGISMOS-B after it had been converted to the VTK format in the subject space using `mris_convert`. This surface was made into a white matter mask file by using `mri_surfacemask`. The segmentation file used by FreeSurfer when segmenting the surfaces was recoded so that the label numbers matched those used by BRAINSAutoWorkup and LOGISMOS-B. Another segmentation file from FreeSurfer with the right and left hemispheres defined was used to split the recoded label map into separate hemispheres. Finally, the unmodified averaged T1 file from FreeSurfer was used as the input T1 to LOGISMOS-B.

CHAPTER 5
CORTICAL EDGE DETECTION USING MACHINE LEARNING

5.1 Replacing the Gradient Magnitude Cost Function

LOGISMOS-B uses the gradient magnitude image to define the node costs for its graph optimization. These node costs are inversely related to the gradient magnitude. This allows the algorithm to find areas in the image with sharp changes in pixel intensity. Pixels with a large gradient magnitude are more likely to be edge pixels than those with a small gradient magnitude. Therefore, by assigning the inverse gradient magnitude as the cost function, the lowest cost path will lie along pixels with large gradient magnitudes. This is how LOGISMOS-B detects the white and gray matter surfaces from the MR scans.

Using the inverse of the gradient magnitude as the cost function works well for most applications. However, it causes problems in areas where no hard edge can easily be detected. When LOGISMOS-B gives inaccurate results, it is often because the algorithm is detecting the wrong edge or failing to detect the white or gray matter edge. For example, in many cases LOGISMOS-B will place the white matter edge close to what is truly outer gray matter edge and will place the gray matter edge somewhere in the CSF. Further complicating the use of the gradient magnitude as the cost function is that nonzero gradient values are often reported throughout the gray matter as it is not homogeneous. This makes detecting a single white matter and gray matter edge difficult. Having a separate cost function for both the gray matter

edge and the white matter edge would increase the specificity of the cost function and prevent LOGISMOS-B from detecting the wrong edge.

In order to provide a more specific cost function for the white matter and gray matter edges, machine learning techniques can be applied. By training classifiers that can predict the white and gray matters edge pixels a set of costs for the white matter edges and gray matter edges based on the predicted probability from the classifier.

5.2 Ground Truth

In order to train classifiers to predict the white and gray matter edges, a set of ground truth white and gray matter edges are needed. The Neuromorphometrics data set provides a set of manually traced label maps using ADNI T1w scans[13][39]. By finding the contour from the label map between the white matter and non-white matter labels, the white matter edge pixels were masked. The gray matter edge pixels were masked by finding the contour pixels between the gray matter and the CSF/background pixels. Figure 5.1 shows the gray matter edge mask (blue) and the white matter edge mask (green) along with the original T1w and manual segmentation images.

5.3 Features

Before training the edge classifiers appropriate features that describe the edges were selected and normalized. The machine learning workflow utilized BRAINSAutoWorkup to provide normalized T1w intensity image for each session.

From the normalized T1w image, various image features were computed for

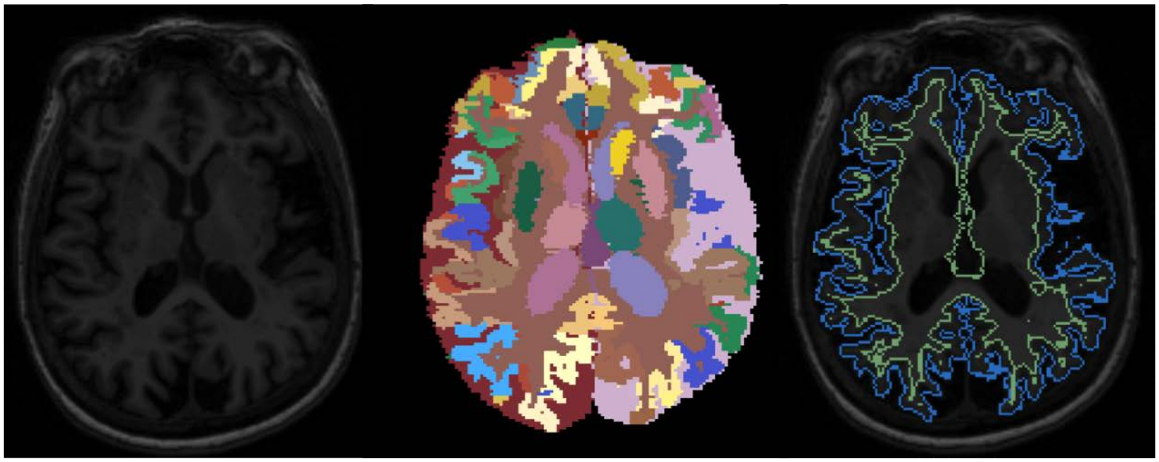

Figure 5.1: Example of determining the ground truth from Neuromorphometrics data set. (Left) ADNI 3 Tesla T1w scan and (center) manually edited label maps from the Neuromorphometrics data set corresponding to the T1w scan. (Right) Ground truth masks derived from the label maps used for training the classifier. The gray matter edge is shown in blue, and the white matter edge is shown in green.

use in training the edge classifier. Included in these features were edge measures such as the gradient magnitude, second order gradient magnitude, and others. In order to provide information on shapes of varying size, a set of features were also calculated with a set of different size Gaussian smoothing kernels. Gaussian smoothing kernels were used with sigma values of 0.5, 1.0, 1.5, 2.0. To provide a higher order features that help describe the shapes in an image, the eigenvalues of the Hessian matrix were used. In order to detect both light and dark shape features, the eigenvalues were sorted by absolute value. Previous research has shown that the sorted eigenvalues can be used to detect blob, plate, and vessel shapes [6].

BRAINSAutoWorkup also provided posterior probabilities for tissue classification. These probabilities were added to the feature vector to allow the classifier to leverage classification data from BRAINSAutoWorkup. For features that encode pixel location with respect to the center of the image, the rho, phi, and theta spherical coordinate images were added to the feature list. In order to normalize the data, all of the features images were sampled with a spacing of 1mm x 1mm x 1mm. The original manually traced label maps were also resampled using nearest neighbor so that their spacing matched that of the feature images. See Table 5.1 for a complete list of the features used for training and classification.

5.4 Classifier Selection

A random forest classifier was selected, as it provided an efficient parametrized classifier that does not store the training data and can produce predictions with a probability percentage. A parameterized classifier was necessary as the memory consumption of storing the features is very high. Therefore, non-parameterized classifiers, such as nearest-neighbor, were not considered. The probability percentage output by the random forest classifier was used as the inverse of the node cost for LOGISMOS-B.

5.5 Classifier Training

The classifier was trained using the Neuromorphometrics data set described in Section 5.2. Only the 3 Tesla scans were used from this data set as they provided higher quality images. The T1w images were processed using BRAINSAutoWorkup and the features described in Section 5.3 were combined into a single data table along

Table 5.1: List of features used for edge classification.

	T1w Features without Smoothing
1	T1w Pixel Intensity
2	T1w Gradient Magnitude
3	T1w Second Order Gradient Magnitude
4	T1w with Sobel Filter
5 - 7	T1w Eigenvalues of Hessian Matrix
8	T1w with Laplacian Filter
	T1w Features with Recursive Gaussian Smoothing (sigma = 0.5, 1.0, 1.5, 2.0)
9 - 20	Smoothed T1w Eigenvalues of Hessian Matrix
21 - 23	Smoothed T1w Laplacian
24 - 26	Smoothed T1w Pixel Intensity
27 - 29	Smoothed T1w Gradient Magnitude
	Posterior Probabilities from BAW
24	White Matter Posterior
25	Surface Gray Matter Posterior
26	Cerebellum Gray Matter Posterior
27	Cerebellum White Matter Posterior
28	Cerebrospinal Fluid Posterior
29	Venous Blood Posterior
	Spherical Coordinates
30 - 32	rho, phi, theta Coordinate Images

with the gray matter and white matter edge masks. The feature data and the truth columns were then used to train a white matter edge classifiers and a gray matter edge classifier using `scikit-learn`[27]. The classifiers were then saved to file.

5.6 Edge Prediction Pipeline

A pipeline was built that takes advantage of the trained classifiers. The pipeline was built on top of the previously built FreeSurfer and BRAINSAutoWorkup pipelines. Figure 5.2 provides a visualization of the general processing steps in the edge prediction pipeline. Due to the promising results of the FreeSurfer + LOGISMOS-B pipeline, the pipeline using the trained classifiers used the white matter surfaces output from `recon-all`. The features were all extracted from BRAINSAutoWorkup as described in Section 5.3. The brain label segmentation as well as the registered HNCMA atlas from BRAINSAutoWorkup was used as inputs for LOGISMOS-B.

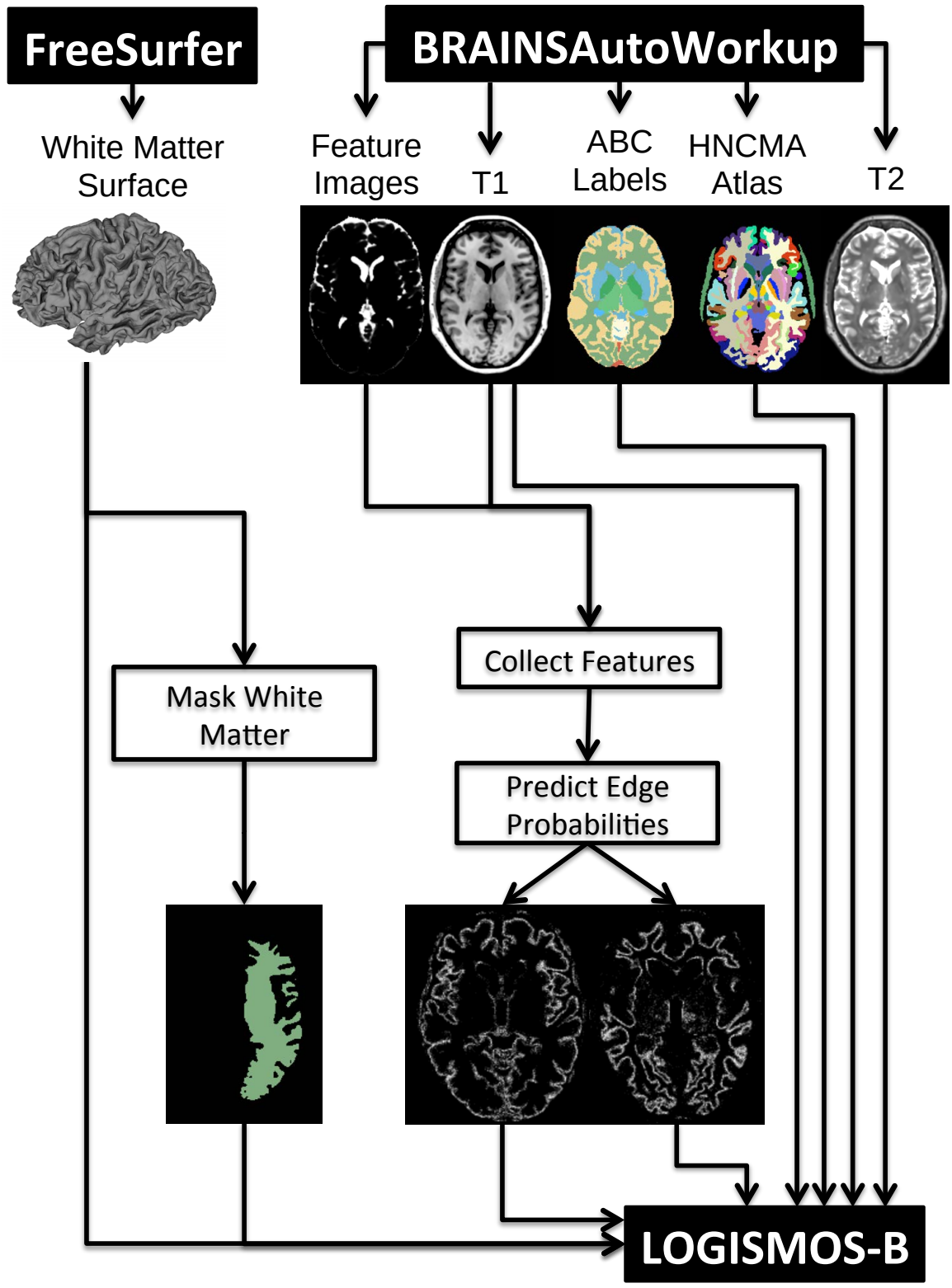

Figure 5.2: The Machine Learning Pipeline. This combines pipeline FreeSurfer and BRAINSAutoWorkup.

CHAPTER 6
RESULTS

6.1 Evaluating Algorithm Performance

Table 6.1 shows the root mean square error in mm^3 of each of the three workflows tested. The known simulated changes were used as the ground truth. The method that used a trained random forest (RF) to predict the cortical edges and then pass the predicted probabilities to LOGISMOS-B provided the lowest root mean square error (RMSE). The RMSE was lower for the RF + LOGIMSOS-B pipeline was lower than all the other workflows when the cortical volume change was in the Left Middle Temporal region as well as when the cortical volume change was in the Right Rostral Middle Frontal region. Also, the RMSE were higher for the workflows involving LOGISMOS-B when the volume change was in the Right Rostral Middle Frontal region than in the Left Middle Temporal Region. On the other hand, FreeSurfer showed the opposite trend and performed better in Right Rostral Middle Frontal region than in the Left Middle Temporal Region.

Figures 6.1 and 6.2 show the distribution of the measured cortical volume changes as compared to the target volume changes (represented by the black line). The "BAW + LOGISMOS-B" and "FreeSurfer + LOGISMOS-B" workflows tracked well with the atrophy simulations on average, but both methods show large variations in error. These methods also appeared to under represent the amount of cortical volume increase in the growth simulations. Both the "RF + LOGISMOS-B" and the

Table 6.1: Root mean square error in mm^3 of each of the workflows utilized to measure the simulated volume changes.

	Root Mean Square Error (mm^3)		
	Left Middle Temporal Region (n=282)	Right Rostral Middle Frontal Region (n=296)	Total (n=578)
BAW + LOGISMOS-B	1.42e+03	1.80e+03	1.62e+03
FreeSurfer + LOGISMOS-B	1.05e+03	1.50e+03	1.30e+03
FreeSurfer	1.07e+03	9.54e+02	1.01e+03
RF + LOGISMOS-B	**5.91e+02**	**6.87e+02**	**6.42e+02**

FreeSurfer methods appear to track well with the target volume changes, but the FreeSurfer method consistently measured too much volume decrease for the atrophy simulations and too little volume increase for the growth simulations. Figures 6.3 and 6.4 further reinforce these observations. The "BAW + LOGISMOS-B" and "FreeSurfer + LOGISMOS-B" methods both had higher variances in the amount of error, but their median values were close to 0 for the atrophy simulations. The "RF + LOGISMOS-B" and the FreeSurfer methods had much tighter error distributions. The median values for error for the "RF + LOGISMOS-B" method that utilized machine learning was close to zero for both the atrophy simulations and the growth simulations.

6.2 Evaluating the Significance of Differences Between Methods

We used R[28] and lme4[2] to perform a linear mixed effects analysis of the relationship between volume measurements and the tool used to generate the mea-

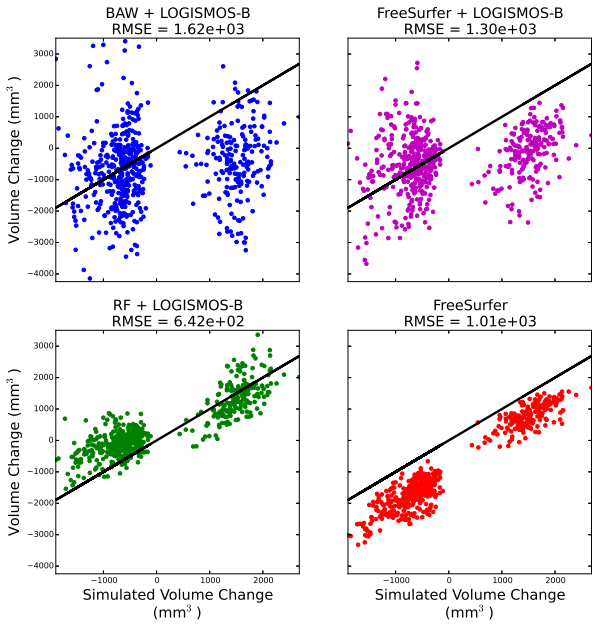

Figure 6.1: The simulated volume change (x-axis) compared to the measured volume changes (y-axis) in the cortex for each pipeline. The black line (y=x) represents the target of the measured volume changes matching the simulated volume changes.

sures. In this analysis we modeled change in volume as the dependant variable. The pipeline being used and the rate of atrophy were modeled as fixed effects. The intercepts for subjects were used as random effects. The general linear hypothesis was that the means are the same. P-values were obtained by likelihood ratio tests of the full model with the effect in question against the model without the effect in question. The p-values of linear mixed effects analysis are shown in Table 6.2. The pipeline utilizing machine learning (RF+LOGB) produced volume changes that were significantly different from the other three pipelines. Therefore, not only did the machine

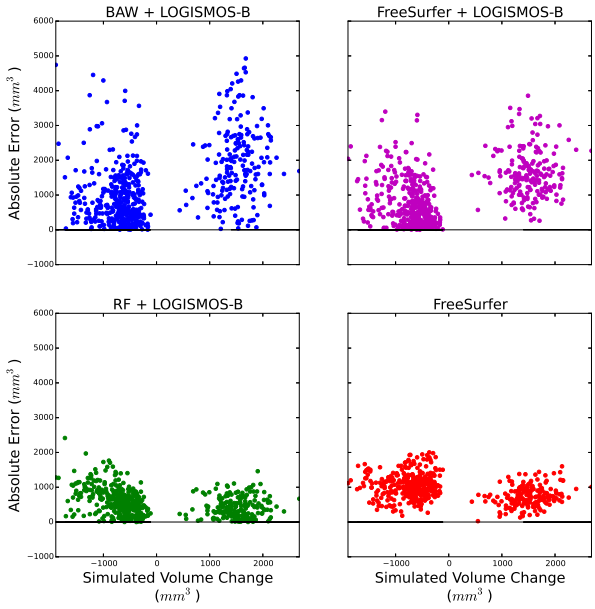

Figure 6.2: Scatter plot of the absolute error (y-axis) of the pipelines to measure the simulated volume change (x-axis) in the cortex. The black line (y=0) represents the target absolute error of the measured volume changes compared to the simulated volume changes.

learning method track the simulated volume changes with lower amounts of error on average, but the difference between the machine learning method and all other methods was significant.

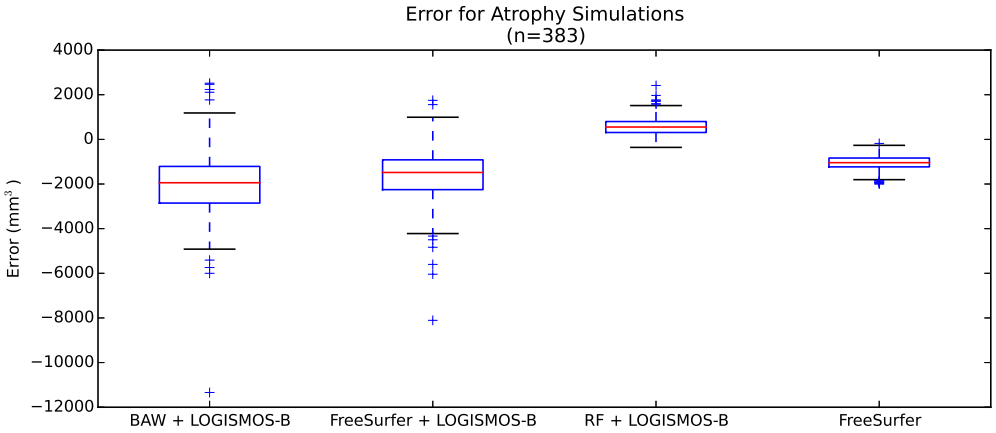

Figure 6.3: Box plot showing the errors of the workflows when measuring the volume changes for the atrophy simulations.

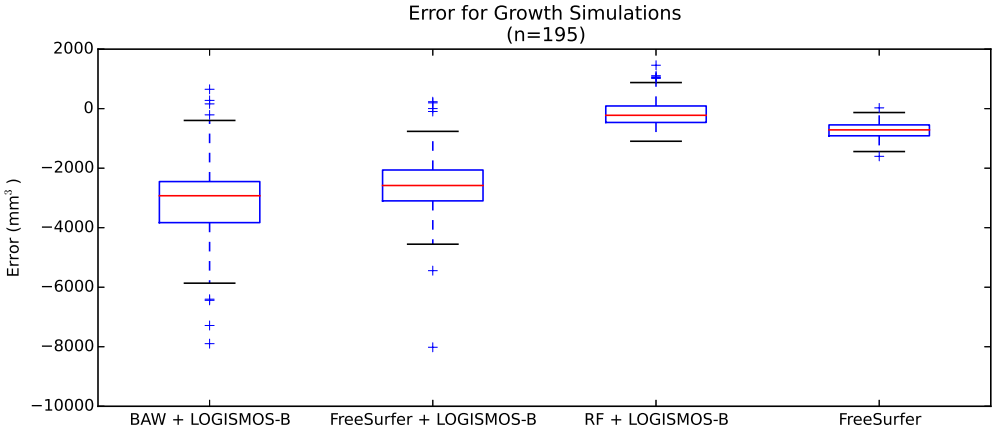

Figure 6.4: Box plot showing the errors of the workflows when measuring the volume changes for the growth simulations.

Table 6.2: P-values of linear mixed effects analysis comparing the means of the volume changes under the general linear hypothesis that the means are the same. The pipeline utilizing machine learning (RF+LOGB) produced volume changes that were significantly different from the other three pipelines.

	p-values			
	BAW+LOGB	FS+LOGB	FS	**RF+LOGB**
BAW+LOGB		.0537	<0.001	**<0.001**
FS+LOGB	.0537		<0.001	**<0.001**
FS	<0.001	<0.001		**<0.001**
RF+LOGB	<0.001	<0.001	<0.001	

CHAPTER 7
CONCLUSIONS

7.1 Summary

In this project, I successfully implemented and compared 4 methods for reconstructing cortical surfaces. I developed image processing pipelines for all 4 of these methods that use computing resources efficiently and are optimized for running in parallel in high performance computing environments. I also created a data set of simulated cortical volume changes in human MR scans. I used that dataset to compare the 4 methods for reconstructing cortical surfaces. Since the volume changes in this dataset are known, I compared the resulting cortical volume changes from the processing methods to the known target changes of the simulation data set. The results of my test showed that utilizing machine learning to predict the edges of the cerebral cortex prior to reconstructing those cortical edges as surfaces resulted in an improved ability to track the cortical volume changes.

The first method for reconstructing the cortical surfaces utilized BRAINSAutoWorkup(BAW) to normalize and segment the input scans. The next step in this method was to use the segementations provided by BAW to estimate the white matter surface. This information was then passed to the graph segmentation tool, LOGISMOS-B, which reconstructed the cortical surfaces. While this method (BAW+LOGISMOSB) was able to track with the target volume changes on average, the precision of the measurements was poor, and this method resulted in the highest

standard error out of all the methods tested.

The second method used FreeSurfer instead of BRAINSAutoworkup for the preprocessing of the MR scans. When tested, the use FreeSurfer for the preprocessing and initial estimation of the white matter surface proved to reduce the error. However, this method still resulted in the second highest standard error out of the four methods.

The third method combined the first two methods and added a machine learning classifier (more specifically, a random forest) to the previous method. This classifier was trained to predict the edges of the cerebral cortex using manually segmented atlases of human MR brain scans. During processing, this classifier takes in a set of precomputed image features from the input T1w scan and outputs a probability map for both the inside and outside edges of the cerebral cortex. This probability map is then used by LOGISMOS-B to reconstruct the cortical surfaces. This method produced the lowest standard error of the 4 methods tested.

The final method was FreeSurfer's `recon-all` method for reconstructing the cortical surfaces. Though I did not develop FreeSurfer's `recon-all` processing method, I did develop a novel method for implementing FreeSurfer's `recon-all`. I created a pipeline using `Nipype` that executed the exact same processing steps as `recon-all`, but was better optimized for parallel processing. During testing, FreeSurfer performed better than the first two methods that utilized LOGISMOS-B, but produced larger standard error than the method utilizing machine learning.

7.2 Discussion

Morphometrics describing the form of the cerebral cortex have allowed researchers to quantify differences and changes in the cortex and find correlations to differences and changes in neurological health. Morphometrics such as cortical thickness and surface area have been used to describe abnormalities in Alzheimers disease[21], Parkinsons disease[42], Huntingtons disease[30], and autism[9]. In the case of autism, cortical morphometrics were even used to predict a future diagnosis in infants[9].

Previous studies have shown FreeSurfer, a freely available tool for reconstructing cortical surfaces, to be both accurate and reproducible[29][3][12]. However, a recent study demonstrated that many of the automatic results from FreeSurfer were not reliable[12]. The reproducibility of FreeSurfer for the test-retest data was improved following the removal of the scans that failed visual inspection[12].

Having more precise cortical reconstructions would allow for the researchers to increase the power of studies involving cortical morphometrics. Increased power could in turn result in correlations being found between cortical form and neurological health with fewer numbers of subjects. Furthermore, since increased precision allows for smaller changes in the cortical surfaces to be detected, new correlations between the form of the cerebral cortex and neurological health could be discovered.

My research shows that changes in the cerebral cortex can be tracked with more precision by utilizing machine learning. The machine learning classifier I trained takes advantage of data from manual segmentations and uses it to guide the graph segmentation tool along the cortical edges. The classifier is able to detect the dif-

ference between different strong edges in the brain. This allowed for the classifier to keep the graph segmentation from mistaking an edge such as the CSF-skull boundary for the outer edge of the cortex.

Therefore, using my machine learning method for reconstructing the cortical surfaces could detect changes with more precision than FreeSurfer. By providing more precise reconstructions with less variation, this method could, in turn, reduce the number of subjects required to determine if differences in the cortical morphology between two populations is significant. Since recruiting and enrolling subjects in research studies is expensive and time consuming, reducing the need for the number of subjects would greatly benefit neuroimaging researchers.

Furthermore, adopting my machine learning method could provide more reliable cortical surface reconstructions. In a large, multi-site study of around 800 subjects, if FreeSurfer does not provide reliable cortical surface reconstructions for 38 percent[12] of the subjects than 300 of the 800 subjects would be excluded from analysis. A more reliable method for reconstructing the cortical surfaces could make the number of subjects excluded much less.

The results of this research may also be subject to bias. The most obvious form of bias comes from the simulated cortical volume changes. If the simulation of cortical changes is not realistic, the improved precision found by using machine learning may not be reproducible in non-simulated data. Furthermore, partial volume effects that occur when the cortical surfaces were converted into cortical volume masks may also have biased the results. However, it would be expected that the partial volume

effects would impact all of the reconstructed surfaces similarly. Therefore any bias from partial volume effects should be minimal.

Overall, the results of my research are very exciting. The methods I have demonstrated utilize machine learning to provide information for reconstructing the cortical surfaces could be used to provide more precise measurements to a wide array of future studies that aim to analyze the shape and form of the cortical surfaces.

7.3 Recommendations for Future Work

It should be noted that this research did not evaluate the accuracy of the reconstructed surfaces themselves. Instead, this research compared the ability of the reconstructed surfaces to accurately detect changes. Future research could involve performing more comparisons between the processing methods. Reproducibility of methods for reconstructing cortical surfaces could be tested using a test-retest set of images. A test-retest data set contains images from the same subject scanned at two different times. The two scanning sessions (test and retest) are performed within a short amount of time from each other, so as to minimize any changes in anatomy that might take place. Using a test-retest data set would add insight into the variance that exists in each of the methods measurements.

Another area of future work on this research involving machine learning for cortical surface reconstruction could reduce the number of processing steps required. Right now, the workflow uses BAW+FreeSurfer+RF+LOGISMOS-B, which makes the compute time rather large.

The results might also be improved by adding data to the training set that would allow the workflow to make better predictions. During testing the machine learning workflow erred on the side of underestimating the volume changes for the atrophy simulations. This could be due to the classifier overfitting to a data set that with a limited range of cortical thicknesses. When the cortex becomes too thin, the classifier might be failing to recognize the edge, because it has not been trained to classify brains with that thin of a cortex. Adding more data with diverse anatomies could potentially fix the underestimation of cortical volume changes. Training data that contains both T1w and T2w scans would allow the classifier to utilize multiple modalities when making predictions. Furthermore, training data from many different scanners and sites would allow the machine learning processing workflow to provide accurate segmentations regardless of the scanner used.

More research could also be focused on improving the graph construction used by LOGISMOS-B. Based on qualitative visual analysis, the graph construction contained regions of high node density in the sulci and regions with vary low node density on the gray matter/csf boundaries. Low node density often resulted in the graph optimizer not being able to follow the edge predictions as well as it does in the high density regions. While regions of higher node density qualitatively appeared more accurate and were aligned better with the predictions output by the machine learning classifier. Therefore, a method for a higher and more uniform distribution of nodes could lead to better cortical surface reconstructions.

Finally, creating a workflow that only uses machine learning and FreeSurfer

could provide insight into the effectiveness of LOGISMOS-B's versus FreeSurfer's method for reconstructing cortical surfaces.

REFERENCES

[1] B.B. Avants, N. Tustison, and G. Song. Advanced normalization tools (ANTS). *Insight J*, 2009.

[2] Douglas Bates, Martin Mächler, Ben Bolker, and Steve Walker. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48, 2015. doi: 10.18637/jss.v067.i01.

[3] Francesco Cardinale, Giuseppa Chinnici, Manuela Bramerio, Roberto Mai, Ivana Sartori, Massimo Cossu, Giorgio Lo Russo, Laura Castana, Nadia Colombo, Chiara Caborni, Elena De Momi, and Giancarlo Ferrigno. Validation of FreeSurfer-Estimated Brain Cortical Thickness: Comparison with Histologic Measurements. *Neuroinformatics*, 12(4):535–542, oct 2014. ISSN 1539-2791. doi: 10.1007/s12021-014-9229-2. URL `http://link.springer.com/10.1007/s12021-014-9229-2`.

[4] A M Dale, B Fischl, and M I Sereno. Cortical surface-based analysis. I. Segmentation and surface reconstruction. *NeuroImage*, 9(2):179–94, feb 1999. ISSN 1053-8119. doi: 10.1006/nimg.1998.0395. URL `http://www.sciencedirect.com/science/article/pii/S1053811998903950`.

[5] B. Fischl, A. Liu, and A. M. Dale. Automated manifold surgery: constructing geometrically accurate and topologically correct models of the human cerebral cortex. *IEEE Medical Imaging*, 20(1):70–80, Jan 2001.

[6] Alejandro F Frangi, Wiro J Niessen, Koen L Vincken, and Max A Viergever. *Multiscale vessel enhancement filtering*, pages 130–137. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. ISBN 978-3-540-49563-5. doi: 10.1007/BFb0056195. URL `http://dx.doi.org/10.1007/BFb0056195`.

[7] K. Gorgolewski, C. D. Burns, C. Madison, D. Clark, Y. O. Halchenko, M. L. Waskom, and S. S. Ghosh. Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in python. *Front Neuroinform*, 5:13, 2011.

[8] Michael Halle, Ion-FlorinF. Talos, Marianna Jakab, Nikos Makris, Dominik Meier, Laurence Wald, Bruce Fischl, and Ron Kikinis. Multi-modality mri-based atlas of the brain. 11 2015.

[9] Heather Cody Hazlett, Hongbin Gu, Brent C Munsell, Sun Hyung Kim, Martin Styner, Jason J Wolff, Jed T Elison, Meghan R Swanson, Hongtu Zhu, Kelly N Botteron, D Louis Collins, John N Constantino, Stephen R Dager, Annette M Estes, Alan C Evans, Vladimir S Fonov, Guido Gerig, Penelope Kostopoulos, Robert C McKinstry, Juhi Pandey, Sarah Paterson, John R Pruett, Robert T Schultz, Dennis W Shaw, Lonnie Zwaigenbaum, Joseph Piven, and The IBIS Network. Early brain development in infants at high risk for autism

spectrum disorder. *Nature*, 542(7641):348–351, feb 2017. ISSN 0028-0836. URL http://dx.doi.org/10.1038/nature2136910.1038/nature21369http://www.nature.com/nature/journal/v542/n7641/abs/nature21369.html{\#}supplementary-information.

[10] L. J. Hogstrom, L. T. Westlye, K. B. Walhovd, and A. M. Fjell. The Structure of the Cerebral Cortex Across Adult Life: Age-Related Patterns of Surface Area, Thickness, and Gyrification. *Cerebral Cortex*, 23(11):2521–2530, nov 2013. ISSN 1047-3211. doi: 10.1093/cercor/bhs231. URL http://www.ncbi.nlm.nih.gov/pubmed/22892423https://academic.oup.com/cercor/article-lookup/doi/10.1093/cercor/bhs231.

[11] K. Im, J.-M. Lee, O. Lyttelton, S. H. Kim, A. C. Evans, and S. I. Kim. Brain Size and Cortical Structure in the Adult Human Brain. *Cerebral Cortex*, 18(9):2181–2191, sep 2008. ISSN 1047-3211. doi: 10.1093/cercor/bhm244. URL http://www.ncbi.nlm.nih.gov/pubmed/18234686https://academic.oup.com/cercor/article-lookup/doi/10.1093/cercor/bhm244.

[12] Zafer Iscan, Tony B Jin, Alexandria Kendrick, Bryan Szeglin, Hanzhang Lu, Madhukar Trivedi, Maurizio Fava, Patrick J McGrath, Myrna Weissman, Benji T Kurian, Phillip Adams, Sarah Weyandt, Marisa Toups, Thomas Carmody, Melvin McInnis, Cristina Cusin, Crystal Cooper, Maria A Oquendo, Ramin V Parsey, and Christine DeLorenzo. Test-retest reliability of freesurfer measurements within and between sites: Effects of visual approval process. *Human brain mapping*, 36(9):3472–85, sep 2015. ISSN 1097-0193. doi: 10.1002/hbm.22856. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84939473755{\&}partnerID=tZOtx3y1.

[13] Clifford R. Jack, Matt A. Bernstein, Nick C. Fox, Paul Thompson, Gene Alexander, Danielle Harvey, Bret Borowski, Paula J. Britson, Jennifer L. Whitwell, Chadwick Ward, Anders M. Dale, Joel P. Felmlee, Jeffrey L. Gunter, Derek L G Hill, Ron Killiany, Norbert Schuff, Sabrina Fox-Bosetti, Chen Lin, Colin Studholme, Charles S. DeCarli, Gunnar Krueger, Heidi A. Ward, Gregory J. Metzger, Katherine T. Scott, Richard Mallozzi, Daniel Blezek, Joshua Levy, Josef P. Debbins, Adam S. Fleisher, Marilyn Albert, Robert Green, George Bartzokis, Gary Glover, John Mugler, and Michael W. Weiner. The Alzheimer's Disease Neuroimaging Initiative (ADNI): MRI methods, 2008. ISSN 10531807.

[14] Kang Kang Li, Xiaodong Xiaodong Wu, D.Z. Chen, and M. Sonka. Optimal Surface Segmentation in Volumetric Images-A Graph-Theoretic Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):119–134, jan 2006. ISSN 0162-8828. doi: 10.1109/TPAMI.2006.19. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1542036.

[15] B. Karacali and C. Davatzikos. Estimating topology preserving and smooth

displacement fields. *IEEE Transactions on Medical Imaging*, 23(7):868–880, July 2004. ISSN 0278-0062. doi: 10.1109/TMI.2004.827963.

[16] Eun Young Kim and Hans J. Johnson. Robust multi-site MR data processing: iterative optimization of bias correction, tissue classification, and registration. *Frontiers in Neuroinformatics*, 7(November):1–11, 2013. ISSN 1662-5196. doi: 10.3389/fninf.2013.00029. URL `http://www.frontiersin.org/Neuroinformatics/10.3389/fninf.2013.00029/abstract`.

[17] Eun Young Kim, Vincent a Magnotta, Dawei Liu, and Hans J Johnson. Stable Atlas-based Mapped Prior (STAMP) machine-learning segmentation for multicenter large-scale MRI data. *Magnetic resonance imaging*, 32(7):832–844, may 2014. ISSN 1873-5894. doi: 10.1016/j.mri.2014.04.016. URL `http://www.ncbi.nlm.nih.gov/pubmed/24818817`.

[18] Regina EY Kim, Jane S. Paulsen, Peg Nopoulos, Ergun Uc, and Hans J. Johnson. Efficient and Extensible Workflow: Reliable whole brain segmentation for Large-scale, Multi-center Longitudinal Human MRI Analysis using High Performance/Throughput Computing Resources. In *Clinical Image-Based Procedures. Translational Research in Medical Imaging*, Chicago, USA, 2015. Springer.

[19] Gina R. Kuperberg, Matthew R. Broome, Philip K. McGuire, Anthony S. David, Marianna Eddy, Fujiro Ozawa, Donald Goff, W. Caroline West, Steven C. R. Williams, Andre J. W. van der Kouwe, David H. Salat, Anders M. Dale, Bruce Fischl, Zilles K, von Economo C, Heckers S, Selemon LD, Harrison PJ, Oldfield RC, Nelson HE, Hollingshead AB, Rakic P, Feinberg I, Murray RM, Weinberger DR, Shenton ME, Goldstein JM, Wright IC, Ashburner J, Thompson PM, Cannon TD, Talairach J, Wright IC, Sigmundsson T, Hulshoff Pol HE, Kuperberg G, Fischl B, Dale AM, Dale AM, Fischl B, Fischl B, White K, Spitzer RL, Blair JR, Kay SR, Rosas HD, Fischl B, Duvernoy HM, Crespo-Facorro B, Nelson MD, Konick LC, Pol HE Hulshoff, Foong J, Lewis DA, Selemon LD, Selemon LD, Selemon LD, Jennings J, Heckers S, Manoach DS, Callicott JH, Phillips ML, Russell TA, Shenton ME, Shapleske J, Wilke M, Ananth H, Job DE, Mathalon DH, and Cahn W. Regionally Localized Thinning of the Cerebral Cortex in Schizophrenia. *Archives of General Psychiatry*, 60(9):878, sep 2003. ISSN 0003-990X. doi: 10.1001/archpsyc.60.9.878. URL `http://archpsyc.jamanetwork.com/article.aspx?doi=10.1001/archpsyc.60.9.878`.

[20] Michele Larobina and Loredana Murino. Medical image file formats, 2014. ISSN 1618727X.

[21] Manja Lehmann, Sebastian J Crutch, Gerard R Ridgway, Basil H Ridha, Josephine Barnes, Elizabeth K Warrington, Martin N Rossor, and Nick C Fox. Cortical thickness and voxel-based morphometry in posterior cortical atrophy and typical Alzheimer's disease. *Neurobiology of aging*, 32(8):1466–76, aug

2011. ISSN 1558-1497. doi: 10.1016/j.neurobiolaging.2009.08.017. URL http://www.sciencedirect.com/science/article/pii/S0197458009002966.

[22] Bradley C Lowekamp, David T Chen, Luis Ibáñez, and Daniel Blezek. The Design of SimpleITK. *Frontiers in neuroinformatics*, 7(December): 45, 2013. ISSN 1662-5196. doi: 10.3389/fninf.2013.00045. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3874546{\&}tool=pmcentrez{\&}rendertype=abstract.

[23] Bjoern H. Menze, Andras Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, Justin Kirby, Yuliya Burren, Nicole Porz, Johannes Slotboom, Roland Wiest, Levente Lanczi, Elizabeth Gerstner, Marc-Andre Weber, Tal Arbel, Brian B. Avants, Nicholas Ayache, Patricia Buendia, D. Louis Collins, Nicolas Cordier, Jason J. Corso, Antonio Criminisi, Tilak Das, Herve Delingette, Cagatay Demiralp, Christopher R. Durst, Michel Dojat, Senan Doyle, Joana Festa, Florence Forbes, Ezequiel Geremia, Ben Glocker, Polina Golland, Xiaotao Guo, Andac Hamamci, Khan M. Iftekharuddin, Raj Jena, Nigel M. John, Ender Konukoglu, Danial Lashkari, Jose Antonio Mariz, Raphael Meier, Sergio Pereira, Doina Precup, Stephen J. Price, Tammy Riklin Raviv, Syed M. S. Reza, Michael Ryan, Duygu Sarikaya, Lawrence Schwartz, Hoo-Chang Shin, Jamie Shotton, Carlos A. Silva, Nuno Sousa, Nagesh K. Subbanna, Gabor Szekely, Thomas J. Taylor, Owen M. Thomas, Nicholas J. Tustison, Gozde Unal, Flor Vasseur, Max Wintermark, Dong Hye Ye, Liang Zhao, Binsheng Zhao, Darko Zikic, Marcel Prastawa, Mauricio Reyes, and Koen Van Leemput. The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). *IEEE Transactions on Medical Imaging*, 34(10): 1993–2024, oct 2015. ISSN 0278-0062. doi: 10.1109/TMI.2014.2377694. URL http://www.ncbi.nlm.nih.gov/pubmed/25494501http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4833122http://ieeexplore.ieee.org/document/6975210/.

[24] Nicole R. Nissim, Andrew M. O'Shea, Vaughn Bryant, Eric C. Porges, Ronald Cohen, and Adam J. Woods. Frontal Structural Neural Correlates of Working Memory Performance in Older Adults. *Frontiers in Aging Neuroscience*, 08: 328, jan 2017. ISSN 1663-4365. doi: 10.3389/fnagi.2016.00328. URL http://journal.frontiersin.org/article/10.3389/fnagi.2016.00328/full.

[25] Ipek Oguz. personal communication.

[26] Ipek Oguz and Milan Sonka. LOGISMOS-B: layered optimal graph image segmentation of multiple objects and surfaces for the brain. *IEEE transactions on medical imaging*, 33(6):1220–35, jun 2014. ISSN 1558-254X. doi: 10.1109/TMI.2014.2304499. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=4324764{\&}tool=pmcentrez{\&}rendertype=abstract.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[28] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017. URL `https://www.R-project.org`.

[29] H D Rosas, A K Liu, S Hersch, M Glessner, R J Ferrante, D H Salat, A van der Kouwe, B G Jenkins, A M Dale, and B Fischl. Regional and progressive thinning of the cortical ribbon in Huntington's disease. *Neurology*, 58(5):695–701, mar 2002. ISSN 0028-3878. URL `http://www.ncbi.nlm.nih.gov/pubmed/11889230`.

[30] H Diana Rosas, David H Salat, Stephanie Y Lee, Alexandra K Zaleta, Vasanth Pappu, Bruce Fischl, Doug Greve, Nathanael Hevelone, and Steven M Hersch. Cerebral cortex and the clinical expression of Huntington's disease: complexity and heterogeneity. *Brain : a journal of neurology*, 131(Pt 4): 1057–68, apr 2008. ISSN 1460-2156. doi: 10.1093/brain/awn025. URL `http://brain.oxfordjournals.org/content/131/4/1057.abstract`.

[31] Marie Schaer, Meritxell Bach Cuadra, Nick Schmansky, Bruce Fischl, Jean-Philippe Thiran, and Stephan Eliez. How to Measure Cortical Folding from MR Images: a Step-by-Step Tutorial to Compute Local Gyrification Index. *Journal of Visualized Experiments*, (59):e3417, jan 2012. ISSN 1940-087X. doi: 10.3791/3417. URL `http://www.ncbi.nlm.nih.gov/pubmed/22230945http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3369773http://www.jove.com/details.php?id=3417`.

[32] William J. Schroeder and Kenneth M. Martin. The visualization toolkit. In *Visualization Handbook*, pages 593–614. 2005. ISBN 9780123875822. doi: 10.1016/B978-012387582-2/50032-0.

[33] Qi Song, Junjie Bai, Mona K. Garvin, Milan Sonka, John M. Buatti, and Xiaodong Wu. Optimal Multiple Surface Segmentation With Shape and Context Priors. *IEEE Transactions on Medical Imaging*, 32(2):376–386, feb 2013. ISSN 0278-0062. doi: 10.1109/TMI.2012.2227120. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6352920`.

[34] Arthur W. Toga and Paul M. Thompson. Mapping brain asymmetry. *Nat Rev Neurosci*, 4(1):37–48, jan 2003. ISSN 1471-003X. doi: 10.1038/nrn1009. URL `http://www.nature.com/doifinder/10.1038/nrn1009http://www.nature.com/nrn/journal/v4/n1/full/nrn1007.html`.

[35] Nicholas J Tustison, Philip A Cook, Arno Klein, Gang Song, Sandhitsu R Das, Jeffrey T Duda, Benjamin M Kandel, Niels van Strien, James R Stone, James C

Gee, and Brian B Avants. Large-scale evaluation of ANTs and FreeSurfer cortical thickness measurements. *NeuroImage*, 99:166–79, oct 2014. ISSN 1095-9572. doi: 10.1016/j.neuroimage.2014.05.044. URL http://www.sciencedirect.com/science/article/pii/S1053811914004091.

[36] Hongzhi Wang, Jung W Suh, Sandhitsu R Das, John B Pluta, Caryne Craige, and Paul A Yushkevich. Multi-Atlas Segmentation with Joint Label Fusion. *IEEE transactions on pattern analysis and machine intelligence*, 35(3):611–23, mar 2013. ISSN 1939-3539. doi: 10.1109/TPAMI.2012.143. URL http://www.ncbi.nlm.nih.gov/pubmed/22732662http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3864549.

[37] Anderson M Winkler, Mert R Sabuncu, B T Thomas Yeo, Bruce Fischl, Douglas N Greve, Peter Kochunov, Thomas E Nichols, John Blangero, and David C Glahn. Measuring and comparing brain cortical surface area and other areal quantities. *NeuroImage*, 61(4):1428–43, jul 2012. ISSN 1095-9572. doi: 10.1016/j.neuroimage.2012.03.026. URL http://www.ncbi.nlm.nih.gov/pubmed/22446492http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3641659.

[38] M.W. Woolrich, S. Jbabdi, B. Patenaude, M. Chappell, S. Makni, T. Behrens, C. Beckmann, M. Jenkinson, and S.M Smith. Bayesian analysis of neuroimaging data in FSL. *NeuroImage*, page 45, 2009.

[39] Andrew Worth and Jason Tourville. Acceptable values of similarity coefficients in neuroanatomical labeling in MRI. Technical Report 3, 2015. URL http://www.neuromorphometrics.com/?p=366.

[40] Yin Yin Yin, Xiangmin Xiangmin Zhang, Rachel Williams, Xiaodong Xiaodong Wu, Donald D Anderson, and Milan Sonka. LOGISMOSLayered Optimal Graph Image Segmentation of Multiple Objects and Surfaces: Cartilage Segmentation in the Knee Joint. *IEEE Transactions on Medical Imaging*, 29(12):2023–2037, dec 2010. ISSN 0278-0062. doi: 10.1109/TMI.2010.2058861. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5512662.

[41] Paul A Yushkevich, Hongzhi Wang, John Pluta, and Brian B Avants. From label fusion to correspondence fusion: a new approach to unbiased groupwise registration. *Proceedings. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 956–963, 2012. ISSN 1063-6919. doi: 10.1109/CVPR.2012.6247771. URL http://www.ncbi.nlm.nih.gov/pubmed/24457950http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3896982.

[42] Mojtaba Zarei, Naroa Ibarretxe-Bilbao, Yaroslau Compta, Morgan Hough, Carme Junque, Nuria Bargallo, Eduardo Tolosa, and Maria Jose Martí. Cortical thinning is associated with disease stages and dementia in Parkinson's disease. *Journal of neurology, neurosurgery, and psychiatry*, 84(8):875–81,

aug 2013. ISSN 1468-330X. doi: 10.1136/jnnp-2012-304126. URL http://jnnp.bmj.com/cgi/content/long/84/8/875.

[43] Fei Zhao, Honghai Zhang, Andreas Wahle, Matthew T. Thomas, Alan H. Stolpen, Thomas D. Scholz, and Milan Sonka. Congenital aortic disease: 4D magnetic resonance segmentation and quantitative analysis. *Medical Image Analysis*, 13(3):483–493, 2009. ISSN 13618415. doi: 10.1016/j.media.2009.02.005.