Master's Programme in Health Informatics
Spring Semester 2014
Degree thesis, 30 Credits

Automated magnetic resonance image processing workbench for Alzheimer's disease-related research and diagnosis

Author: Sami Andberg

Author: Sami Andberg
Co-supervisor: Assoc. prof. Tze-Yun Leong, School of Computing, National University of Singapore
Co-supervisor: Assoc. prof. Panagiotis Papapetrou, Dept. of Computer and Systems Sciences, Stockholm University
Examiner: Prof. Sabine Koch, Dept. of Learning, Informatics, Management and Ethics, Karolinska Institutet
Examiner: Prof. Uno Fors, Dept. of Computer and Systems Sciences, Stockholm University

Karolinska Institutet

Stockholm University

Master's Programme in Health Informatics
Spring Semester 2014
Degree thesis, 30 Credits

Affirmation

I hereby affirm that this Master thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text. This work has not been submitted for any other degree or professional qualification except as specified; nor has it been published.

Singapore, 2014-05-15

Sami Andberg

Master's Programme in Health Informatics
Spring Semester 2014
Degree thesis, 30 Credits

Automated magnetic resonance image processing workbench for Alzheimer's disease-related research and diagnosis

Abstract

Background: Dementia-related diseases, especially Alzheimer's disease, are affecting growing number of people as the average life span increases. However the manual processing of magnetic resonance brain images (MRI) needed for research or diagnostic purposes is often laborious and time consuming, which limits, for example, the possibilities of doing advanced machine learning on large datasets of patient data.

Objective: The objective of this thesis project was to evaluate the suitability of available free and open source tools for automating the MRI processing and dementia-related analysis by setting up an example implementation, and then comparing the results to the state of the art in the domain area.

Methods: The selected research method for the project was explorative feasibility study, which enabled time to be spent testing the tools, setting up the environment, developing the needed scripts and evaluating the usefulness and predictive results of the end product.

Results: An example implementation was set up using the free Nipype framework and other related processing tools. A machine learning implementation was also set up to demonstrate and evaluate the possibilities of automated data analysis of the processing results.

Conclusion: The implementation shows that it is possible to build functional workflows using available free and open source tools, but the setup process is time consuming and requires both content knowledge and programming experience. The machine learning results of the implemented example solution were well below the results presented in state of the art papers, and as such the example solution cannot be considered diagnostically beneficial in its current state.

Keywords: Dementia, Alzheimer Disease, Magnetic Resonance Imaging, Artificial Intelligence, Computer Assisted Image processing

Acknowledgements

I would like to thank all of my supervisors for their important feedback and suggestions. Warm gratitude goes also to all of the members of Medical Computing Laboratory at the School of Computing at the National University of Singapore for making me feel a part of the lab for the exchange period, especially recognizing the contributions of Parvathy Sudhir Pillai, who outlined the initial workflow model and provided assistance and additional results related to the thesis work along the way, and Cristina Altomare for rigorious testing. My warm appreciation and thanks also for the staff at the National Neuroscience Institute in Singapore, who acted as the client for this project and came up with important observations and suggestions for the work. Thanks are also in order for John Ashburner and Giullaume Flandin from the Wellcome Trust Centre for Neuroimaging at UCL for the access to the closed beta version of SPM Standalone. And finally, thanks for all of the staff and students at Karolinska Institutet's and Stockholm University's joint health informatics master's program for the fun times, and for Karolinska Institutet for the special opportunity and financial support to facilitate the exchange studies and thesis work in Singapore.

For Alzheimer's Disease Neuroimaging Initiative (ADNI)

Data used in preparation of this thesis were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database (adni.loni.usc.edu). As such, the investigators within the ADNI contributed to the design and implementation of ADNI and/or provided data but did not participate in analysis or writing of this report. A complete listing of ADNI investigators can be found at:
http://adni.loni.usc.edu/wp-content/uploads/how_to_apply/ADNI_Acknowledgement_List.pdf

The ADNI was launched in 2003 by the National Institute on Aging (NIA), the National Institute of Biomedical Imaging and Bioengineering (NIBIB), the Food and Drug Administration (FDA), private pharmaceutical companies and non-profit organizations, as a $60 million, 5-year public- private partnership. The primary goal of ADNI has been to test whether serial magnetic resonance imaging (MRI), positron emission tomography (PET), other biological markers, and clinical and neuropsychological assessment can be combined to measure the progression of mild cognitive impairment (MCI) and early Alzheimer's disease (AD). Determination of sensitive and specific markers of very early AD progression is intended to aid researchers and clinicians to develop new treatments and monitor their effectiveness, as well as lessen the time and cost of clinical trials.

The Principal Investigator of this initiative is Michael W. Weiner, MD, VA Medical Center and University of California – San Francisco. ADNI is the result of efforts of many co- investigators from a broad range of academic institutions and private corporations, and subjects have been recruited from over 50 sites across the U.S. and Canada. The initial goal of ADNI was to recruit 800 subjects but ADNI has been followed by ADNI-GO and ADNI-2. To date these three protocols have recruited over 1500 adults, ages 55 to 90, to participate in the research, consisting of cognitively normal older individuals, people with early or late MCI, and people with early AD. The follow up duration of each group is specified in the protocols for ADNI-1, ADNI-2 and ADNI-GO. Subjects originally recruited for ADNI-1 and ADNI-GO had the option to be followed in ADNI-2. For up-to-date information, see www.adni-info.org.

Data collection and sharing for this project was funded by the Alzheimer's Disease Neuroimaging Initiative (ADNI) (National Institutes of Health Grant U01 AG024904) and DOD ADNI (Department of Defense award number W81XWH-12-2-0012). ADNI is funded by the National Institute on Aging, the National Institute of Biomedical Imaging and Bioengineering, and through generous contributions from the following: Alzheimer's Association; Alzheimer's Drug Discovery Foundation; BioClinica, Inc.; Biogen Idec Inc.; Bristol-Myers Squibb Company; Eisai Inc.; Elan Pharmaceuticals, Inc.; Eli Lilly and Company; F. Hoffmann-La Roche Ltd and its affiliated company Genentech, Inc.; GE Healthcare; Innogenetics, N.V.; IXICO Ltd.; Janssen Alzheimer Immunotherapy Research & Development, LLC.; Johnson & Johnson Pharmaceutical Research & Development LLC.; Medpace, Inc.; Merck & Co., Inc.; Meso Scale Diagnostics, LLC.; NeuroRx Research; Novartis Pharmaceuticals Corporation; Pfizer Inc.; Piramal Imaging; Servier; Synarc Inc.; and Takeda Pharmaceutical Company. The Canadian Institutes of Health Research is providing funds to support ADNI clinical sites in Canada. Private sector contributions are facilitated by the Foundation for the National Institutes of Health (www.fnih.org). The grantee organization is the Northern California Institute for Research and Education, and the study is coordinated by the Alzheimer's Disease Cooperative Study at the University of California, San Diego. ADNI data are disseminated by the Laboratory for Neuro Imaging at the University of Southern California.

Table of Contents

1. Introduction 1
1.1 Background information 1
1.1.1 Dementia and Alzheimer's disease 1
1.1.2 Magnetic Resonance Imaging 1
1.2 Problem description 2
1.3 Research aim and objectives 2
1.4 Research questions 2
2. Methods 3
2.1 Research approach 3
2.2 Development and prototyping environment 3
2.3 Image processing and analysis tools 3
2.3.1 Nipype framework 4
2.3.2 SPM8 Standalone with Matlab runtime 4
2.3.3 Other tools utilised by the development environment 4
2.3.5 Planned workflow structure 4
3.1 Workflows designed 5
3.1.1 Image pre-processing workflow 6
3.1.2 Image processing workflow 7
3.1.3 Image co-registration and analysis workflows 8
2.5 Machine learning 9
2.3 Data collection 9
2.6 Ethical considerations 10
3. Results 11
3.1 Workflow efficiency versus manual execution 11
3.2 Machine learning results 13
3.3 Analysis 14
3.3.1 Image pre-processing 14
3.3.2 Image processing 15
3.3.3 Image analysis 16
3.3.4 Machine learning 16
3.4 Other contributions from the thesis project 17
4. Discussion 18
4.1 Discussion on methods 18
4.1.1 Limitations 18
4.2 Existing solutions 19
4.3 Future development 19
5. Conclusion 21

References **22**
Appendices **25**
Appendix A – Adding acpcdetect-interface for Nipype 25
Appendix B – Adding modulate-parameter for CreateWarped interface in Nipype 27
Appendix C – Adding get_totals -tool for Nipype 28
Appendix D – Scripts for generating MySQL tables and views 32
Appendix E – Modifications for the SVM integration 33
Appendix F – Example pre-processing workflow 34
Appendix G – Example processing workflow 36
Appendix H – Example co-registering workflow 39
Appendix I – Example analysis workflow 41
Appendix J – Test environment install script 43

List of Figures

Figure 1: Initially proposed structure for the workflow 5
Figure 2: Final structure for the three workflows developed 5
Figure 3: Pre-processing workflow 6
Figure 4: Processing workflow 7
Figure 5: Co-registration and Analysis workflows, including the sub-workflow for iterating over subjects 8

List of Images

Image 1: Example slice of a MR-image to be processed 14
Image 2: Example of image calibrated on anterior commissure 14
Image 3: Example of extracted brain (i.e. after skull removal) 15
Image 4: Example of normal grey matter segmentation result. 15
Image 5: Example of normal white matter segmentation result 15
Image 6: Example of corrupted grey matter segmentation result. 15
Image 7: Example of template image 16
Image 8: Example of subject's flowfield 16
Image 9: Example of normalised GM image 16

List of Tables

Table 1: Demographics of the test data set (a subset of ADNI MRI) 10
Table 2: Tools selected for Pre-processing, processing and analysis steps in the final workflows. 11
Table 3: Pre-processing steps and related durations 12
Table 4: Processing steps and related durations 12
Table 5: Analysis steps and related durations 13
Table 6: Results of SVM classification for Alzheimer's patients (AD) and healthy controls (HC) 14

List of Abbreviations

AC	Anterior commissure – a brain region often used as origin for image calibration
AD	Alzheimer's disease
BET	Brain extraction tool, part of FSL package
CSF	Cerebrospinal fluid – the liquid in the brain filling the empty cavities in the skull
DARTEL	DARTEL is part of SPM package, used to make templates and flowfields.
DICOM	Digital imaging and communications in medicine - refers to a related picture format.
DJANGO	Python-based framework for web programming
FSL	FMRIB Software Library – a package for brain image processing
GM	Grey matter – area packed with the neuronal cell bodies and synapses
MATLAB	A high level programming, computing and visualisation tool.
MNI	Montreal Neurological Institute – refers to 'MNI standard space' for brain images.
MR	Magnetic resonance (see MRI)
MRI	Magnetic resonance imaging – a method for acquiring images of brain physiology
NIFTI	Neuroimaging informatics technology initiative - refers to a related file format NIfTI-1.
NIPYPE	Neuroimaging in Python: Pipelines and Interfaces – a tool for automating workflows
ROI	Region of Interest
SPM	Statistical parametric mapping – a brain image-processing package for Matlab
SVM	Support vector machine
T1	MRI technique for brain imaging, often utilised for capturing structural images
VBM	Voxel-based morphometry
VPN	Virtual private networking
WM	White matter – brain area consisting mostly of myelinated axons

1. Introduction

1.1 Background information

1.1.1 Dementia and Alzheimer's disease

In a world where average life span is increasing, dementia is a growing concern for disability in the late years of live. According to a Global Burden of Disease -report from 2013, the prevalence of dementia-related disability adjusted life years almost doubled (increase of 99.3%) from 1990 to 2010 (1). A recent meta-study estimated that from the global population of people over 60 year years old there were 35.6 million people living with dementia, which would give it a prevalence between 5% to 7% in most regions of the world (2). This amounts to an estimated global dementia-related cost to society in the order of US$600 billion annually, with most of the costs occurring in western Europe and North America (3). A Delphi consensus study on dementia predicts, that in the near future the biggest increase of dementia cases will occur in southern Asia and western Pacific region (including India and China), and with a growth of over 300% this area would soon surpass western Europe as the area with most dementia cases in absolute numbers (4).

Clinically dementia is categorized as a neurodegenerative disease, which can be subdivided into different subtypes, the most common ones being Alzheimer's disease and vascular dementia (5). Other common forms of dementia include frontotemporal dementia, alcohol related dementia and dementia with Lewy bodies, among others. In this thesis, the focus is on Alzheimer's disease as it has the greatest prevalence, but the used approach and tools should also be usable for other types of dementia as well.

1.1.2 Magnetic Resonance Imaging

Neuroscientific imaging often requires quite a lot of processing in order to extract meaningful data from the acquired images. In this thesis project the imaging modality utilised was magnetic resonance imaging (MRI) where patients' head is placed inside a large imaging machine, which induces a strong magnetic field onto a specific part of the brain, and the observes and records the phenomenon where the molecules, taken out from their initial state by the magnetic field, rebound after the magnetic field is released. There are many different MR imaging techniques, which emphasise different aspects of brain physiology. The image processing as utilised in this thesis in relation to Alzheimer's disease classification is based on the property of the T1-weighted MR-images to show different types of brain matter in different intensities; grey matter (GM) dominated areas, i.e. the neuronal bodies and synapses, are darker, white matter (WM) areas, the myelinated connections going mainly between different brain areas, are lighter, and the hollow cavities, which are filled with cerebrospinal fluid (CSF), are the darkest. This makes it possible to use different algorithms to automatically process and extract certain types of data from the images.

1.2 Problem description

General practitioners, as being the physician most often meeting the patient, can be assumed to be in a good position for diagnosing dementia due to their access to the patient, but they contribute to roughly half of the dementia diagnosis in the areas studied in the US (6). A study on general practitioners' knowledge and skills about dementia diagnosis and management found out, that one third of participating physicians expressed concern about their diagnostic skills related to dementia (7). Thus it could be beneficial, if there would be an automated tool that could provide some decision support in the subject area.

It has been stated, that "the past 25 years have seen a revolution in imaging technology and with it a revolutions of in vivo analysis of neuropathological changes" when referring to dementia pathology in structural neuroimaging (8). Along with the improved magnetic resonance imaging (MRI) technologies there have been proof-of-concept trials on using automated processing of the magnetic resonance brain images, and the results have been encouraging (9,10). A study of automatically estimating the volumetric data over patients' segmented brain images and overlaying it on the image itself was seen having significant impact on diagnostic confidence on forced-choice Alzheimer's disease diagnosis (11). Thus automation could possibly contribute with the decision support assistance sometimes needed in this area.

1.3 Research aim and objectives

Some of the research published in the area related to processing structural magnetic images is based on tools that are developed by the scientific community and made freely available for users. However, even when using these tools, processing patients' magnetic resonance images manually one by one for diagnostic or research purposes is a laborious and time-consuming work, which includes many subtasks utilising multiple tools. Recently new tools have been in development, which can automate this process of handling different subject's images and executing processing steps by calling other tools, one such tool being the Nipype framework (12).

The aim of this research is to evaluate the feasibility of setting up an automated workflow using Nipype framework and other free open source tools available, and to evaluate the usefulness and robustness of the workflow in relation to (Alzheimer's) dementia diagnosis. Also the possible contributions of automated workflows to enable new scientific analysis possibilities are considered.

1.4 Research questions

- Is the Nipype framework and related available free and open source tools developed enough for providing all the tools needed for building automated MR image processing workflows for Alzheimer's diagnosis?
- Can automated Nipype-based MR image processing workflows be beneficial for Alzheimer's disease-related diagnosis or research?

2. Methods

2.1 Research approach

The research approach used is this thesis was exploratory feasibility study. The feasibility of using available free and open source tools for setting up a MR-image processing workbench was evaluated by trying to construct one such implementation using available tools. For estimating the beneficial contribution of such a system, the results of an example workflow process culminating in machine learning –based classification of patient's images was observed.

2.2 Development and prototyping environment

There are a number of freely available and widely utilized tools for neuroscientific research. Many of them are developed using Python or Matlab, and most are working in the Linux environment. For this reason, the selected platform for the project was also Linux, namely Ubuntu Desktop 12.04LTS 64-bit distribution, as this allowed the access for the NeuroDebian repository for easy and convenient installation of many of the tools needed in the project (13). The initial installation of the virtual machine was done on an Oracle VirtualBox virtualization environment, but moved to VMware Fusion later on in the project. The approach of using virtual machines for development and testing was preferred as it allowed easy maintenance of the testing environment, including taking and storing snapshots of the development system before any major installations or changes to the environment.

The initial hardware utilised to run and operate the virtual machine was an old Dell desktop with Core 2 duo processor, 4GB of memory and 256 GB of hard drive. This was later on deemed insufficient and was replaced with a 13-inch MacBook Pro Retina, late 2013 model with 2.8GHz Intel Core i7 processor, 16 GB of RAM and a 512 GB solid state drive. The initial virtual machine setup used was run on 2,5GB of RAM and 2 cores on a 32-bit Ubuntu version, but as this caused memory run outs when processing larger data sets at once, so a new 64-bit Ubuntu virtual machine was taken into use with 6GB of memory allocated to the virtual machine. As most of the intensive calculation in the workflows was executed in the Matlab+SPM steps, which only utilised 1 core for calculations, allocating two cores to the virtual machine was deemed enough as adding additional cores would not have speeded up the related processing steps.

2.3 Image processing and analysis tools

For designing the actual workflows and selecting the free or open source tools for the image processing and analysis, the author went through related literature and consulted experts in the field. In the following the main tools selected to be utilised in the thesis setup are described in some detail.

2.3.1 Nipype framework

Nipype - “a flexible, lightweight and extensible neuroimaging data processing framework in Python” (12) - is an open source, community-developed tool which enables integrated workflows among different neuroscience related software packages. This allows automation and even batch execution of workflows, which can include multiple steps utilizing multiple different software packages and be run on the local machine or on special clusters. In the test environment Nipype was utilised to run the workflows and to coordinate the execution and parameters of other related software.

2.3.2 SPM8 Standalone with Matlab runtime

SPM8 is a widely used tool for neuroscientific medical imaging, which is executed using Matlab (14,15). While the normal SPM8 package distribution requires a commercially licenced version of Matlab to run, the authors of SPM8 have also published information of a closed beta version of SPM8 Standalone, which comes pre-compiled with free Matlab runtime, and doesn’t require full Matlab suite to run (16). This standalone version was requested and received from the Wellcome Trust Centre for Neuroimaging at UCL, and it was the version used in this thesis work unless otherwise stated. The main tools planned to be utilised from the SPM8 toolbox in the workflow were related to the DARTEL (Diffeomorphic Anatomical Registration Through Exponentiated Lie Algebra), including newsegment and generating templates and processing individual images based on the templates (17).

2.3.3 Other tools utilised by the development environment

Other related tools in the area included MRIcron DCM2NII converter for converting the original DICOM images to NIfTI-1 format and Bet2 from the FSL package for removing skull and other unnecessary tissues (18,19). For the calculation of grey matter volumes the literature suggested Ged Ridgway’s get_totals.m –script, with related masks generated in Matlab by using WFU PickAtlas (20,21).

Initially a web based user interface (UI) for using the workbench for non-technical end users was planned to be one major part of the thesis project. However, this was gradually dropped out due to time constrains. A simple web-based user interface was added back to the project in the last weeks due to the request of the customer. The main function of the UI setup is to help demonstrate the system and to act as a guideline implementation to assist in the possible continuation project. The Django framework was selected as the user interface platform due to the fact that it is also based on Python, supports MySQL databases and allows fast prototyping and implementation (22). The example user interface implementation is partly based on codes adapted from several Django tutorials (23,24). The user interface implementation is mostly left out of this thesis report, and the codes are not attached as appendices, but the codes will be made available online (25).

2.3.5 Planned workflow structure

The initial workflow design consisted of running through all the tools observed to be relevant to the process, as described above. The workflow was initially planned to be based on Nipype framework, which would be used to automate and initiate the specific tools responsible for doing the actual processing for each step.

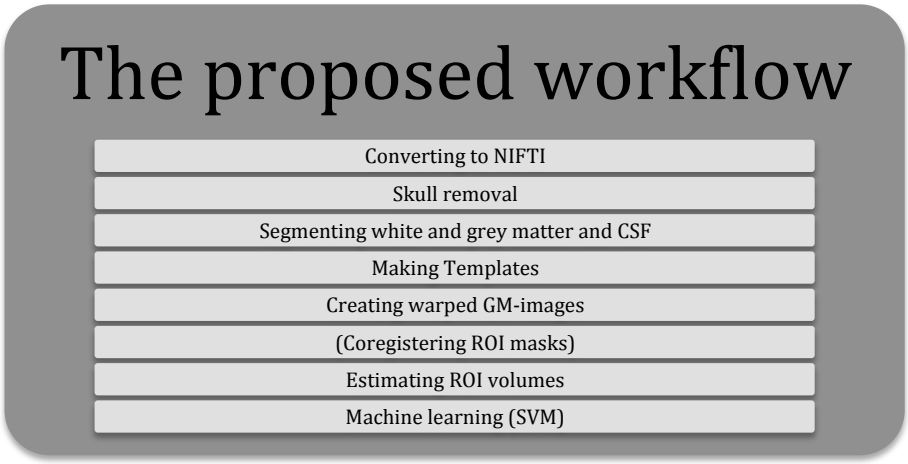

Figure 1: Initially proposed structure for the workflow

3.1 Workflows designed

During the development phase it was found out to be beneficial to break the workflow into several smaller sub-workflows, which could be independently executed. This would provide to be useful both for troubleshooting as well as normal usage, especially if there would be need to process different subgroups from the original data – when using the partial workflows, it's possible to do processing once, and still use the data for multiple different processing and analysis steps.

The tools selected for the workflows were modelled after expert suggestions and literary sources, with the main processing structure modelled after Ashburner's often cited VBM tutorial (32). Excluding the last step - machine learning - all of the steps were executed as a part of a Nipype workflow. In relation to the initially planned workflow setup, step 2 was added as a new step, and step 6 was changed to improve the reliability of the results. Their motivation is presented in the corresponding workflow's descriptions.

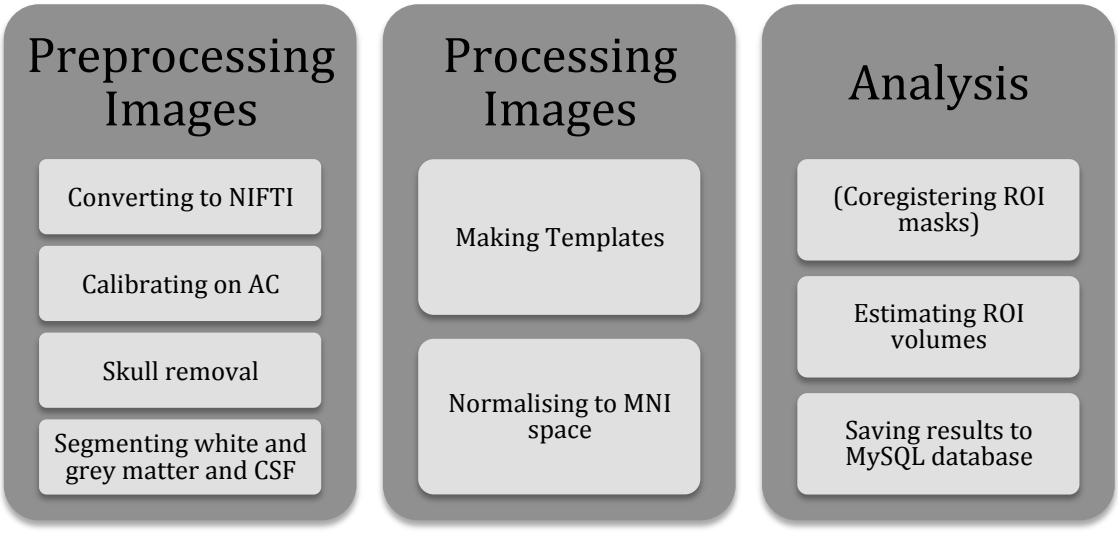

Figure 2: Final structure for the three workflows developed

3.1.1 Image pre-processing workflow

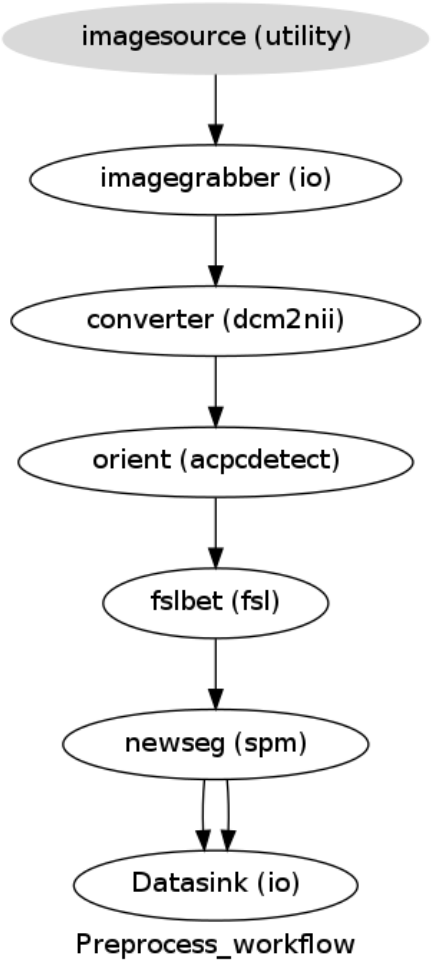

Figure 3: Pre-processing workflow

The pre-processing workflow consists of four main steps. As the workflow processed an individual DICOM image set, so if there are three new image sets, the workflow will be run for three times, once for each image set. In the first processing step (named converter), the subject's DICOM image files are converted into a single NIFTI-1 image file – NIFTI-1 being the file format used throughout the rest of the workflows. In the second step (orient) the images are automatically oriented by setting the origin on anterior commissure, thus providing a fixed point of reference for the following tasks. This step was added to the workflow following a suggestion from the NNI and it was set up using the acpcdetect tool from the ART package (33). Third step of the workflow (fslbet) is the skull removal, which is responsible for cleaning out irrelevant aspects of the image, such as skull, nose, eyes and teeth, and leaving just the intracranial structures, cerebellum and brainstem. In the last pre-processing step the images are segmented into three different image modalities, namely ones containing only grey matter (GM), white matter (WM) or cerebrospinal fluid (CSF) per each new image. These images are then stored to be used as source for the processing workflow.

3.1.2 Image processing workflow

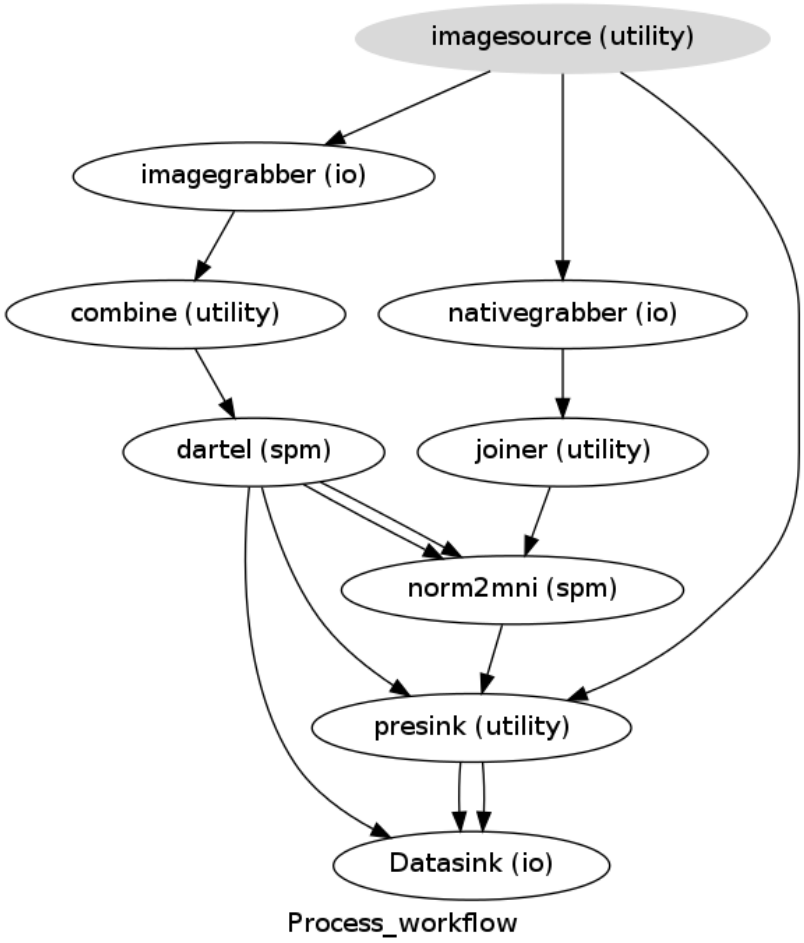

Figure 4: Processing workflow

The processing workflow is centred on the DARTEL process, where a special template is generated from all subject images involved in the execution of the workflow (15). DARTEL also generates so called flow fields, indicating how specific images differ from the template. Both of these data (along with the native grey matter segmentations) are then utilised in the next step, transforming individual's previously segmented grey matter image to normalised MNI-space, MNI standing for Montreal Neurological Institute which defined the MNI brain atlas used by the SPM tool. MNI space gives general coordinate system that enables comparison of brain structures between different images. The motivation for this normalisation is to enable the usage of masks, which are already set in MNI space, in the analysis workflow.

3.1.3 Image co-registration and analysis workflows

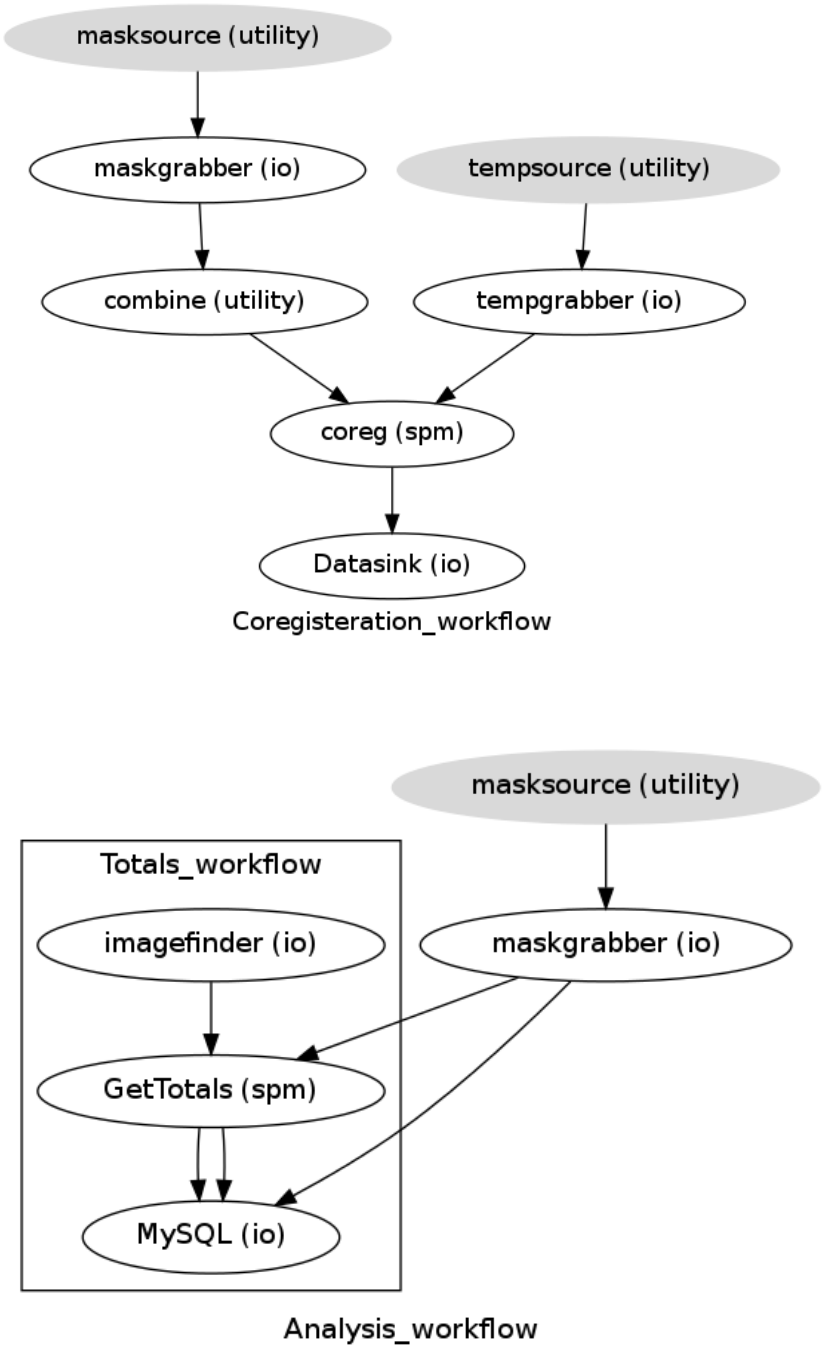

Figure 5: Co-registration and Analysis workflows, including the sub-workflow for iterating over subjects

In the analysis workflow the selected masks are first co-registered to the same resolution as the images being analysed. Then the co-registered masks are used in the analysis_workflow to select areas in the MNI-normalised grey matter images to estimate the amount of grey matter in that area. The totals workflow is executed once per each mask, and each execution is iterated over all the subjects' normalised grey matter images. The estimation of volumes was done by utilising a version of Ged Ridgway's Matlab SPM script get_totals.m which was further developed to work with Nipype and in SPM Standalone environment as past of the

thesis work (21)(Appendix C). The resulting grey matter volume estimates are then stored to the MySQL database table Nipytab as subject, mask, totals –tuple, where *totals* is the volume calculated from *subject* image using the *mask*.

In the MySQL database exists also another table, Nipydiag, which includes subject identifier and diagnostic result (0 – healthy control, 1 – Alzheimer's patient). These results are then combined in a separate script to build a third view, called Nipyview, which lists all masks as columns and just one row per subject with the totals in correct columns, and the diagnostic result as column no 1. This data can then utilised for the machine learning process, which was left out of the analysis workflow as the same data could be used to execute multiple machine learning processed.

2.5 Machine learning

Machine learning generally refers to using special algorithms on a certain type of data to 'learn' the data in question so that when new similar data is presented, the algorithm could classify if the new data is similar to a certain group in the already presented data. The machine learning technique utilised in this thesis is called Support Vector Machine, or SVM, which takes in numeric data and a classification (here a number denoting if the patient has Alzheimer's disease or not) (26). Then the classifier tries to find a way, which could be used to separate these two groups, and it can then utilise the same method to see which group new data would belong into.

Initial processing of the input images through the workflows was supposed to result in a database table, where data from each subject image is expressed with estimated grey matter volumes for each pre-selected mask. This data provides the base for different kinds of machine learning approaches. The data is joined with clinical information (whether patient has Alzheimer's disease or not), and then exported to a CSV file. The subject identifier is removed, and the data is then used to teach an SVM classifier that can in turn be used to review more data and to predict if a new patient (i.e. a set of grey matter volumetric estimates from the selected areas) should be classified as an AD patient or as a healthy control.

The machine learning implementation was planned to be setu up using the SVM capabilities based on LIBSVM included in the Scikit-learn bundle (27–29). However, the utilised machine-learning algorithm could be quite easily interchanged with another, that could use then be taught and testing using the same data as the input material.

2.3 Data collection

During the workflow testing and development phase, the data used was obtained from Alzheimer's Disease Neuroimaging Initiative's ADNI MRI repository (30). The used material, which consisted of structural MR images of healthy controls and Alzheimer patients, was utilized for testing the suitable tools for different operations, and for setting up the pipeline structure and testing suitable machine learning methods. Due to the long processing times involved in some of the workflow steps, most of the development and initial testing of the workflows was executed on a reduced set of ADNI data – demographics reported courtesy of Parvathy Sudhir Pillai (31).

	AD	(22	subjects)	HC	(25	subjects)
	Mean	SD	Range	Mean	SD	Range
Age	75.2	7.4	59-88	75.3	5.2	62-85

Table 1: Demographics of the test data set (a subset of ADNI MRI).

2.6 Ethical considerations

As both the image processing and machine learning steps require real data as inputs to give out relevant results, the work had to be done using real images from real people. In the initial setting-up phase the images used were from ADNI repository that is available for researchers by request (30). The National Neuroscience Institute's own data was supposed to be used for the final testing of the system, but the data was not approved for this use within the time frame available for the thesis project. Thus ADNI data was also utilised for the final testing of the implementation.

The processed data, despite being de-identified by removing personal identification data from the image files, has still some recognizable aspects (like skull structure), and thus special attention was paid so that the data would be only stored in encrypted directories on machines, that only the author had access to, and naturally keeping the computer systems properly up-to-date with latest malware protection tools and operating system security updates. Some parts of the research process (especially the long processing runs) were also conducted remotely, utilising a password-protected, encrypted remote desktop connection within a password-protected, encrypted VPN tunnel.

When considering using the implementation in a research or clinical setting, it should be taken into account that the implementation hasn't been clinically verified or evaluated, and it shouldn't be taken as such – the focus of this thesis work was technical, i.e. setting up an example implementation of the workflow using free and open source tools. More work is needed for testing the actual accuracy and diagnostic correctness of the framework.

3. Results

3.1 Workflow efficiency versus manual execution

	Step	Tool used
1	Convert DICOM files into NIfTI-1	Dcm2nii
2	Image orientation based on anterior commissure	ART Acpcdetect
3	Skull removal	FSL Bet2
4	Segmentation of image into white matter (WM), grey matter (GM) and cerebrospinal fluid (CSF)	SPM NewSegment
5	Create a template from selected of images	SPM DARTEL
6	Normalizing images to Montreal Neurological Institute (MNI) space	SPM DARTEL Norm2MNI
7	Co-registering region of interest (ROI) masks (generated earlier)	SPM Coregister
8	Estimate grey matter volumes in selected regions of interest	SPM Get_totals
9	Storing results to database	MySQL
10	Machine learning from the data	Scikit-learn (LIBSVM)

Table 2: Tools selected for Pre-processing, processing and analysis steps in the final workflows.

The results presented in this chapter were acquired by running a subset of ADNI data including Alzheimer's patients' and healthy controls' magnetic resonance images. Observed results were the lengths of different processing steps and the classifier results from the machine-learning phase. The results are also compared to estimated manual processing times, as reported by Dr. Ming-Ching Wen from the National Neuroscience Institute.

Pre-processing step	O(n) Duration of processing per subject
Conversion of images from DICOM to NIfTI-1	seconds
Orientation on anterior and posterior commissure	seconds
Skull removal	seconds
Segmentation	minutes

Table 3: Pre-processing steps and related durations

The execution of the pre-processing workflow took minutes per subject, with most of the time being spent in the segmentation step. As a comparison, manual processing takes less than a minute per subject for both of the first two steps, while the two last steps take two days for a group of 70 patients. On comparison, automated processing times for a group of 70 patients should be 70 subject * less than 5 minutes/subject = less than 350 minutes, which is under six hours i.e. much less than in the manual condition.

Processing step	O(n) Duration of processing per subject
Creating template	tens of minutes
Normalising to MNI space	minutes

Table 4: Processing steps and related durations

The processing workflow took the longest of all of the workflows, as the generation of template took tens of minutes per subject – for larger processing runs this can amount up to several days. The normalising step is noticeably faster than the template generation. Manual processing for 70 subjects takes about 5 hours for template creation and less than an hour to normalising. This is quite equal to the automatic processing times, as in both cases the same SPM functions are executed just once per each step, and the manual work overhead consists only of selecting the associated input files and initiating the tasks.

Analysis step	O(n) Duration of processing per subject
Co-registering masks	tens of seconds (per mask, i.e. not per subject)
Estimating volumes	seconds per mask
Saving to database	less than a second

Table 5: Analysis steps and related durations

The analysis workflow execution was fast compared to the other workflows, with co-registering taking tens of seconds per mask, while all other steps were executed at or below seconds per subject -scale. When doing the processing manually the co-registering and volume estimation steps both take less than a minute per subject, while saving the values takes tens of seconds per subject. Excluding the co-registration step (which is not scaled by the number of subjects but by the number of used masks), doing the processing using the automated workflow could cut away about half of the processing time, as the processing run for 70 patients would take approx. 30 sec + 5 sec = 35 sec per subject, which is clearly less than up to two minutes taken by the manual process.

3.2 Machine learning results

The machine learning results described are based on a subset of ADNI data (22 advanced Alzheimer's patients and 25 healthy controls), and a set of masks, namely medial temporal gyrus (Left-L & Right-R), entorhinal cortices (L & R), hippocampal formation (L & R), nucleus accumbens (L & R), middle occipital gyrus (L & R), parahippocampal gyrus (L & R), medial temporal gyrus (L & R), inferior frontal gyrus (L & R) and medial fronto-orbital gyrus (L & R). The execution of the machine learning and testing step was fast, in the order of less than a second per subject.

The main machine learning method utilised was SVM classifier, and it was evaluated using 10-fold cross-validation. The results of the fully automated workflow were also compared to the machine learning results from the same initial images and masks executed using a manual method, where the same steps were performed for each of the same subjects one at a time in Matlab environment – including the training and evaluating of the SVM classifier (manual processing results courtesy of Parvathy Sudhir Pillai (31)).

Workflow execution method	Precision AD	Recall AD	Precision HC	Recall HC
Automatic	0.771	0.713	0.792	0.828
Manual	0.786	0.708	0.750	0.764

Table 6: Results of SVM classification for Alzheimer's patients (AD) and healthy controls (HC)

3.3 Analysis

The results presented are based on the subject of ADNI-data mentioned earlier (22 AD patients and 25 healthy controls). Using this dataset to evaluate the performance of the workflow and especially the machine-learning phase can be a bit problematic as a larger dataset would be better. However due to time constrains the results are presented for this smaller dataset.

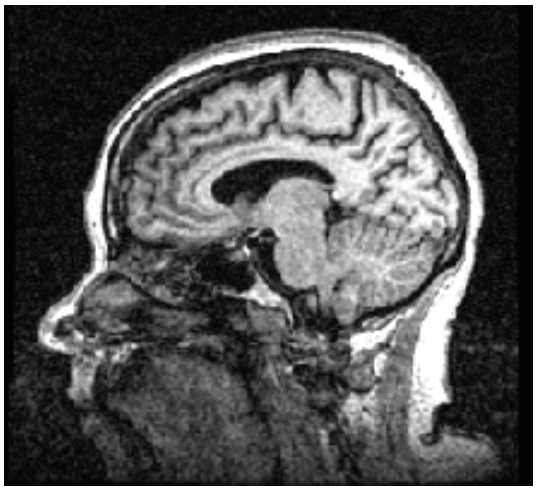

Image 1: Example slice of a MR-image to be processed

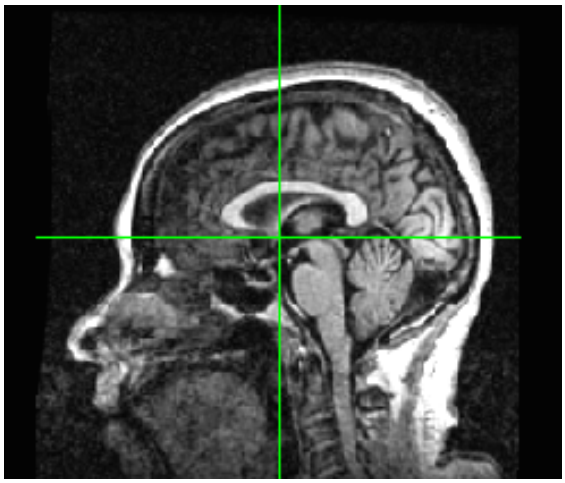

Image 2: Example of image calibrated on anterior commissure

3.3.1 Image pre-processing

During the development and testing phases many ADNI images were used to test different parts of the workflows as well as complete workflows. In pre-processing the images go through a number of different processes. Especially the segmentation step proved to be problematic, as roughly 20% of images seemed to manifest corrupted grey matter segmentation results. In the example dataset, 7 AD patients' and 4 healthy controls' images were corrupted and removed after pre-processing workflow before running the image processing workflow. The reason for this might be related to the fact that ADNI data contains

images taken with different MRI scanners and with different settings (resolution, spacing, etc.). This would require more detailed attention to be paid to map out the problem so that the subjects' images could be executed throughout all workflows without the need for manual quality control.

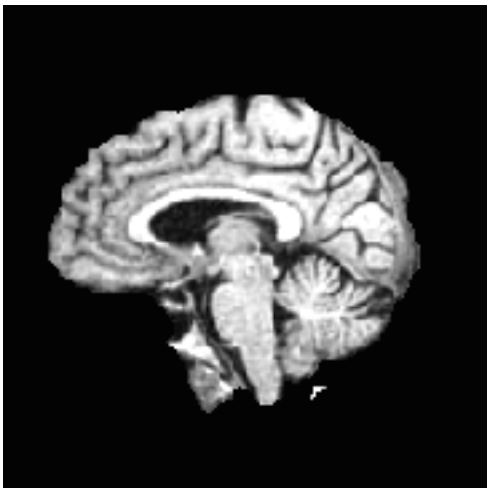

Image 3: Example of extracted brain (i.e. after skull removal)

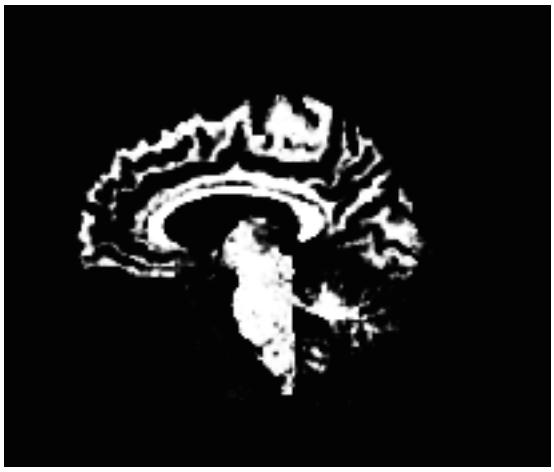

Image 5: Example of normal white matter segmentation result

Image 4: Example of normal grey matter segmentation result.

Image 6: Example of corrupted grey matter segmentation result.

3.3.2 Image processing

Image processing workflow is the most calculation-intensive from all of the workflows, with most of the time in the generation of DARTEL templates. This single step in the workflow could take for many hours when processing larger number of images, as building the template is iterated many times over. The normalisation of subject's grey matter images to MNI space is based on the flowfields generated in the previous step.

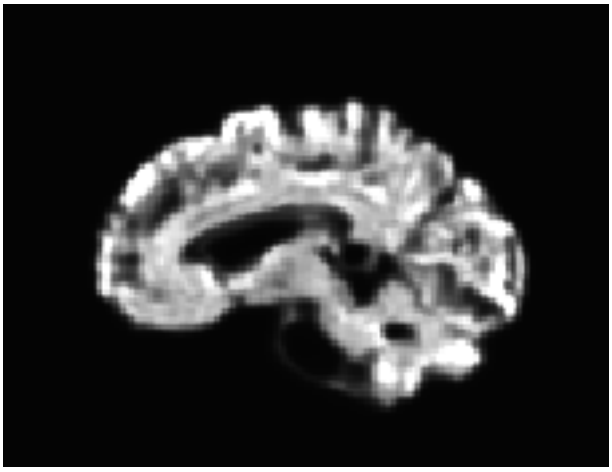

Image 7: Example of template image

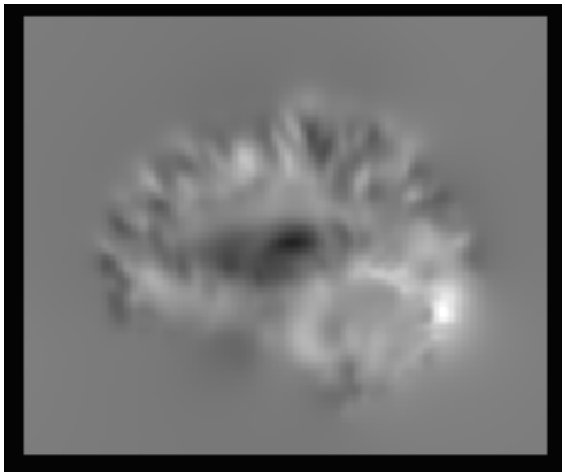

Image 8: Example of subject's flowfield

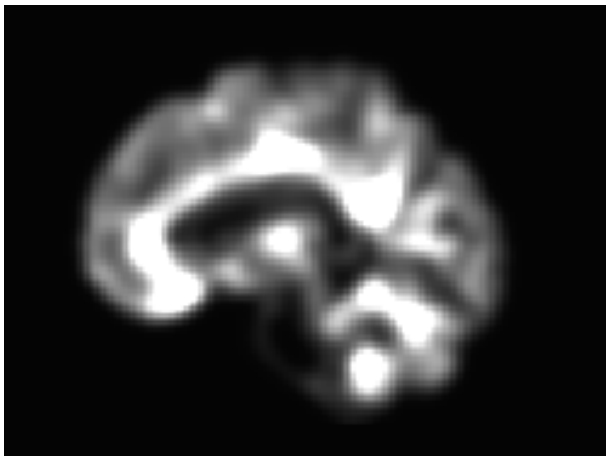

Image 9: Example of normalised GM image

3.3.3 Image analysis

The analysis workflow is the fastest of the workflows. At the end of the workflow, the MySQL database is populated with the data of grey matter volumes for each mask per each subject's masked area. Additional script is then required to run to generate the MySQL view where the data is arranged in a format suitable for the machine-learning tool. This view then needs to be exported by using a third script. These are attached to the report, but not yet automatically executed from the workflow (Appendix D, E).

3.3.4 Machine learning

The obvious finding from the machine learning step is the huge difference between manual and automatic processing results, which shouldn't be possible as the same steps were supposed to be carried out in the same way in both occasions. Thus it is reasonable to suspect that in one of the methods, some step was carried out differently or perhaps some default parameter of one of the used programs was different between the environments so that even when all of the entered parameters were the same, the result was different due to different

defaults. This discrepancy calls out for more detailed observation and comparison of the automatic and manual processing steps and results, and casts an uncertainty shadow over the whole automatic pipeline implementation if the accuracy of the processing cannot be trusted.

3.4 Other contributions from the thesis project

The months doing the thesis work at Medical Computing Laboratory amounted up to more than just this thesis report. An example environment for processing the MR-images was developed, consisting of the used tools, workflows, and other related scripts for database and web front end. Although not fully completed, the example environment was outlined in a set of rough setup scripts and instructions which could be used to build new test or development instances, for example on virtual machines. A conference paper on the framework was also written for the APAMI 2014 conference by the author in collaboration with Parvathy Sudhir Pillai and Tze-Yun Leong – not yet published (31).

While the thesis work utilised available free open source tools, there were some requirements in the planned workflow that were not possible to achieve with the available tools as such. To meet this need, additions and alterations were made for the source code of some of the tools, and also some new interfaces and scripts were constructed. Selected ones of these were offered back to the community via related discussion forums and GitHub *pull requests*. By the time of writing this, some additions have already been approved and included into the official Nipype codebase and distribution (34). Detailed list of all codes and scripts generated during the thesis process can be found at the appendixes at the end of this thesis report, and some of the codes are also planned to be made publicly available to download at the thesis GitHub project folder (25).

4. Discussion

4.1 Discussion on methods

The method of this thesis was exploratory feasibility study, and thus most of the time was spent setting up and developing the test environment including the software setup and script and workflow design to see if it would be possible to set up a working solution within the time period. This part of the process took much more time than initially anticipated, and thus the actual testing of the environment was done hastily at the last moments of the project. The user interface, which was initially considered to be one major part of the project, was also diminished to just a simple and quite limited proof-of-concept implementation due to time constrains with the project and as such it was mainly left outside the scope of this thesis report.

The selection of Linux environment as the development environment was encouraged due to the fact, that all of the tools used in the workflows were available as ready compiled packages in Linux. Most of the tools could also be utilised on OSX, but the setup process might require more time, and some of the tools didn't have Windows based distributions available. Also, by utilising Linux also the operating system environment was consistent with the *free and open source* ideology that was one of the main aspects related to the research questions in this thesis. Also, the usage NIfTI-1 file format instead of DICOM for workflow processing was essential as some of the free and open source tools utilised in the workflows handle only the NIfTI-1 file format.

Some of the main problems faced with the environment setup were possibly due to the fact that the author was not previously deeply experienced with Matlab, Python, neuroscience, machine learning nor MR-image processing beyond brief introduction to some of the topics – so the whole process can be said to have been a continuous learning experience. Still, taking this into consideration, the approach of using free and open source tools can be considered feasible, as it turned out to be doable even with limited prior knowledge in the field and in the tools and technologies utilised.

4.1.1 Limitations

The problems with the initial set up of the technical tools and the workflows limited the thesis project approach towards simple prototyping setup, leaving out certain aspects, which could have been interesting to pursue in more detail. The same applies to the machine-learning dimension of the project, which also manifested in a simple proof-of-concept implementation, limiting the possible results that could have been acquired if there would have been more time to set up multiple different machine learning techniques to be tested and compared. More attention should also be paid for selecting properly validated and diagnostically relevant masks for the volumetric estimation as this data forms the base on which the machine learning processes function on.

4.2 Existing solutions

In the last years there have been many published articles, where MR-images are segmented and used for SVM classification. The existing work in the area include, for example, papers by Klöppel et al. and Magnin et al., which utilised quite similar approaches but reported much higher SVM classifier results in the area of 90% accuracy (35,36). However, even as the methods used in this area are usually outlined in the articles in a general level, the results might not be easily replicated because of other evaluators not having access to the same data or to all of the tools and especially all of the parameters used. The novelty in this thesis is more in the approach of making all the parts of the process open (including the scripts and setting used in the thesis work) to enable transparency and to make it possible to re-use and build upon the work done in the thesis project.

One interesting approach in this field is presented by Miller et al. in their recent paper in Frontiers in Neuroinformatics (37). Their approach, called BrainClould, uses special BrainGPS method with DiffeoMaps to enable multimodal structural mapping over different imaging modalities. They also describe using large representation index in the order of 1000-10.000 dimensions per patient to enable the usage of multiple machine learning techniques including Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA).

PCA was also utilised by Nika et al. in their recently published conference paper about Eigen-Block Change Detection algorithm (EigenBlockCD) (38). Their system is best suited for comparing subjects longitudinal images (MRI images taken from the same person in different times) and calculating the differences of the images directly without requiring any additional segmentation or preprocessing steps while also taking into account the possible differences with patient positioning etc. This direct processing method removes the possible bias or other artifacts or errors that can be produced by taking an image through multiple sequential processing steps, and their initial results from testing the EigenBlockCD algorithm are impressive, as it yielded a mean 99.9% correct classification, with 75.6% sensitivity, and 100% specificity in a study of 48 images.

Both of the aforementioned approaches seem very interesting and are definitely something worth keeping eyes on.

4.3 Future development

It would be interesting to further develop this initial implementation set up in the thesis process. The product could offer user-friendly web frontend for importing data, selecting and automating processing steps from multiple different options and routing that data for a number of machine learning tools. In this way it would be possible and relatively easy to, for example, compare the predictive values of different masks, or the effects of different processing parameters. Other machine learning tools could also be conjoined, for example using AdaBoost with some weak learner tools, or PCA and LDA as they seem to have been utilised in this field for high quality results. Towards this end, the example codes developed in the thesis project are published freely to enable end user creativity by helping users in trying out new novel combinations easily – something that might not happen when setting up the environment itself requires a lot of work, time and pre-knowledge. The codes as well as this thesis report are planned to be distributed via GitHub for free for anyone to fix, modify, rip,

criticize and build upon (25). One really interesting potential approach would be to try to add genetic meta-algorithms to the system to automate the testing of different variations of data processing and classification tools, thus making an approach which could partly automate the possibility of finding out novel ways of processing the data while getting confirmedly better results.

5. Conclusion

This thesis work aimed to find out how feasible it would be to utilise available free and open source tools to automate processing of MR-images, and to find out is the resulting solution could be useful for dementia related diagnosis and research, focusing mainly on Alzheimer's disease in this case.

The results show that implementing such a workflow is possible and the resulting setup can be used to process MR images and to do machine learning on the data extracted from the images. The work also highlights some obstacles and possible problems with the approach – there is still a need for human intervention to evaluate the setup and possibly double check some results before they are processed further, as the number of failed segmentation results show. Also, setting up the test environment and related workflows takes time and required both programming and content skills. Thus it is the hope of the author, that by publishing the developed codes and scripts, others might be able to build on that work, and to correct and expand it, instead of doing the same initial mistakes again.

The approach itself has been proved feasible, and faster than the manual method. To be useful and reliable for diagnostic or research purposes, the produced environment needs still more work, including proper validation of all the steps in the workflow, better user interface and more sophisticated programming implementation which would remove some of the *ugly hacks* still present in the current version. Furthermore, the accuracy and predictive powers of the current machine-learning phase based on the results from the workflow is too low to be beneficial in the current state of the implementation. The author suspects that there is some fault still present in the example pipeline that causes the data to be biased, misaligned or somehow corrupted, as other similar approaches reported in the literature achieved much better results. Thus, the current technical implementation is feasible but the practical workflow implementation and the results it provides are not beneficial. However, with more time and effort, it should be possible to fix the workflow to achieve useful results, and to improve the user interface to make the system user-friendly and easy to use.

References

1. Institute for Health Metrics and Evaluation. The Global Burden of Disease: Generating Evidence, Guiding Policy [Internet]. Seattle: IHME; 2013. Available from: http://www.healthmetricsandevaluation.org/sites/default/files/policy_report/2011/GBD_Generating Evidence_Guiding Policy FINAL.pdf

2. Prince M, Bryce R, Albanese E, Wimo A, Ribeiro W, Ferri CP. The global prevalence of dementia: a systematic review and metaanalysis. Alzheimers Dement [Internet]. Elsevier Ltd; 2013 Jan [cited 2014 Jan 23];9(1):63–75.e2. Available from: http://www.ncbi.nlm.nih.gov/pubmed/23305823

3. Wimo A, Jönsson L, Bond J, Prince M, Winblad B. The worldwide economic impact of dementia 2010. Alzheimers Dement [Internet]. 2013 Jan [cited 2014 Feb 4];9(1):1–11.e3. Available from: http://www.ncbi.nlm.nih.gov/pubmed/23305821

4. Ferri CP, Prince M, Brayne C, Brodaty H, Fratiglioni L, Ganguli M, et al. Global prevalence of dementia: a Delphi consensus study. Lancet [Internet]. 2005 Dec 17 [cited 2014 Jan 22];366(9503):2112–7. Available from: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2850264&tool=pmcentrez&rendertype=abstract

5. WHO. The ICD-10 classification of mental and behavioural disorders: diagnostic criteria for research [Internet]. Geneva: World Health Organization; 1993 [cited 2014 Feb 10]. Available from: http://www.who.int/classifications/icd/en/GRNBOOK.pdf

6. Boise L, Camicioli R, Morgan DL, Rose JH, Congleton L. Diagnosing Dementia : Perspectives of Primary Care Physicians. Gerontologist. 1999;39(4):457–64.

7. Turner S, Iliffe S, Downs M, Wilcock J, Bryans M, Levin E, et al. General practitioners' knowledge, confidence and attitudes in the diagnosis and management of dementia. Age Ageing [Internet]. 2004 Sep [cited 2014 Feb 10];33(5):461–7. Available from: http://www.ncbi.nlm.nih.gov/pubmed/15271637

8. Karas G, Scheltens P, Barkhof F. Feature extraction and strategy of analyzing structural neuroimaging in dementia. Handb Clin Neurol [Internet]. 2008 Jan;89:75–86. Available from: http://www.ncbi.nlm.nih.gov/pubmed/18631732

9. Mikheev A, Nevsky G, Govindan S, Grossman R, Rusinek H. Fully automatic segmentation of the brain from T1-weighted MRI using Bridge Burner algorithm. J Magn Reson Imaging [Internet]. 2008 Jun [cited 2014 Jan 22];27(6):1235–41. Available from: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3840426&tool=pmcentrez&rendertype=abstract

10. Heckemann R a, Hajnal J V, Aljabar P, Rueckert D, Hammers A. Automatic anatomical brain MRI segmentation combining label propagation and decision fusion. Neuroimage [Internet]. 2006 Oct 15 [cited 2014 Jan 23];33(1):115–26. Available from: http://www.ncbi.nlm.nih.gov/pubmed/16860573

11. Heckemann R a, Hammers A, Rueckert D, Aviv RI, Harvey CJ, Hajnal J V. Automatic volumetry on MR brain images can support diagnostic decision making. BMC Med Imaging [Internet]. 2008 Jan [cited 2014 Jan 22];8:9. Available from: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2413211&tool=pmcentrez&rendertype=abstract

12. Gorgolewski K, Burns CD, Madison C, Clark D, Halchenko YO, Waskom ML, et al. Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in python. Front Neuroinform [Internet]. 2011 Jan [cited 2014 Jan 15];5(August):13. Available from: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3159964&tool=pmcentrez&rendertype=abstract

13. Halchenko YO, Hanke M. Open is Not Enough. Let's Take the Next Step: An Integrated, Community-Driven Computing Platform for Neuroscience. Front Neuroinform [Internet]. 2012 Jan [cited 2014 Mar 19];6(June):22. Available from: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3458431&tool=pmcentrez&rendertype=abstract

14. Litvak V, Mattout J, Kiebel S, Phillips C, Henson R, Kilner J, et al. EEG and MEG data analysis in SPM8. Comput Intell Neurosci [Internet]. 2011 Jan [cited 2014 Jan 21];2011:852961. Available from: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3061292&tool=pmcentrez&rendertype=abstract

15. Ashburner J. Computational anatomy with the SPM software. Magn Reson Imaging [Internet]. Elsevier Inc.; 2009 Oct [cited 2014 Mar 20];27(8):1163–74. Available from: http://www.ncbi.nlm.nih.gov/pubmed/19249168

16. SPM. SPM/Standalone [Internet]. 2013 [cited 2014 Feb 14]. Available from: http://en.wikibooks.org/wiki/SPM/Standalone

17. Ashburner J. A fast diffeomorphic image registration algorithm. Neuroimage [Internet]. 2007 Oct 15 [cited 2014 Jan 23];38(1):95–113. Available from: http://www.ncbi.nlm.nih.gov/pubmed/17761438

18. Jenkinson M, Pechaud M, Smith S. BET2: MR-based estimation of brain, skull and scalp surfaces. Eleventh annual meeting of the organization for human brain mapping. 2005.

19. Rorden C, Karnath H-O, Bonilha L. Improving lesion-symptom mapping. J Cogn Neurosci [Internet]. 2007 Jul;19(7):1081–8. Available from: http://www.ncbi.nlm.nih.gov/pubmed/17583985

20. Maldjian JA, Laurienti PJ, Kraft RA, Burdette JH. An automated method for neuroanatomic and cytoarchitectonic atlas-based interrogation of fMRI data sets. Neuroimage [Internet]. 2003 Jul [cited 2014 Mar 22];19(3):1233–9. Available from: http://linkinghub.elsevier.com/retrieve/pii/S1053811903001691

21. Ridgway G. get_totals.m [Internet]. 2007. Available from: http://www0.cs.ucl.ac.uk/staff/g.ridgway/vbm/get_totals.m

22. Django [Internet]. Available from: https://djangoproject.com

23. Ravindran A. Building a blog in 30 mins with Django (Screencast HD) [Internet]. YouTube. 2012. Available from: http://www.youtube.com/watch?v=srHZoj3ASmk

24. Palén A. minimal-django-file-upload-example [Internet]. GitHub; 2013. Available from: https://github.com/axelpale/minimal-django-file-upload-example

25. Andberg S. NipyUbu - codes for master's thesis [Internet]. 2014. Available from: https://github.com/andsam/dsv_thesis

26. Hearst M, Dumais S, Osman E. Support vector machines. … Syst their … [Internet]. 1998 [cited 2014 May 15]; Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=708428

27. Chang C-C, Lin C-J. LIBSVM: A Library for Support Vector Machines. ACM Trans Intell Syst Technol [Internet]. 2011 Apr 1 [cited 2014 Mar 19];2(3):1–27. Available from: http://dl.acm.org/citation.cfm?doid=1961189.1961199

28. Pedregosa F, Varoquaux G. Scikit-learn: Machine learning in Python. … Mach Learn … [Internet]. 2011 [cited 2014 Apr 12];12:2825–30. Available from: http://dl.acm.org/citation.cfm?id=2078195

29. MySQL [Internet]. Available from: http://www.mysql.com

30. Jack CR, Bernstein MA, Fox NC, Thompson P, Alexander G, Harvey D, et al. The Alzheimer's Disease Neuroimaging Initiative (ADNI): MRI methods. J Magn Reson Imaging [Internet]. 2008 Apr [cited 2014 Feb 10];27(4):685–91. Available from: http://onlinelibrary.wiley.com/doi/10.1002/jmri.21049/full

31. Andberg S, Pillai PS, Leong TY. MRI Image Processing Workbench for Alzheimer's Disease Classification. Indian J Med Informatics. 2014;

32. Ashburner J. VBM Tutorial [Internet]. 2010. p. 1–14. Available from: http://www.fil.ion.ucl.ac.uk/~john/misc/VBMclass10.pdf

33. Ardekani BA, Bachman AH. Model-based Automatic Detection of the Anterior and Posterior Commissures on MRI Scans. Neuroimage. 2009;46(3):677–82.

34. Ghosh SS, Gorgolewski CF. Nipype source code on GitHub [Internet]. Available from: https://github.com/nipy/nipype

35. Klöppel S, Stonnington CM, Chu C, Draganski B, Scahill RI, Rohrer JD, et al. Automatic classification of MR scans in Alzheimer's disease. Brain [Internet]. 2008 Mar [cited 2014 Apr 30];131(Pt 3):681–9. Available from: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2579744&tool=pmcentrez&rendertype=abstract

36. Magnin B, Mesrob L, Kinkingnéhun S, Pélégrini-Issac M, Colliot O, Sarazin M, et al. Support vector machine-based classification of Alzheimer's disease from whole-brain anatomical MRI. Neuroradiology [Internet]. 2009 Feb [cited 2014 Mar 28];51(2):73–83. Available from: http://www.ncbi.nlm.nih.gov/pubmed/18846369

37. Miller MI, Faria A V, Oishi K, Mori S. High-throughput neuro-imaging informatics. Front Neuroinform [Internet]. 2013 Jan [cited 2014 Mar 24];7(December):31. Available from: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3865387&tool=pmcentrez&rendertype=abstract

38. Nika V, Babyn P, Zhu H. Change detection of medical images using dictionary learning techniques and PCA. Proc SPIE [Internet]. 2014. p. 903506–14. Available from: http://dx.doi.org/10.1117/12.2038751

Appendices

Appendix A – Adding acpcdetect-interface for Nipype

The new interface added to local nipype copy for utilizing the acpcdetect –tool from ART. Code available at: https://github.com/andsam/nipype/blob/thesis_additions/nipype/interfaces/acpcdetect.py

nipype/interfaces/acpcdetect.py

```
...  ...  @@ -0,0 +1,45 @@
     1    +from nipype.interfaces.base import (
     2    +    TraitedSpec,
     3    +    CommandLineInputSpec,
     4    +    CommandLine,
     5    +    File
     6    +)
     7    +from nipype.interfaces.traits_extension import isdefined
     8    +import os
     9    +
    10    +class ACPCInputSpec(CommandLineInputSpec):
    11    +    infile = File(desc="Infile", exists=True, mandatory=True, argstr='-i %s')
    12    +    outname = File(genfile=True, desc="outfile", position= -1, argstr='-o %s')
    13    +
    14    +class ACPCOutputSpec(TraitedSpec):
    15    +    outfile = File(desc = "Outfile", exists = True)
    16    +
    17    +class ACPCDetect(CommandLine):
    18    +    input_spec = ACPCInputSpec
    19    +    output_spec = ACPCOutputSpec
    20    +    cmd = 'acpcdetect'
    21    +
    22    +    def _list_outputs(self):
    23    +        outputs = self.output_spec().get()
    24    +        outputs['outfile'] = self.inputs.outname
    25    +        if not isdefined(outputs['outfile']):
    26    +            outputs['outfile'] = os.path.abspath(self._gen_outfilename())
    27    +        else:
    28    +            outputs['outfile'] = os.path.abspath(outputs['outfile'])
```

```
29 +        return outputs
30 +
31 +    def _gen_filename(self, name):
32 +        if name is 'outname':
33 +            return self._gen_outfilename()
34 +        else:
35 +            return None
36 +
37 +    def _gen_outfilename(self):
38 +        name = os.path.basename(self.inputs.infile)
39 +        return 'ac_' + name
40 +
41 +if __name__ == '__main__':
42 +
43 +    acd = ACPCDetect(infile='an_existing_file')
44 +    print acd.cmdline
45 +    acd.run()
```

Appendix B – Adding modulate-parameter for CreateWarped interface in Nipype

Listing of changes implemented for the interfaces/spm/preprocess.py and contributed to the main Nipype codebase via the Nipype GitHub pull request #827:
https://github.com/nipy/nipype/pull/827/commits

nipype/interfaces/spm/preprocess.py

```
@@ -1084,6 +1084,8 @@ class CreateWarpedInputSpec(SPMCommandInputSpec):
                                   field='crt_warped.K')
     interp = traits.Range(low=0, high=7, field='crt_warped.interp',
                           desc='degree of b-spline used for interpolation')
+    modulate = traits.Bool(field='crt_warped.jactransf',
+                           desc="Modulate images")

 class CreateWarpedOutputSpec(TraitedSpec):
@@ -1127,8 +1129,12 @@ def _list_outputs(self):
         outputs['warped_files'] = []
         for filename in self.inputs.image_files:
             pth, base, ext = split_filename(filename)
-            outputs['warped_files'].append(os.path.realpath('w%s%s' % (base,
-                                                                       ext)))
+            if isdefined(self.inputs.modulate) and self.inputs.modulate:
+                outputs['warped_files'].append(os.path.realpath('mw%s%s' % (base,
+                                                                            ext)))
+            else:
+                outputs['warped_files'].append(os.path.realpath('w%s%s' % (base,
+                                                                           ext)))
         return outputs
```

These additions were made early in the thesis process, when the workflow still utilized CreateWarped method instead of DARTEL Norm2MNI which replaced it in later versions (as the masks are already in MNI space).

Appendix C – Adding get_totals -tool for Nipype

The get_totals –tool is a modification from Ged Ridgway's get_totals.m, which is available from: http://www0.cs.ucl.ac.uk/staff/g.ridgway/vbm/get_totals.m

The original script is a Matlab function, and as such not usable with the free SPM Standalone version, so the script was modified and implemented as a Nipype SPM tool to provide same functionality also on SPM Standalone with the added benefit of becoming usable as part of a workflow. The version below also includes some extra functionality to make workflow processing more convenient. Code can be found at: https://github.com/andsam/nipype/blob/thesis_additions/nipype/interfaces/spm/model.py

Additions to nipype/interfaces/spm/model.py

```
@@ -230,6 +230,86 @@ def _list_outputs(self):
230 230         return outputs
231 231
232 232
    233 +class GetTotalsModelInputSpec(SPMCommandInputSpec):
    234 +    in_image = File(exists=True, desc='moludated warped image to analyse',
                            copyfile=False, mandatory=True)
    235 +    in_mask = File(exists=True, desc='binary mask to use', copyfile=False,
                           mandatory=False)
    236 +    #threshold still not implemented
    237 +
    238 +class GetTotalsModelOutputSpec(TraitedSpec):
    239 +    total_volume = traits.Float()
    240 +    subject = traits.Str()
    241 +    mask = traits.Str()
    242 +
    243 +class GetTotals(SPMCommand):
    244 +    """Uses Ged Ridgway's get_totals.m -based script to estimate
         the volume of matter in masked area of image
    245 +
    246 +    http://www0.cs.ucl.ac.uk/staff/g.ridgway/vbm/get_totals.m
    247 +
    248 +    Examples
    249 +    --------
    250 +    >>> gett = GetTotals()
    251 +    >>> gett.inputs.in_image = 'structural.nii'
    252 +    >>> gett.inputs.in_mask = 'mask.nii'
    253 +    >>> out = gett.run() # doctest: +SKIP
    254 +    >>> print out.outputs.total_volume # doctest: +SKIP
    255 +    """
```

```
256 +
257 +    input_spec = GetTotalsModelInputSpec
258 +    output_spec = GetTotalsModelOutputSpec
259 +
260 +    def _make_matlab_command(self, _):
261 +        """validates spm options and generates job structure
262 +        """
263 +        script = "% generated by nipype.interfaces.spm\n"
264 +        script += "% based on Ged Ridgway's get_totals.m -
                        http://www0.cs.ucl.ac.uk/staff/g.ridgway/vbm/get_totals.m\n"
265 +        script += "files = '%s';\n" % self.inputs.in_image
266 +        if isdefined(self.inputs.in_mask):
267 +            script += "msk = '%s';\n" % self.inputs.in_mask
268 +        script += """
269 +if ( ~exist('thr', 'var') || isempty(thr) )
270 +    thr = -inf; % default to include everything (except NaNs)
271 +end
272 +if ~exist('msk', 'var')
273 +    msk = 1; % default to include everything
274 +end
275 +
276 +if ischar(msk)
277 +    msk = spm_vol(msk);
278 +end
279 +if isstruct(msk)
280 +    msk = spm_read_vols(msk);
281 +end
282 +msk = msk ~= 0;
283 +
284 +vols = spm_vol(files);
285 +N = length(vols);
286 +
287 +t = zeros(N,1);
288 +for n = 1:N
289 +    vsz = abs(det(vols(n).mat));
290 +    img = spm_read_vols(vols(n));
291 +    img = img .* msk;
292 +    t(n) = sum(img(img > thr)) * vsz / 1000; % vsz in mm^3 (= 0.001 ml)
293 +end
294 +
295 +fprintf('totals = %f\\n',t);
```

```
296 +"""
297 +        return script
298 +
299 +    def aggregate_outputs(self, runtime=None):
300 +        outputs = self._outputs()
301 +        for line in runtime.stdout.split('\n'):
302 +            if line.startswith("totals = "):
303 +                setattr(outputs, 'total_volume', float(line[len("totals = "):].strip()
304 +        setattr(outputs, 'subject',
                    os.path.splitext(os.path.basename(self.inputs.in_image))[0])
305 +        setattr(outputs, 'mask',
                    os.path.splitext(os.path.basename(self.inputs.in_mask))[0])
306 +        return outputs
307 +
308 +    def _list_outputs(self):
309 +        outputs = self._outputs().get()
310 +        return outputs
311 +
312 +
233 313  class EstimateContrastInputSpec(SPMCommandInputSpec):
234 314      spm_mat_file = File(exists=True, field='spmmat',
235 315                          desc='Absolute path to SPM.mat',
```

(some line brakes added for the code to fit to the word document width)

Also needs ‘GetTotals, ’ to be added into a row in file: nipype/interfaces/spm/__init__.py

```
@@ -7,7 +7,7 @@
7  7  from .preprocess import (SliceTiming, Realign, Coregister, Normalize, Segment,
8  8                          Smooth, NewSegment, DARTEL, DARTELNorm2MNI,
9  9                          CreateWarped, VBMSegment)
10    -from .model import (Level1Design, EstimateModel, EstimateContrast, Threshold,
   10 +from .model import (Level1Design, GetTotals, EstimateModel, EstimateContrast, Threshold
11 11                      OneSampleTTestDesign, TwoSampleTTestDesign,
12 12                      PairedTTestDesign, MultipleRegressionDesign)
13 13  from .utils import (Analyze2nii, CalcCoregAffine, ApplyTransform, Reslice,
```

Currently these modifications related to GetTotals are not included into the main Nipype codebase as the suggestion from developers was to write the whole Matlab code in Python - which would make it faster to execute - but which was considered a secondary objective as far as the thesis process is concerned. Due to time constrains the conversion of the algorithm to python was left out as attention was focused on more pressing issues to get the project finished within required timeframe.

Appendix D – Scripts for generating MySQL tables and views

Shell script to make the SQL tables

```
#!/bin/bash
#to create the database for Nipype SVM mining   @20140416/SA
MYSQL=`which mysql`
MKDATAB="nipybas"
MKTABLE="nipytab"
MKUSER="nipyuser"
MKPASS="nipypass"
Q1="CREATE DATABASE iF NOT EXISTS $MKDATAB;"
Q2="GRANT USAGE ON *.* TO $MKUSER@localhost IDENTIFIED BY '$MKPASS';"
Q3="GRANT ALL PRIVILEGES ON $MKDATAB.* TO $MKUSER@localhost;"
Q4="FLUSH PRIVILEGES;"
Q5="USE $MKDATAB;"
Q6="CREATE TABLE $MKTABLE (id INT NOT NULL AUTO_INCREMENT PRIMARY KEY, generated TIMESTAMP,
subject varchar(100), mask varchar(100), totals FLOAT);"
SQL="${Q1}${Q2}${Q3}${Q4}${Q5}${Q6}"
$MYSQL -uroot -p -e "$SQL"
```

Shell script to make the SQL views (to be run after the table has been populated). Please note, that a separate SUM(CASE –line is needed for all masks used in the analysis. Generating this file automatically from within the user interface based on selected masks would be encouraged, but was left out at this stage due to time constrains.

```
#!/bin/bash
#to create the database views for Nipype SVM mining   @20140417/SA
MYSQL=`which mysql`
MKDATAB="nipybas"
MKTABLE="nipytab"
MKDIAG="nipydiag"
MKUSER="nipyuser"
MKPASS="nipypass"
MKVIEW="nipyview"
Q1="USE $MKDATAB;"
Q2="CREATE TABLE $MKDIAG (id INT NOT NULL AUTO_INCREMENT PRIMARY KEY, generated TIMESTAMP,
subject varchar(100) NOT NULL, disease int);"
Q3="DROP VIEW $MKVIEW;"
Q4="CREATE VIEW $MKVIEW as SELECT d.disease, d.subject, "\
"SUM(CASE t.mask WHEN 'rec28' THEN t.totals END) as ec28, "\
"SUM(CASE t.mask WHEN 'rec34' THEN t.totals END) as ec34, "\
"SUM(CASE t.mask WHEN 'rhfl' THEN t.totals END) as hfl, "\
"SUM(CASE t.mask WHEN 'rhfr' THEN t.totals END) as hfr, "\
"SUM(CASE t.mask WHEN 'rifgl' THEN t.totals END) as ifgl, "\
"SUM(CASE t.mask WHEN 'rifgr' THEN t.totals END) as ifgr, "\
"SUM(CASE t.mask WHEN 'rmefogl' THEN t.totals ELSE NULL END) as mefogl, "\
"SUM(CASE t.mask WHEN 'rmefogr' THEN t.totals ELSE NULL END) as mefogr, "\
"SUM(CASE t.mask WHEN 'rmetg21' THEN t.totals ELSE NULL END) as metg21, "\
"SUM(CASE t.mask WHEN 'rmiogl' THEN t.totals ELSE NULL END) as miogl, "\
"SUM(CASE t.mask WHEN 'rmiogr' THEN t.totals ELSE NULL END) as miogr, "\
"SUM(CASE t.mask WHEN 'rmitgl' THEN t.totals ELSE NULL END) as mitgl, "\
"SUM(CASE t.mask WHEN 'rmitgr' THEN t.totals ELSE NULL END) as mitgr, "\
"SUM(CASE t.mask WHEN 'rnal' THEN t.totals ELSE NULL END) as nal, "\
"SUM(CASE t.mask WHEN 'rnar' THEN t.totals ELSE NULL END) as nar, "\
"SUM(CASE t.mask WHEN 'rphgl' THEN t.totals ELSE NULL END) as phgl, "\
"SUM(CASE t.mask WHEN 'rphgr' THEN t.totals ELSE NULL END) as phgr "\
"FROM $MKDIAG d INNER JOIN $MKTABLE t "\
"ON d.subject = t.subject "\
"GROUP BY d.subject;"
SQL="${Q1}${Q3}${Q4}"
$MYSQL -uroot -p -e "$SQL"
```

Appendix E – Modifications for the SVM integration

Shell script for exporting the data from MySQL and removing the subject number column:

```
#!/bin/bash
#to export data from SQL to CSV for Nipype SVM mining   @20140417/SA

MYSQL=`which mysql`
MKDATAB="nipybas"
MKTABLE="nipytab"
MKDIAG="nipydiag"
MKUSER="nipyuser"
MKPASS="nipypass"
MKVIEW="nipyview"
CSVFILE="/tmp/results.csv"
LOCALFILE="results.csv"

Q1="USE $MKDATAB;"
Q2="SELECT * FROM $MKVIEW INTO OUTFILE '$CSVFILE' FIELDS TERMINATED BY ',' LINES
TERMINATED BY '\\n';"

SQL="${Q1}${Q2}"

$MYSQL -uroot -p -e "$SQL"

cp $CSVFILE $LOCALFILE
#and to remove subjects (column no 2)
cut -d, -f-1,3- $LOCALFILE
```

Python code for testing the svm functionality:

```
import numpy as np
from sklearn import svm
from sklearn import metrics
from sklearn.cross_validation import train_test_split

data_in = np.loadtxt('results.csv',delimiter=',')
X = data_in[:, 2:]
#X = data_in[:, data_in.str(",",3):]
y = data_in[:, 0]
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.1)

clf = svm.SVC(kernel='linear', C=1.0).fit(X_train, y_train)
y_predicted = clf.predict(X_test)

print "Classification report for %s" % clf
print
print metrics.classification_report(y_test, y_predicted)
print
print "Confusion matrix"
print metrics.confusion_matrix(y_test, y_predicted)
print
print "Accuracy"
print metrics.accuracy_score(y_test, y_predicted)
```

Appendix F – Example pre-processing workflow

```
import nipype.pipeline.engine as pe

experiment_dir = '/home/medcomplab/workflows'

from nipype.interfaces.utility import IdentityInterface
#from pipeline - first pipeline demo by andsam 2014-04-15

imagelist =
['1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19','20','2
1','22','23','24','25','26','27','28','29','30']
imageinfo = dict(imaged=[['imageid']])
imagesource = pe.Node(IdentityInterface(fields=['imageid']), name='imagesource')
imagesource.iterables = ('imageid', imagelist)

from nipype.interfaces.io import DataGrabber
ig = pe.Node(DataGrabber(infields=['imageid'],
                         outfields='imaged'),
             name="imagegrabber")
ig.inputs.template = '%s/*.dcm'
#lets skip the dcm2nii and acpcdetect for now
ig.inputs.template_args = imageinfo
ig.inputs.sort_filelist = True
ig.inputs.base_directory = '/home/medcomplab/HCs/'

from nipype.interfaces.dcm2nii import Dcm2nii
converter = pe.Node(Dcm2nii(), name="converter")
converter.inputs.gzip_output = False
converter.inputs.reorient = False
converter.inputs.reorient_and_crop = False

from nipype.interfaces.acpcdetect import ACPCDetect
acpcd = pe.MapNode(ACPCDetect(), name="orient", iterfield=['infile'])

import nipype.interfaces.fsl as fsl
fsl.FSLCommand.set_default_output_type('NIFTI')
fslbet = pe.MapNode(fsl.BET(), name="fslbet", iterfield=['in_file'])

import nipype.interfaces.spm as spm
newseg = pe.MapNode(spm.NewSegment(), name="newseg", iterfield=['channel_files'])
newseg.inputs.matlab_cmd = "/opt/spm8/run_spm8.sh /opt/MATLAB/MATLAB_Compiler_Runtime/v713/
script"
newseg.inputs.use_mcr = True
pathtobox = '/opt/spm8/spm8_mcr/spm8/toolbox/Seg/TPM.nii'
tissue1 = ((pathtobox, 1), 2, (True,True), (False, False))
tissue2 = ((pathtobox, 2), 2, (True,True), (False, False))
tissue3 = ((pathtobox, 3), 2, (True,False), (False, False))
tissue4 = ((pathtobox, 4), 2, (False,False), (False, False))
tissue5 = ((pathtobox, 5), 2, (False,False), (False, False))
tissue6 = ((pathtobox, 5), 2, (False,False), (False, False))
newseg.inputs.tissues = [tissue1, tissue2, tissue3, tissue4, tissue5, tissue6]

from nipype.interfaces.io import DataSink
dsink = pe.Node(DataSink(), name="Datasink")
dsink.inputs.base_directory = experiment_dir
dsink.inputs.container = '/home/medcomplab/django/'

import nipype.interfaces.io as nio
mysqlt = pe.Node(nio.MySQLSink(input_names=['GM']), name='MySQL')
mysqlt.inputs.database_name = 'nipybas'
mysqlt.inputs.table_name = 'preproc_results'
mysqlt.inputs.username = 'nipyuser'
mysqlt.inputs.password = 'nipypass'

def get1class(dartel_files):
    class1images = []
    for session in dartel_files:
        class1images.extend(session[0])
    return class1images

def get2classes(dartel_files):
    class1images = []
```

```
    class2images = []
    for session in dartel_files:
        class1images.extend(session[0])
        class2images.extend(session[1])
    return [class1images, class2images]

def get3classes(dartel_files):
    class1images = []
    class2images = []
    class3images = []
    for session in dartel_files:
        class1images.extend(session[0])
        class2images.extend(session[1])
        class3images.extend(session[1])
    return [class1images, class2images, class3images]

preproc_workflow = pe.Workflow(name="Preprocess_workflow")
preproc_workflow.connect([(imagesource,ig,[('imageid','imageid')])])
preproc_workflow.connect([(ig,converter,[('imaged','source_names')])])
preproc_workflow.connect([(converter,acpcd,[('converted_files','infile')])])
preproc_workflow.connect([(acpcd,fslbet,[('outfile','in_file')])])
preproc_workflow.connect([(fslbet,newseg,[('out_file','channel_files')])])
preproc_workflow.connect([(newseg,dsink,[(('dartel_input_images',get3classes),'images.dartel')
])])
preproc_workflow.connect([(newseg,dsink,[(('native_class_images',get1class),'images.native')])
])
#preproc_workflow.connect([(newseg,mysqlt,[(('dartel_input_images',get1class),'GM')])])

#preproc_workflow.write_graph()
preproc_workflow.run()
```

Appendix G – Example processing workflow

```
I
import nipype.pipeline.engine as pe

from nipype import config
config.enable_debug_mode()

experiment_dir = '/home/medcomplab/workflows'
origmasks_dir = '/home/medcomplab/workflows/origmasks/'

from nipype.interfaces.utility import IdentityInterface
from nipype.interfaces.io import DataGrabber

#from pipeline - first pipeline demo by andsam 2014-04-15

imagelist = ['3','4','5']
imageinfo = dict(imaged=[['imageid']])
imagesource = pe.Node(IdentityInterface(fields=['imageid']), name='imagesource')
imagesource.iterables = ('imageid', imagelist)

ig = pe.Node(DataGrabber(infields=['imageid'],
                         outfields='imaged'),
               name="imagegrabber",
               iterfield=["imageid"])
ig.inputs.template = 'images/dartel/_imageid_%s/_newseg0/*.nii'
ig.inputs.template_args = imageinfo
ig.inputs.sort_filelist = True
ig.inputs.base_directory = '/home/medcomplab/papaya/segmentation_results/'

ng = pe.Node(DataGrabber(infields=['imageid'],
                         outfields='imaged'),
               name="nativegrabber",
               iterfield=["imageid"])
ng.inputs.template = 'images/native/_imageid_%s/_newseg0/*.nii'
ng.inputs.template_args = imageinfo
ng.inputs.sort_filelist = True
ng.inputs.base_directory = '/home/medcomplab/papaya/segmentation_results/'


import nipype.interfaces.spm as spm

def apu(in_val):
      apuri = map(list,zip(*in_val))
      return apuri

def simpleF(in_val):
      return in_val

def simpleD(flowfields,mni,imageid):
      return flowfields,mni

def simpleT(in_val,in_val2,in_val3):
      return in_val, in_val2, in_val3


def getFirst(in_val,temp):
      out_val = []
      for x in xrange(len(temp)):
          out_val.append([in_val[x]])
      return out_val

from nipype.interfaces.utility import Function
# joinnode implemented using Function because joinnode on DARTEL causes a crash
combine = pe.JoinNode(Function(input_names=["in_val"],output_names=["out_val"],function=apu),
                      name="combine", joinsource="imagesource", joinfield="in_val")

dartel = pe.Node(spm.DARTEL(), name="dartel")
dartel.inputs.matlab_cmd = "/opt/spm8/run_spm8.sh /opt/MATLAB/MATLAB_Compiler_Runtime/v713/
```

```
script"
dartel.inputs.use_mcr = True

joiner =
pe.JoinNode(Function(input_names=["in_val"],output_names=["out_val"],function=simpleF),
                      name="joiner", joinsource="imagesource", joinfield="in_val")

norm2mni = pe.Node(spm.DARTELNorm2MNI(), name="norm2mni")
norm2mni.inputs.matlab_cmd = '/opt/spm8/run_spm8.sh /opt/MATLAB/MATLAB_Compiler_Runtime/v713/
script'
norm2mni.inputs.use_mcr = True
norm2mni.inputs.modulate = True

coreg = pe.Node(spm.Coregister(), name="coreg")
coreg.inputs.matlab_cmd = '/opt/spm8/run_spm8.sh /opt/MATLAB/MATLAB_Compiler_Runtime/v713/
script'
coreg.inputs.use_mcr = True

presink =
pe.MapNode(Function(input_names=["flowfields","mni","imageid"],output_names=["flowfields","mni
"],function=simpleD),
                  name="presink",iterfield=["flowfields","mni"])

midsink =
pe.MapNode(Function(input_names=["flowfields","mni","imageid"],output_names=["flowfields","mni
"],function=simpleT),
                  name="midsink",iterfield=["flowfields","mni","imageid"])

from nipype.interfaces.io import DataSink
dsink = pe.Node(DataSink(), name="Datasink")
dsink.inputs.base_directory = experiment_dir
dsink.inputs.container = 'processing_results/'

def zipper(dartel_files):
      return zip(dartel_files[0],dartel_files[1])

def get2classes(dartel_files):
      class1images = []
      class2images = []
      for session in dartel_files:
          class1images.extend(session[0])
          class2images.extend(session[1])
      return [class1images, class2images]

def get1classes(dartel_files):
      class1images = []
      for session in dartel_files:
          class1images.extend(session[0])
      return [class1images]

def getclass1images(class_images):
    class1images = []
    for session in class_images:
        class1images.extend(session[0])
    return class1images

def getjustones(class_images):
    return class_images[:1]

process_workflow = pe.Workflow(name="Process_workflow")
process_workflow.connect([(imagesource,ig,[('imageid','imageid')])])
process_workflow.connect([(imagesource,ng,[('imageid','imageid')])])
process_workflow.connect([(ig,combine,[('imaged','in_val')])])
process_workflow.connect([(combine,dartel,[('out_val','image_files')])])
process_workflow.connect([(dartel,norm2mni,[('dartel_flow_fields','flowfield_files'),
                                     ('final_template_file','template_file')])])

process_workflow.connect([(ng,joiner,[('imaged','in_val')])])
process_workflow.connect([(joiner,norm2mni,[('out_val','apply_to_files')])])
process_workflow.connect([(dartel,dsink,[('final_template_file','dartel.template')])])
process_workflow.connect([(dartel,presink,[('dartel_flow_fields','flowfields')])])
process_workflow.connect([(norm2mni,presink,[('normalized_files','mni')])])
process_workflow.connect([(imagesource,presink,[('imageid','imageid')])])
process_workflow.connect([(presink,dsink,[('mni','mni'),("flowfields","dartel.flowfields")])])
```

```
#process_workflow.write_graph()
process_workflow.run()
```

Appendix H – Example co-registering workflow

This workflow is used to co-register selected masks to the template.

```
import nipype.pipeline.engine as pe

experiment_dir = '/home/medcomplab/workflows/'

from nipype.interfaces.utility import IdentityInterface
masklist = ['ec28','ec34']
maskinfo = dict(maski=[['maskid']])
masksource = pe.Node(IdentityInterface(fields=['maskid']), name='masksource')
masksource.iterables = ('maskid', masklist)

from nipype.interfaces.io import DataGrabber
mg = pe.Node(DataGrabber(infields=['maskid'],
                         outfields='maski'),
             name='maskgrabber')
mg.inputs.template = '/home/medcomplab/masks/%s.nii'
mg.inputs.template_args = maskinfo
mg.inputs.sort_filelist = True
mg.inputs.base_directory = experiment_dir

templist = ['Template_6']
tempinfo = dict(tempi=[['tempid']])
tempsource = pe.Node(IdentityInterface(fields=['tempid']), name='tempsource')
tempsource.iterables = ('tempid', templist)

tg = pe.Node(DataGrabber(infields=['tempid'],
                         outfields='tempi'),
             name='tempgrabber')
tg.inputs.template = 'processing_results/dartel/template/%s.nii'
tg.inputs.template_args = tempinfo
tg.inputs.sort_filelist = True
tg.inputs.base_directory = experiment_dir

import nipype.interfaces.spm as spm
import nipype.interfaces.io as nio

def apu(in_val):
    return in_val

from nipype.interfaces.utility import Function
# joinnode implemented using Function because joinnode on DARTEL causes a crash
combine = pe.JoinNode(Function(input_names=["in_val"],output_names=["out_val"],function=apu),
                      name="combine", joinsource="masksource", joinfield="in_val")

coreg = pe.Node(spm.Coregister(), name="coreg")
coreg.inputs.matlab_cmd = '/opt/spm8/run_spm8.sh /opt/MATLAB/MATLAB_Compiler_Runtime/v713/
script'
coreg.inputs.use_mcr = True
coreg.inputs.jobtype = 'write'

from nipype.interfaces.io import DataSink
dsink = pe.Node(DataSink(), name="Datasink")
dsink.inputs.base_directory = experiment_dir
dsink.inputs.container = 'processing_results/'

coreg_workflow=pe.Workflow(name="Coregisteration_workflow")
coreg_workflow.connect([(masksource,mg,[('maskid','maskid')])])
coreg_workflow.connect([(tempsource,tg,[('tempid','tempid')])])

coreg_workflow.connect([(tg,coreg,[('tempi','target')])])
coreg_workflow.connect([(mg,combine,[('maski','in_val')])])
coreg_workflow.connect([(combine,coreg,[('out_val','source')])])

coreg_workflow.connect([(coreg,dsink,[('coregistered_source','rmasks')])])

#coreg_workflow.write_graph()
coreg_workflow.run()
```

Appendix I – Example analysis workflow

This workflow requires either changes to directories or some symbolic linking to be done after last workflow to execute properly.

```
import nipype.pipeline.engine as pe

experiment_dir = '/home/medcomplab/workflows/processing_results'

from nipype.interfaces.utility import IdentityInterface
masklist = ['ec28','ec34']
maskinfo = dict(maski=[['maskid']])
masksource = pe.Node(IdentityInterface(fields=['maskid']), name='masksource')
masksource.iterables = ('maskid', masklist)

from nipype.interfaces.io import DataGrabber
mg = pe.Node(DataGrabber(infields=['maskid'],
                         outfields='maski'),
             name='maskgrabber')
mg.inputs.template = '/home/medcomplab/masks/r%s.nii'
mg.inputs.template_args = maskinfo
mg.inputs.sort_filelist = True
mg.inputs.base_directory = experiment_dir

imagelist = ['4','5']
imageinfo = dict(imagi=[['imagid']])
imagesource = pe.Node(IdentityInterface(fields=['imagid']), name='imagesource')
imagesource.iterables = ('imagid', imagelist)

ig = pe.Node(DataGrabber(infields=['imagid'],
                         outfields='imagi'),
             name='imagegrabber')
ig.inputs.template = 'mni/*.nii'
#ig.inputs.template = 'mni/_imageid_%s/_norm2mni0/*.nii'
ig.inputs.template_args = imageinfo
ig.inputs.sort_filelist = True
ig.inputs.base_directory = experiment_dir

from nipype.interfaces.io import DataFinder
imaf = pe.Node(DataFinder(outfileds='images'), name='imagefinder')
imaf.inputs.root_paths = '/home/medcomplab/workflows/processing_results/mni'
imaf.inputs.match_regex = 'swmc.+\.nii'
imaf.inputs.max_depth = 0
imaf.inputs.min_depth = 0


import nipype.interfaces.spm as spm
import nipype.interfaces.io as nio

def apu(in_val):
    return in_val

from nipype.interfaces.utility import Function
# joinnode implemented using Function because joinnode on DARTEL causes a crash
combine = pe.JoinNode(Function(input_names=["in_val"],output_names=["out_val"],function=apu),
                      name="combine", joinsource="masksource", joinfield="in_val")

coreg = pe.Node(spm.Coregister(), name="coreg")
coreg.inputs.matlab_cmd = '/opt/spm8/run_spm8.sh /opt/MATLAB/MATLAB_Compiler_Runtime/v713/
script'
coreg.inputs.use_mcr = True

gett = pe.MapNode(spm.GetTotals(), name='GetTotals', iterfield='in_image')
gett.inputs.matlab_cmd = '/opt/spm8/run_spm8.sh /opt/MATLAB/MATLAB_Compiler_Runtime/v713/
script'
gett.inputs.use_mcr = True

mysqlt = pe.MapNode(nio.MySQLSink(input_names=['subject','totals','mask']), name='MySQL',
iterfield=['totals', 'subject'])
mysqlt.inputs.database_name = 'nipybas'
```

```
mysqlt.inputs.table_name = 'nipytab'
mysqlt.inputs.username = 'nipyuser'
mysqlt.inputs.password = 'nipypass'

def getasstring (filepath):
    import os
    return os.path.basename('%s' % filepath)

totals_workflow=pe.Workflow(name="Totals_workflow")
#totals_workflow.connect([(imagesource,ig,[('imagid','imagid')])])
#totals_workflow.connect([(ig, gett, [('imagi','in_image')])])
totals_workflow.connect([(imaf,gett, [('images','in_image')])])
totals_workflow.connect([(gett, mysqlt, [('total_volume','totals'),
                                         ('subject','subject')])])

analyze_workflow=pe.Workflow(name="Analysis_workflow")
analyze_workflow.connect([(masksource,mg,[('maskid','maskid')])])
analyze_workflow.connect([(mg,totals_workflow,[('maski',
                                                'GetTotals.in_mask')])])
analyze_workflow.connect([(mg,totals_workflow,[(('maski',getasstring),'MySQL.mask')])])

#analyze_workflow.write_graph()
analyze_workflow.run()
```

Appendix J – Test environment install script

With this script it is possible to install all needed programs and packages to a new Ubuntu Desktop 12.04.4 LTS 64-bit environment (to make the system functional, enter then the changes and additions outlined in previous appendices). Please note, that the URLs for SPM-standalone package and Matlab Runtime need to be inserted into the script for it to work properly. The URLs are not included as by the time of writing the thesis, SPM standalone is still in closed beta testing, and to download it permission needs to be acquired from UCL.

```
#/bin/sh
echo "Give sudo password if needed - and remember to fix URL before running script!"
sudo echo " Passed!"
echo "Getting system up to date"
sudo apt-get update
sudo apt-get upgrade -y
sudo apt-get update
sudo apt-get dist-upgrade -y
read -t10 -n1 -p "Press ctrl+c to abort - continuing in 10 secs... " key
echo "Setting up NeuroDebian repositories"
wget -O- http://neuro.debian.net/lists/precise.au.full | sudo tee
/etc/apt/sources.list.d/neurodebian.sources.list
sudo apt-key adv --recv-keys --keyserver pgp.mit.edu 2649A5A9
sudo apt-get update
read -t10 -n1 -p "Press ctrl+c to abort - continuing in 10 secs... " key
echo "Installing tools: git"
sudo apt-get install -y git
echo "Installing Nipype"
sudo apt-get install -y python-nipype
read -t10 -n1 -p "Press ctrl+c to abort - continuing in 10 secs... " key
echo "Loading SPM8 standalone packages for a64 (check correct URL!)"
if wget http://FIX-CORRECT-URL-HERE/spm8_r5236.zip; then
 if wget http://FIX-CORRECT-URL-HERE/glnxa64/MCRInstaller.bin; then
  echo "Installing Matlab standalone environment - this might take minutes"
  chmod a+x MCRInstaller.bin
  sudo ./MCRInstaller.bin -silent
  echo "UnZipping SPM8"
  sudo unzip spm8_r5236.zip -d /opt
  echo "Trying to run SPM8 fmri on Matlab runtime - if case of errors, check paths!"
  sudo chmod a+x /opt/spm8/run_spm8.sh
  sudo /opt/spm8/run_spm8.sh /opt/MATLAB/MATLAB_Compiler_Runtime/v713 script
  echo "Matlab and SPM installed"
else
  echo "Loading Matlab runtime package failed – please check URL"
 fi
else
 echo "Loading SPM-standalone package failed – please check URL"
fi
read -t10 -n1 -p "Press ctrl+c to abort - continuing in 10 secs... " key
echo "Installing FSL"
if sudo apt-get install -y fsl-complete; then
 echo "Setting up environment for FSL"
 echo "export FSLDIR=/usr/share/fsl/5.0" >>~/.bashrc
 echo "export FSLOUTPUTTYPE=NIFTI" >>~/.bashrc
 echo "export PATH=/usr/lib/fsl/5.0:\$PATH" >>~/.bashrc
 echo "export LD_LIBRARY_PATH=/usr/lib/fsl/5.0" >>~/.bashrc
 echo ". /usr/share/fsl/5.0/etc/fslconf/fsl.sh" >>~/.bashrc
 echo "Ready!"
else
 echo "There were problems with FSL installation! Please try again!"
fi
read -t10 -n1 -p "Press ctrl+c to abort - continuing in 10 secs... " key
echo “Setting up environment variables for SPM, FSL and acpcdetect” >~/.bashrc
echo "export SPMMRCCMD=\"/opt/spm8/run_spm8.sh /opt/MATLAB/MATLAB_Compiler_Runtime/v713/
script\"" >>~/.bashrc
```

```
echo "export MATLABCMD=\"/opt/spm8/run_spm8.sh /opt/MATLAB/MATLAB_Compiler_Runtime/v713/
script\"" >>~/.bashrc
echo "export FORCE_SPMMRC=true" >>~/.bashrc
read -t10 -n1 -p "Press ctrl+c to abort - continuing in 10 secs... " key
echo "Downloading Nipype-Tutorial materials"
wget http://sourceforge.net/projects/nipy/files/nipype/nipype-0.2/nipype-
tutorial.tar.bz2/download
echo Unzipping
tar xvf download
read -t10 -n1 -p "Press ctrl+c to abort - continuing in 10 secs... " key
wget http://www.nitrc.org/frs/download.php/6494/acpcdetect.tar.gz
mkdir acpcdetect
if cd acpcdetect; then
 tar xvf acpcdetect.tar.gz
 cd ..
fi
sudo apt-get install libstdc++5
echo "export ARTHOME=/home/medcomplab/install/acpcdetect" >>~/.bashrc
echo "export PATH=\$ARTHOME:\$PATH" >>~/.bashrc
read -t10 -n1 -p "Press ctrl+c to abort - continuing in 10 secs... " key
echo "Installing MySQL and other tools"
sudo apt-get install -y mysql-server
sudo apt-get install -y mysql-client
sudo apt-get install -y python-mysqldb
read -t10 -n1 -p "Press ctrl+c to abort - continuing in 10 secs... " key
sudo apt-get install -y python-sklearn
read -t10 -n1 -p "Press ctrl+c to abort - continuing in 10 secs... " key
sudo apt-get install -y python-django
read -t10 -n1 -p "Press ctrl+c to abort - continuing in 10 secs... " key
#echo "Installing PHPMYAdmin"
#sudo apt-get install -y phpmyadmin

echo "Install script finnished, please take a Snapshot of the virtual machine!”
```


KAROLINSKA INSTITUTET
ANNO 1810
Karolinska
Institutet


UNIVERSITET
STOCKHOLMS
Stockholm
University