CONVERTING A NEURON MORPHOLOGY EXTRACTION SYSTEM FOR OPEN SCIENCE: DESIGN AND IMPLEMENTATION

A Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Zakariyya Mughal

December 2015

CONVERTING A NEURON MORPHOLOGY EXTRACTION SYSTEM FOR OPEN SCIENCE: DESIGN AND IMPLEMENTATION

Zakariyya Mughal

APPROVED:

Dr. Ioannis A. Kakadiaris, Chairman
Dept. of Computer Science

Dr. Emmanuel Papadakis
Dept. of Mathematics

Dr. Shishir Shah
Dept. of Computer Science

Dean, College of Natural Sciences and Mathematics

CONVERTING A NEURON MORPHOLOGY EXTRACTION SYSTEM FOR OPEN SCIENCE: DESIGN AND IMPLEMENTATION

An Abstract of a Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Zakariyya Mughal

December 2015

Abstract

The thesis describes the conversion of the ORION system for neuron morphology extraction from an interpreted language to a compiled language. The purpose of this conversion is to provide a tool that can be used by neuroscience researchers to analyse their own neuron data and compare the output against both manual and automated tracings. This is in line with the goals of open science: a movement that seeks to make the findings and processes of research more widely available for peer review and reproducibility. By collaboratively sharing both neuron imaging data and code between organisations, it is possible to compare results of multiple methods without reimplementing all the stages of the reconstruction pipeline.

In order to release the existing algorithm so that it can easily be incorporated into other tools, the implementation must be rewritten in a different language. This presents a challenge because the languages have vastly different paradigms. As such, much of the existing code needs to be analysed to determine any changes to the design. Creating a new implementation also means that the new system can be designed with modifiability in mind so that future changes can be easily incorporated. The specific objectives are to (i) analyse the ORION algorithm and implementation to determine the architecture for the new system that is efficient and extensible; (ii) integrate the system into a popular toolkit for biomedical image analysis for ease-of-use and visualisation; (iii) develop a test suite of both the individual components (unit testing) and across the whole system (integration tests); and (iv) ensure that the software gives reproducible results by making it easy to build and distribute.

The extraction of neuron morphology from microscopy imaging data is an invaluable method for understanding neuron characteristics. However, due to the cost in

time and effort, manual neuron reconstruction is not feasible for large-scale analysis of neuron datasets. This implementation provides a working method for determining neuron morphology that can be used to collect statistical properties from various neuron data.

Keywords: neurons, cell morphology, biomedical image analysis, software engineering

Contents

1 Introduction **1**

1.1 Motivation 2

1.1.1 Scientific software 3

1.1.2 Open science and scientific software engineering 6

1.1.3 Open neuroscience 7

1.1.4 BigNeuron 8

1.2 Literature review 10

1.2.1 Neuron reconstruction and tracing 10

1.3 Survey of libraries used 11

2 Planning and Analysis **12**

2.1 System objectives 13

2.2 Benefits 13

2.3 Design principles 14

2.4 Challenges and risks 14

2.5 MATLAB version of ORION 3 16

2.5.1 Callgraph 16

2.5.2 Processing pipeline 19

2.5.3 Algorithm 19

3 Design **20**

3.1 Architecture 20

4 Implementation **21**

5 Testing and verification **22**

6 Discussion **23**

7 Conclusion **24**

List of Figures

1.1 Systems development life cycle 5

List of Tables

> “Each discovery made by an investigator in a basic research laboratory has much larger implications today. The sum of the work in basic biology represents a rapidly expanding tool kit for engineers and inventors to use to construct items of value to society.”
>
> David Baltimore in *How Biology Became an Information Science*, 2001 (Baltimore 2001)

1 Introduction

This thesis aims to detail the design and implementation of a neuron morphology extraction system. This system is the result of converting the existing ORION 3 (Santamaria-Pang et al. 2015) system from MATLAB (MATLAB 2013) code to native C/C++ code. This conversion process requires an analysis of the existing code to understand its structure and create a plan for replicating the same functionality

in the in the new system.

This software project can be categorised as a *rewrite*; that is, a replication of an existing software system without reusing the existing code. In software engineering, the consensus on rewriting software from scratch is that it is difficult and that teams should avoid rewrites (Spolsky 2000). This view arises because there are several challenges and risks associated with rewriting large systems. Rewrites often take a long time and instead of adding features, development time is spent on redesign and reimplementation of old features. In addition, all the institutional knowledge that came from years of bug fixes is often lost with a rewrite. Rewrites are often expensive in terms of time and effort and are not known to payoff as much as product owners wish. As such, it is preferable to work on slowly refactoring the code rather than a complete rewrite. Refactoring is a process where instead of throwing away the existing system, the development proceeds by making small, incremental changes over an extended period in order to avoid getting the software into a broken state, while steadily improving the readability and reusability of the software.

1.1 Motivation

In order to understand why this project is being undertaken, it is important to understand why a rewrite is necessary as opposed to refactoring the existing codebase. Both options require an analysis of the project deliverables, that is, a concrete set of goals that will direct the project development. These deliverables can be classified as either scientific or engineering deliverables. Scientific deliverables are motivated

by goals that advance the state of scientific knowledge while engineering deliverables focus on specific technical aspects of the project that make future project maintenance and growth possible. This section will only cover the scientific deliverables while engineering deliverables are covered in Section 2.1.

The *primary scientific deliverable* of this thesis is a system for neuron reconstruction. This system is designed with the principles of open science in mind so that it is usable by biologists to study neuroanatomy. Section 1.1.2 contains further discussion on open-science and how this project achieves this goal.

The *secondary scientific deliverable* is to make this system compatible with existing tools for image analysis so that it can be compared against other methods as part of the BigNeuron project. The role of BigNeuron in neuroscience research is covered in Section 1.1.4.

The following sections contain an overview of the context of this project first within the general context of scientific software (Section 1.1.1) and finally narrow down its role relative to the specific context of neuron reconstruction (Section 1.2.1).

1.1.1 Scientific software

Quantitative methods are an essential part of scientific research. The experimental sciences depend on the dissemination of the methods used for each study so that it is clear how results are obtained and analysed. With the rapid increase in computing power, storage, and availability, it has become easier to collect and process larger and more complex datasets using sophisticated methods and this has made software and

software development an important part of all experimental fields (Baxter et al. 2006; Hettrick et al. 2014). To produce reproducible results, some common approaches to this change are to publish either a description of the algorithm or refer to software that can be obtained separately (either in binary or source code form).

Despite these approaches, there are still several challenges to successfully reproducing a published method. A textual description of an algorithm is rarely a complete description of how an algorithm is implemented. Sophisticated methods often have tiny details that are mistakenly left out which may be essential for the rest of the processing. One specific area where this occurs often is in data preprocessing and annotation stages which can involve human interaction and, as such, some assumptions may not be written down. Referencing readily available software packages gives other researchers direct access to the original method used. However, even here, there can be problems. Even before running the software, it is important to ensure that the software is available years later. This means not only the software itself, but all its dependencies. This can quickly become complicated as technology advances: changes in the Application Programming Interface (API) or Application Binary Interface (ABI) can cause incompatibility issues for both software distributed in source form or binary form. Both source code and binary software can have dependencies on platforms (e.g., operating systems, runtimes, computer architecture, etc.) and licensing of components that can hinder others from using the software.

Even more precarious is when a dependency used by the software no longer behaves the same way as it did in previous versions. This can lead to software that appears to run, but gives unintended results. Maintaining backwards compatibility

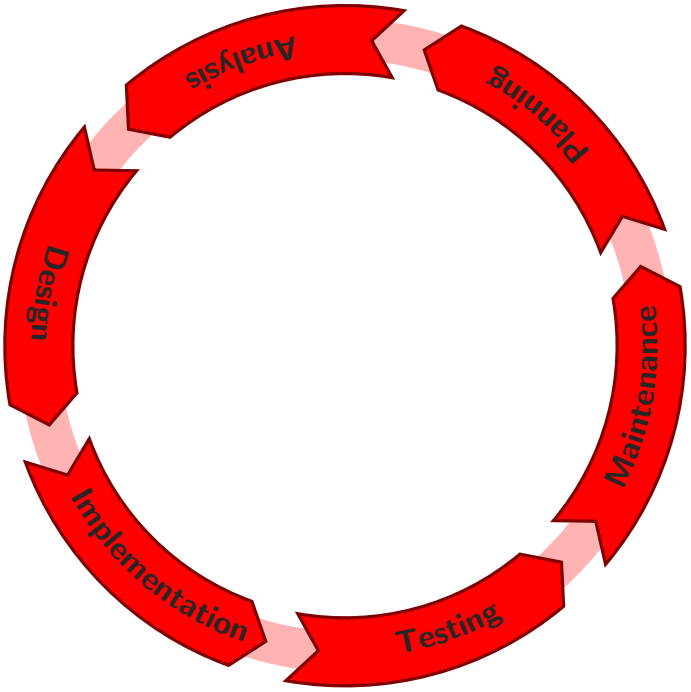


Figure 1.1: **Systems development life cycle**: This figure depicts an example of a life cycle used to delineate the phases of development (Justice Management Division 2003). The phases are: a) **Planning:** In this phase, project management and resource allocation details are determined. This includes scheduling, tools, and defining project objectives; b) **Analysis:** This phase analyses the project requirements and defines specific technical milestones that relate to the objectives defined in the Planning phase; c) **Design:** The parts of the system are delineated and the expected input/output characteristics of each part are determined; d) **Implementation:** The physical system is built during this phase using the system design from the previous phase; e) **Testing:** As the Implementation phase proceeds, each standalone part is tested individually in what are known as unit tests. When the parts interact with each other, integration tests are conducted to determine that these parts are compatible; f) **Maintenance:** In this phase, the system is put in production and monitored for changes in system performance and project requirements. When these changes necessitate an update to the system, the project may return to the Planning phase. It is important to note that the phases of this life cycle are not discrete; there is overlap between phases. For example, parts of the design may change as the implementation continues as more information about the physical system is available.

for software is difficult since there are many parts to a non-trivial software system. It may be possible to identify these compatibility problems using testing (for more discussion, see Chapter 5). Resolving the exact versions of libraries and toolchains needed to build and run can be frustrating and is commonly known as *dependency hell*. This problem can become daunting when dealing with multiple platforms and many libraries. As newer versions of these libraries are released, the maintenance phase of the *systems development life cycle* becomes more important (see Figure 1.1). Unmaintained software is prone to what is known as *bit rot* — that is, the process through which software that was working no longer works due to changes in the surrounding software ecosystem. There has been some work to prevent bit rot by recording a static copy of the software environment, but digital preservation (comprising both computing machinery and software) is in its infancy and has not caught up to that of paper-based materials (National Digital Information Infrastructure and Preservation Program 2013; Thain et al. 2015; Meng et al. 2015).

1.1.2 Open science and scientific software engineering

As science becomes more oriented towards using computational tools, the ideals of reproducibility and statistical hypothesis testing become more difficult to achieve. In recent years, there has been a push by researchers to incorporate *open science* practices in their work. One prominent advocate for open science defines this as follows:

> Open science is the idea that scientific knowledge of all kinds should be openly shared as early as is practical in the discovery process.

Michael Nielsen[1]

The "scientific knowledge mentioned in the above quote encompasses any kind of knowledge including problem definitions, ideas, code, data, methodology, and journal articles. The goal is to allow for more opportunities for collaboration and sharing of information.

When it comes to computer science research, sharing of the code

According to a survey by the Software Sustainability Institute, scientific disciplines implement their own code for research (Hettrick et al. 2014).

Software causing retractions (Miller 2006; Merali 2010; Joppa et al. 2013) on top of already shakey statistical practices (Ioannidis 2005; Button et al. 2013).

teaching best practices through the Software Carpentry (Gregory Wilson 2006; Greg Wilson et al. 2014)

1.1.3 Open neuroscience

Biology and neuroscience in particular have been using more quantitative imaging tools: patch clamp, EEG, fMRI, fluorescent microscopy ... need to measure what is being looked at ... perform statistical analysis across samples

access to data (Parekh et al. 2015; Poldrack 2011).

Allen Institute for Brain Science, Big Brain project, Big Neuron NITRC Neuinfo Framework Neuromorpho EyeWire Knossos / Brainflight VirtualFlyBrain (Halchenko et al. 2012)

1.1.4 BigNeuron

The secondary deliverable is to make the system compatible with the Vaa3D biomedical imaging toolkit (Hanchuan Peng Group 2015; Peng, Ruan, et al. 2010; Peng, Bria, et al. 2014). This toolkit allows biologists to visualise and analyse biomedical imaging datasets. It can be extended through the development of plugins. This allows algorithm developers to make their both their interactive and non-interactive methods for image analysis available for biologists to use without having to switch between multiple programs for processing and visualising data.

Creating an image analysis tool that can be easily integrated into other systems allows others to reproduce the results in order to compare with ground truth data and other algorithms. There has been an attempt at comparing neuron morphology extraction algorithms in the past under the DIADEM Challenge which contributed datasets and a metric for comparing the neuron tracings from algorithms against tracings from gold standard segmentations (Liu 2011; Brown et al. 2011; Todd A. Gillette et al. 2011). The DIADEM Challenge used 6 datasets from different neuroscience institutions in order to have a diversity in terms of source of the neurons (i.e., from different species and structures from different brain regions) and in terms

of the laboratory protocols (i.e., different labelling and microscopy techniques).

The DIADEM Challenge was successful in raising awareness of the problem of neuron reconstruction and several new 3D reconstruction methods and metrics have been proposed after the end of the DIADEM Challenge (see Section 1.2.1). However, the lack of larger public datasets, standardised metrics, and readily available algorithms have made it difficult to compare new methods for neuron reconstruction.

To solve this problem, an open-science project called BigNeuron (Peng, Hawrylycz, et al. 2015; Peng, Meijering, et al. 2015) continues where the DIADEM Challenge left off. Instead of comparing the output on a few datasets as in the DIADEM Challenge, BigNeuron aims to enable high-throughput analysis of neuron microscopy stacks using multiple methods contributed from various algorithm developers. By standardising on the Vaa3D platform, this precludes differences between how each method handles data which means that all the algorithms can be bench-tested at once without a need for translating data between formats.

However, since the Vaa3D is written in C++, incorporating plugin code written in non-native languages poses a problem. For this reason, the BigNeuron project organisers recommend that all algorithms that are submitted be in C or C++. From the BigNeuron FAQ (BigNeuron Consortium 2015)

> Q3. **How can I incorporate code written in Matlab, Java, Python or another language other than C/C++?**
>
> A. BigNeuron is a very large scale project, and enforcing a unified API is critical to ensure fair comparison for any pre-defined assessment. We thus discourage usage of Matlab, Java, Python or other programming languages besides C/C++ for this bench testing.

1.2 Literature review

1.2.1 Neuron reconstruction and tracing

(Cannon et al. 1998; Brown et al. 2011)

(Liu 2011; Halavi et al. 2012; Meijering 2010)

Post-DIADEM (Bauer et al. 2010; Xie et al. 2011; Xie et al. 2010; Jeong et al. 2015; Luo et al. 2015; De et al. 2015; Gulyanon et al. 2015; Santamarıa-Pang et al. 2015; Mukherjee, Condron, et al. 2014; Hernandez-Herrera, Papadakis, et al. 2014; Basu and Racoceanu 2014; Xiao et al. 2013; Jiménez et al. 2013; Basu, Kulikova, et al. 2013; Mukherjee and Acton 2013; Hernandez-Herrera, Jiménez, et al. 2013; Ming et al. 2013; Lee et al. 2012; Czarnecki 2012)

metrics (Mayerich, Bjornsson, Jonothan Taylor, et al. 2011; Mayerich, Bjornsson, Jonathan Taylor, et al. 2012; Torben-Nielsen 2014; Costa et al. 2014; Todd Aaron Gillette 2015)

(i) to evaluate the MATLAB codebase of ORION 3 and determine how to structure

the new codebase

(ii) integrate the

1.3 Survey of libraries used

> “Measure twice and cut once.”
>
> Carpentry rule of thumb

> “But cut ting is more fun than m eas uri ng!”
>
> Anonymous

2

Planning and Analysis

Before writing a single line of code, it is a good practice to understand the scope of the problem. Gathering the deliverables and requirements of the project is necessary not only for understanding how long the project will take, but also the order in which to implement each component. Establishing this order proves very useful for later in the testing phase of the project (Chapter 5).

2.1 System objectives

As opposed to the the scientific delieverables discussed in Section 1.1, the system objectives are the engineering delieverables; these are more closely tied to the Design and Implementation phases as these objectives influence decisions made about the underlying resources and system architecture.

The *primary engineering deliverable* is a complete conversion of the existing code from MATLAB to native code. This involves an analysis of the existing code to see which parts of the algorithm will be converted.

The *second engineering deliverable* is a test suite to verify that the components of the system operate to the design

2.2 Benefits

In addition, a rewrite allows for more thorough design and testing which is necessary to verify that the code behaves as expected and will continue to do so in the future.

2.3 Design principles

2.4 Challenges and risks

While the code for the algorithm already exists, starting with a line-by-line translation of the MATLAB code has some limitations outlined as follows.

Toolbox The code is written to use MATLAB's extensive specialised toolboxes for image processing and statistics which means that equivalents must be incorporated into the new codebase.

Memory management Since MATLAB is a dynamic array language with automatic memory management, it is simple to create multidimensional array and extend it without having to keep track of the variable's size or the variable lifetime. Since C uses manual memory management, it is necessary to manually allocate and release memory to avoid memory leaks.

Caching The ORION MATLAB code makes frequent use of the file system to cache calculations between runs. The purpose of this is to speed up experiments so that when an experiment is rerun, any images that have been processed in an earlier stage (i.e., segmentation) do not need to be reprocessed in later stages (i.e., centreline extraction). Code written in this form imposes an algorithm structure that is no longer strictly imperative — the code is now interspersed with checks to see if the data already exists and instead of passing the data between functions using multidimensional arrays as parameters, the parameters

to the functions are filenames.

Subvolume The MATLAB code breaks up the input data into subvolumes. This allows the computation to run a small region of the data which allows for processing data that may be too large to fit entirely memory. Furthermore, when used in conjunction with the aforementioned caching, the steps used for each processing stage can be more granular which means that if any processing is incomplete (e.g., because the computer runs out of memory or disk space), the data is not entirely lost. However, this complicates the algorithm because any calculation involving coordinates in a volume must map indices in subvolumes to indices in the corresponding supervolume.

The Caching and Subvolume issues both indicate issues that can be described as cross-cutting concerns. Cross-cutting concerns are

This project has

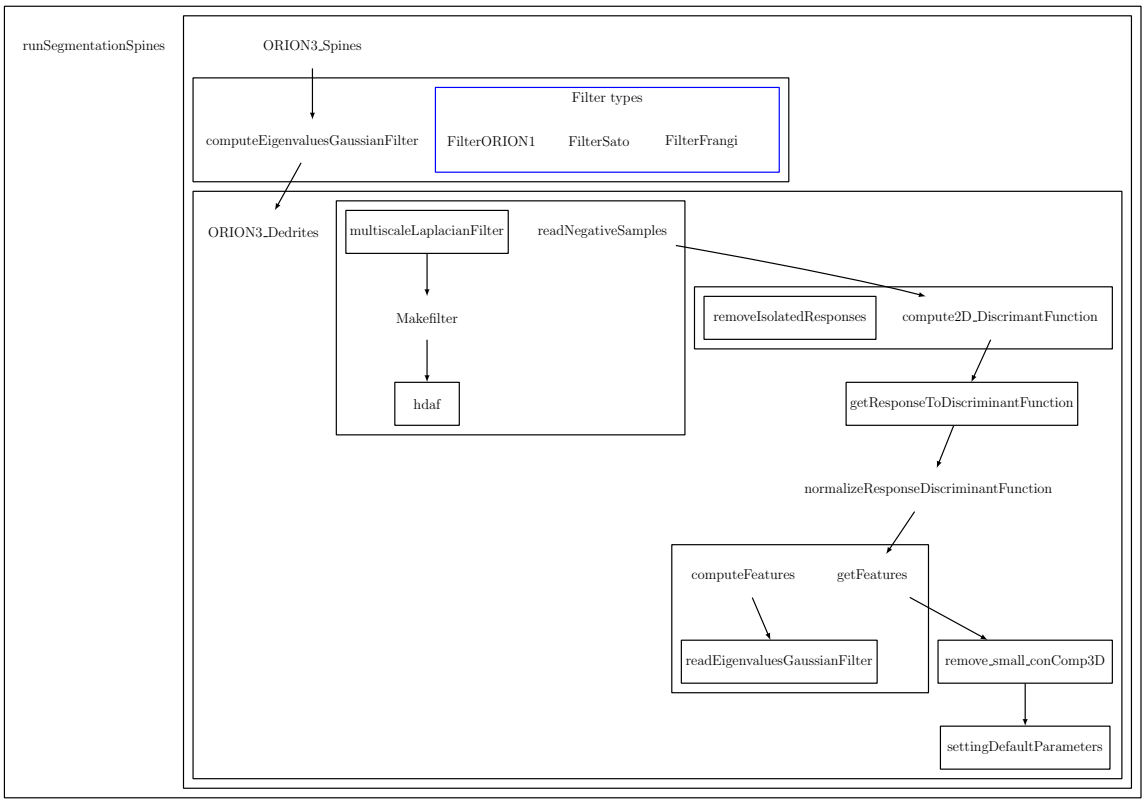
- starting with no tests
- no automated build system
- works entirely inside of MATLAB
- has executable components that are called via `system()` which means that they need to be compiled and the path to the components needs to be set

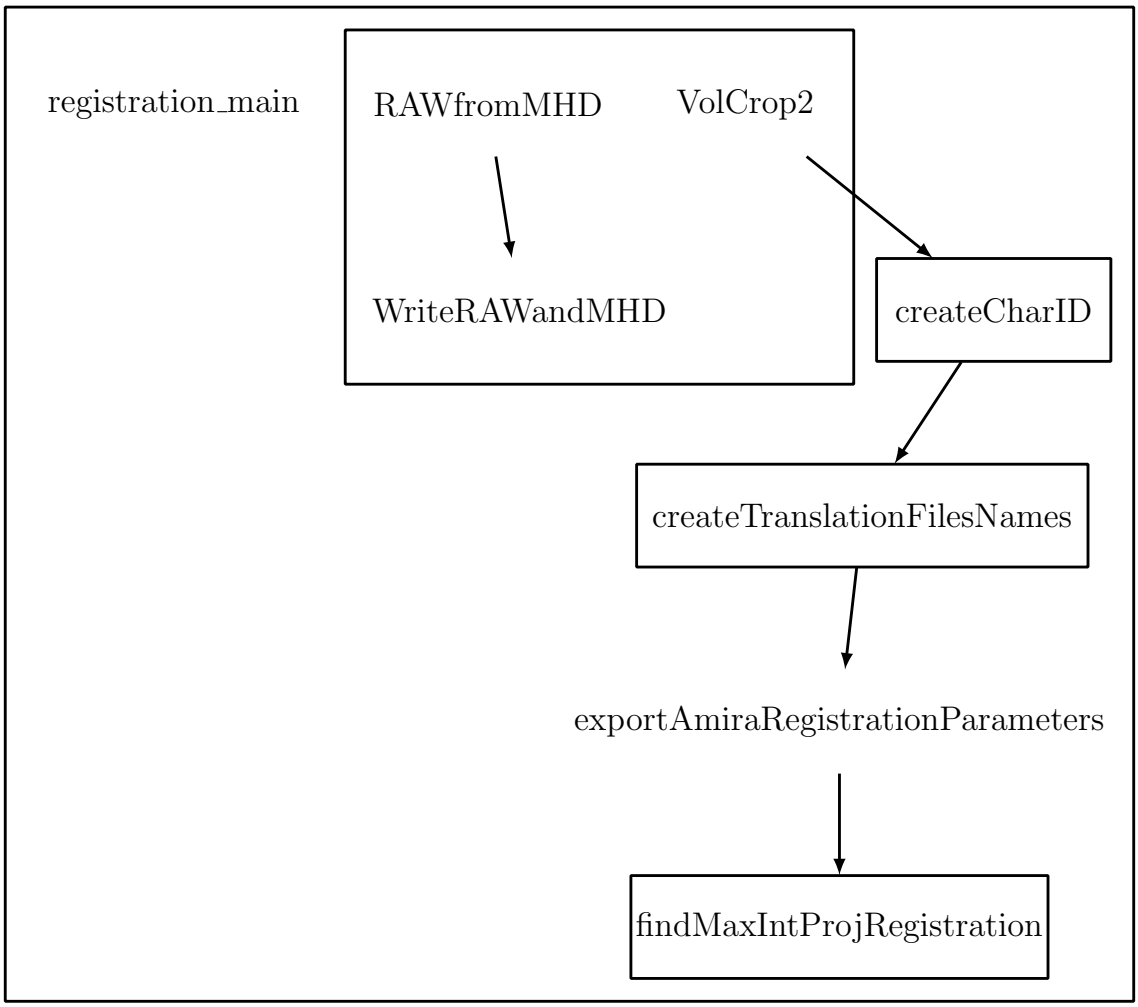
2.5 MATLAB version of ORION 3

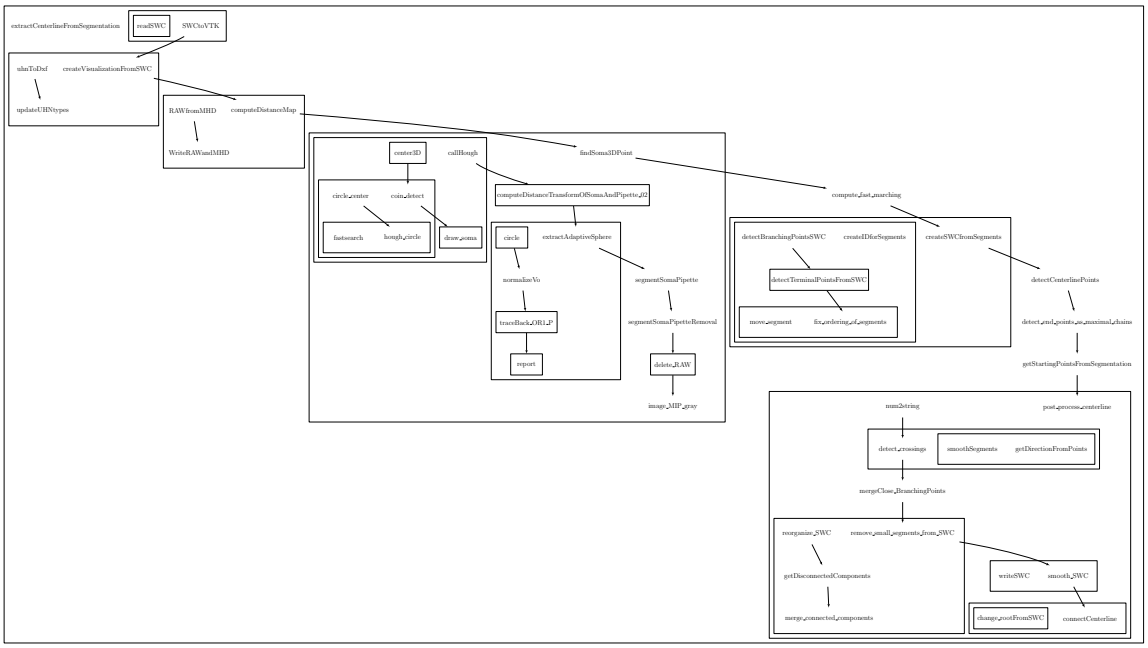
2.5.1 Callgraph

In order to understand how the ORION MATLAB code is structured, it is necessary to first get the call graph of the code. There is a tool built in to MATLAB to do this called depfun `http://www.mathworks.com/help/matlab/ref/depfun.html` , however this tool runs slowly when running on the entire codebase. There is an alternative called fdep `http://www.mathworks.com/matlabcentral/fileexchange/17291-fdep--a-pedestrian-function-dependencies-finder`

The output of this tool is graph that


runSegmentationSpines
ORION3_Spines
computeEigenvaluesGaussianFilter
Filter types
FilterORION1
FilterSato
FilterFrangi
ORION3_Dedrites
multiscaleLaplacianFilter
readNegativeSamples
Makefilter
hdaf
removeIsolatedResponses
compute2D_DiscrimantFunction
getResponseToDiscriminantFunction
normalizeResponseDiscriminantFunction
computeFeatures
getFeatures
readEigenvaluesGaussianFilter
remove_small_conComp3D
settingDefaultParameters


registration_main
RAWfromMHD
VolCrop2
WriteRAWandMHD
createCharID
createTranslationFilesNames
exportAmiraRegistrationParameters
findMaxIntProjRegistration



2.5.2 Processing pipeline

2.5.3 Algorithm

> The trouble with computers is you *play* with them. They are so wonderful. You have these switches — if it's an even number you do this, if it's an odd number you do that — and pretty soon you can do more and more elaborate things if you are clever enough, on one machine.
>
> Richard Feynman in *Surely You're Joking, Mr. Feynman!: Adventures of a Curious Character*, 1985

3 Design

3.1 Architecture

> One of my most productive days was throwing away 1000 lines of code.
>
> Ken Thompson

4

Implementation

> “ Everyone knows that debugging is twice as hard as writing a program in the first place. So if you’re as clever as you can be when you write it, how will you ever debug it? ”
>
> Brian W. Kernighan and P. J. Plauger in
>
> *The Elements of Programming Style*, 1978

5

Testing and verification

> The purpose of computing is insight, not numbers.
>
> Richard W. Hamming in *Numerical Methods for Scientists and Engineers*, 1962

> There's no sense in being precise when you don't even know what you're talking about.
>
> John von Neumann

6

Discussion

> This duality can be pursued further and is related to a duality between past and future and the notions of control and knowledge. Thus we may have knowledge of the past but cannot control it; we may control the future but have no knowledge of it.
>
> Claude E. Shannon in *Coding theorems for a discrete source with a fidelity criterion*, 1959

7

Conclusion

Bibliography

1. Baltimore, David (2001). In: *The Invisible Future: The Seamless Integration of Technology into Everyday Life*. Ed. by Peter J. Denning. New York, NY, USA: McGraw-Hill, Inc. Chap. How Biology Became an Information Science, pp. 43–55. URL: http://dl.acm.org/citation.cfm?id=504949.504953.

2. Basu, Sreetama, Maria Kulikova, et al. (2013). "A stochastic model for automatic extraction of 3D neuronal morphology". In: *Medical Image Computing & Computer-Assisted Intervention: MICCAI* 16, pp. 396–403. DOI: 10.1007/978-3-642-40811-3{_}50. URL: http://link.springer.com/chapter/10.1007/978-3-642-40811-3_50.

3. Basu, Sreetama and Daniel Racoceanu (2014). "Reconstructing neuronal morphology from microscopy stacks using fast marching". In: *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 3597–3601. DOI: 10.1109/ICIP.2014.7025730. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7025730.

4. Bauer, Christian et al. (2010). "Segmentation of interwoven 3D tubular tree structures utilizing shape priors and graph cuts". In: *Medical image analysis* 14.2, pp. 172–184. URL: http://www.sciencedirect.com/science/article/pii/S1361841509001406.

5. Baxter, Susan M. et al. (2006). "Scientific software development is not an oxymoron". In: *PLoS Computational Biology* 2.9, pp. 0975–0978. DOI: 10.1371/journal.pcbi.0020087.

6. BigNeuron Consortium (2015). *Frequently Asked Questions*. URL: https://alleninstitute.org/bigneuron/faq/ (visited on 08/07/2015).

7. Brown, Kerry M et al. (2011). "The DIADEM data sets: representative light microscopy images of neuronal morphology to advance automation of digital reconstructions". In: *Neuroinformatics* 9.2-3, pp. 143–57. DOI: 10.1007/s12021-010-9095-5. URL: http://www.ncbi.nlm.nih.gov/pubmed/21249531.

8. Button, Katherine S. et al. (2013). "Power failure: why small sample size undermines the reliability of neuroscience". In: *Nature Reviews Neuroscience* 14.5, pp. 365–376. DOI: 10.1038/nrn3475. URL: http://www.nature.com/doifinder/10.1038/nrn3475.

9. Cannon, R. C. et al. (1998). "An on-line archive of reconstructed hippocampal neurons". In: *Journal of neuroscience methods* 84.1, pp. 49–54. DOI: 10.1016/S0165-0270(98)00091-0. URL: http://www.sciencedirect.com/science/article/pii/S0165027098000910.

10. Costa, M. et al. (2014). "NBLAST: Rapid, sensitive comparison of neuronal structure and construction of neuron family databases". In: *bioRxiv*, p. 006346. DOI: 10.1101/006346. URL: http://biorxiv.org/content/early/2014/08/09/006346.abstract.

11. Czarnecki, Wojciech (2012). "Multilayer Neural Networks with Receptive Fields as a Model for the Neuron Reconstruction Problem". In: *Artificial Intelligence and Soft Computing, Pt II* 7268, pp. 242–250.

12. De, Jaydeep et al. (2015). "A Graph-Theoretical Approach for Tracing Filamentary Structures in Neuronal and Retinal Images". In: *IEEE Transactions on Medical Imaging* 0062.To appear. DOI: 10.1109/TMI.2015.2465962. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7219463.

13. Gillette, Todd A., Kerry M. Brown, and Giorgio A. Ascoli (2011). "The DIADEM Metric: Comparing Multiple Reconstructions of the Same Neuron". en. In: *Neuroinformatics* 9.2-3, pp. 233–245. DOI: 10.1007/s12021-011-9117-y. URL: http://link.springer.com/10.1007/s12021-011-9117-y.

14. Gillette, Todd Aaron (2015). "Comparative Topological Analysis of Neuronal Arbors via Sequence Representation and Alignment". Dissertation. George Mason University, p. 274. URL: http://digilib.gmu.edu/jspui/handle/1920/9628.

15. Gulyanon, S et al. (2015). "Three-Dimensional Neurite Tracing Under Globally Varying Contrast". In: *Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on*. IEEE, pp. 875–879.

16. Halavi, Maryam et al. (2012). "Digital Reconstructions of Neuronal Morphology: Three Decades of Research Trends". In: *Frontiers in Neuroscience* 6. DOI: 10.3389/fnins.2012.00049. URL: http://dx.doi.org/10.3389/fnins.2012.00049.

17. Halchenko, Yaroslav O. and Michael Hanke (2012). "Open is not enough. Let's take the next step: An integrated, community-driven computing platform for neuroscience". In: *Frontiers in Neuroinformatics* 6.22. DOI: 10.3389/fninf.2012.00022. URL: http://www.frontiersin.org/neuroinformatics/10.3389/fninf.2012.00022/full.

18. Hanchuan Peng Group (2015). *Vaa3D: Open-Source, Multi-dimensional Data Visualization and Analysis*. URL: http://www.vaa3d.org/ (visited on 08/07/2015).

19. Hernandez-Herrera, Paul, David Jiménez, et al. (2013). "A Harmonic Analysis View on Neuroscience Imaging". In: *Excursions in Harmonic Analysis, Volume 2*. Springer, pp. 423–450. URL: http://dx.doi.org/10.1007/978-0-8176-8379-5_21.

20. Hernandez-Herrera, Paul, Manos Papadakis, and Ioannis A Kakadiaris (2014). "Segmentation of Neurons based on One-Class Classification". In: *Biomedical Imaging (ISBI), 2014 IEEE 11th International Symposium on*. IEEE, pp. 1316–1319.

21. Hettrick, Simon et al. (2014). *UK Research Software Survey 2014*. DOI: 10.5281/zenodo.14809. URL: http://dx.doi.org/10.5281/zenodo.14809.

22. Ioannidis, John P. A. (2005). "Why Most Published Research Findings Are False". In: *PLoS Med* 2.8, e124. DOI: 10.1371/journal.pmed.0020124. URL: http://dx.doi.org/10.1371/journal.pmed.0020124.

23. Jeong, Seong-Gyun et al. (2015). *Inference of Curvilinear Structure based on Learning a Ranking Function and Graph Theory*. Research Report RR-8789. Inria Sophia Antipolis. URL: https://hal.inria.fr/hal-01214932.

24. Jiménez, David et al. (2013). "Improved automatic centerline tracing for dendritic structures". In: *2013 IEEE 10th International Symposium on Biomedical Imaging*, pp. 1050–1053. DOI: 10.1109/ISBI.2013.6556658. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6556658.

25. Joppa, L. N. et al. (2013). "Troubling Trends in Scientific Software Use". In: *Science* 340.6134, pp. 814–815. DOI: 10.1126/science.1231535. URL: http://www.sciencemag.org/cgi/doi/10.1126/science.1231535.

26. Justice Management Division (2003). *The Department of Justice Systems Development Life Cycle Guidance Document*. URL: http://www.justice.gov/archive/jmd/irm/lifecycle/table.htm (visited on 11/10/2015).

27. Lee, Ping-Chang et al. (2012). "High-throughput computer method for 3D neuronal structure reconstruction from the image stack of the Drosophila brain and its applications." In: *PLoS computational biology* 8.9, e1002658. DOI: 10.1371/journal.pcbi.1002658. URL: http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002658.

28. Liu, Yuan (2011). "The DIADEM and Beyond". In: *Neuroinformatics* 9.2-3, pp. 99–102. DOI: 10.1007/s12021-011-9102-5. URL: http://dx.doi.org/10.1007/s12021-011-9102-5.

29. Luo, Gongning et al. (2015). "Neuron anatomy structure reconstruction based on a sliding filter". In: *BMC Bioinformatics* 16.1, p. 342. DOI: 10.1186/s12859-015-0780-0. URL: http://www.biomedcentral.com/1471-2105/16/342.

30. MATLAB (2013). *version 8.1 (R2013a)*. Natick, Massachusetts: The MathWorks Inc.

31. Mayerich, David, Chris Bjornsson, Jonathan Taylor, et al. (2012). "NetMets: software for quantifying and visualizing errors in biological network segmentation." In: *BMC bioinformatics* 13 Suppl 8.Suppl 8, S7. DOI: 10.1186/1471-2105-13-S8-S7. URL: http://www.biomedcentral.com/1471-2105/13/S8/S7.

32. Mayerich, David, Chris Bjornsson, Jonothan Taylor, et al. (2011). "Metrics for comparing explicit representations of interconnected biological networks". In: *IEEE Symposium on Biological Data Visualization 2011, BioVis 2011 - Proceedings*. IEEE, pp. 79–86. DOI: 10.1109/BioVis.2011.6094051. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6094051.

33. Meijering, Erik (2010). "Neuron tracing in perspective". In: *Cytometry Part A* 77A.7, pp. 693–704. DOI: 10.1002/cyto.a.20895. URL: http://dx.doi.org/10.1002/cyto.a.20895.

34. Meng, H et al. (2015). "A case study in preserving a high energy physics application with Parrot". In: *Journal of Physics: Conference Series*. URL: http://ccl.cse.nd.edu/research/papers/tauroast-casestudy-jpcs2015.pdf.

35. Merali, Zeeya (2010). "Computational science: ...Error." In: *Nature* 467.7317, pp. 775–777. DOI: 10.1038/467775a.

36. Miller, Greg (2006). "Scientific Publishing. A Scientist's Nightmare: Software Problem Leads to Five Retractions". In: *Science* 314.5807, pp. 1856–1857. DOI: 10.1126/science.314.5807.1856. URL: http://www.sciencemag.org/cgi/doi/10.1126/science.314.5807.1856.

37. Ming, Xing et al. (2013). "Rapid reconstruction of 3D neuronal morphology from light microscopy images with augmented rayburst sampling". en. In: *PLoS ONE* 8.12. Ed. by William W. Lytton, e84557. DOI: 10.1371/journal.pone.0084557. URL: http://dx.plos.org/10.1371/journal.pone.0084557.

38. Mukherjee, Suvadip and Scott T. Acton (2013). "Vector field convolution medialness applied to neuron tracing". In: *2013 IEEE International Conference on Image Processing*. 1. IEEE, pp. 665–669. DOI: 10.1109/ICIP.2013.6738137. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6738137.

39. Mukherjee, Suvadip, Barry Condron, and Scott Acton (2014). "Tubularity Flow Field - A Technique For Automatic Neuron Segmentation." In: *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society* 24.1, pp. 374–389. DOI: 10.1109/TIP.2014.2378052. URL: http://www.ncbi.nlm.nih.gov/pubmed/25494506.

40. National Digital Information Infrastructure and Preservation Program (2013). *Preserving.exe: Toward a national strategy for software preservation*. Tech. rep. Library of Congress, pp. 1–42. URL: http://www.digitalpreservation.gov/multimedia/documents/PreservingEXE_report_final101813.pdf.

41. Parekh, Ruchi, Rubén Armañanzas, and Giorgio a. Ascoli (2015). "The importance of metadata to assess information content in digital reconstructions of neuronal morphology". In: *Cell and Tissue Research* 360.1, pp. 121–127. DOI: 10.1007/s00441-014-2103-6. URL: http://dx.doi.org/10.1007/s00441-014-2103-6.

42. Peng, Hanchuan, Alessandro Bria, et al. (2014). "Extensible visualization and analysis for multidimensional images using Vaa3D". In: *Nature Protocols* 9.1, pp. 193–208. DOI: 10.1038/nprot.2014.011. URL: http://dx.doi.org/10.1038/nprot.2014.011.

43. Peng, Hanchuan, Michael Hawrylycz, et al. (2015). "BigNeuron: Large-Scale 3D Neuron Reconstruction from Optical Microscopy Images". In: *Neuron* 87.2, pp. 252–256. DOI: 10.1016/j.neuron.2015.06.036. URL: http://dx.doi.org/10.1016/j.neuron.2015.06.036.

44. Peng, Hanchuan, Erik Meijering, and Giorgio A. Ascoli (2015). "From DIADEM to BigNeuron". In: *Neuroinformatics* 13.3, pp. 259–260. DOI: 10.1007/s12021-015-9270-9. URL: http://dx.doi.org/10.1007/s12021-015-9270-9.

45. Peng, Hanchuan, Zongcai Ruan, et al. (2010). "V3D enables real-time 3D visualization and quantitative analysis of large-scale biological image data sets". In: *Nature Biotechnology* 28.4, pp. 348–353. DOI: 10.1038/nbt.1612. URL: http://dx.doi.org/10.1038/nbt.1612.

46. Poldrack, Russell A (2011). "Inferring Mental States from Neuroimaging Data: From Reverse Inference to Large-Scale Decoding". In: *Neuron* 72.5, pp. 692–697. DOI: 10.1016/j.neuron.2011.11.001. URL: http://dx.doi.org/10.1016/j.neuron.2011.11.001.

47. Santamarıa-Pang, Alberto et al. (2015). "Automatic Morphological Reconstruction of Neurons from Multiphoton and Confocal Microscopy Images Using 3D Tubular Models". en. In: *Neuroinformatics*. DOI: 10.1007/s12021-014-9253-2. URL: http://link.springer.com/10.1007/s12021-014-9253-2.

48. Spolsky, Joel (2000). *Joel on Software: Things You Should Never Do, Part I*. URL: http://www.joelonsoftware.com/articles/fog0000000069.html (visited on 08/04/2015).

49. Thain, Douglas, Peter Ivie, and Haiyan Meng (2015). "Techniques for Preserving Scientific Software Executions: Preserve the Mess or Encourage Cleanliness?" In: *Proceedings of the 12th International Conference on Digital Preservation (iPres)*. Chapel Hill, USA. DOI: 10.7274/R0CZ353M.

50. Torben-Nielsen, Benjamin (2014). "An Efficient and Extendable Python Library to Analyze Neuronal Morphologies". en. In: *Neuroinformatics* 12.4, pp. 619–622. DOI: 10.1007/s12021-014-9232-7. URL: http://link.springer.com/10.1007/s12021-014-9232-7.

51. Wilson, Gregory (2006). "Where's the Real Bottleneck in Scientific Computing?" In: *American Scientist* 94.1, p. 5. DOI: 10.1511/2006.57.3473.

52. Wilson, Greg et al. (2014). "Best Practices for Scientific Computing". In: *PLoS Biology* 12.1. Ed. by Jonathan A. Eisen, e1001745. DOI: 10.1371/journal.pbio.1001745. URL: http://dx.plos.org/10.1371/journal.pbio.1001745.

53. Xiao, H. and H. Peng (2013). "APP2: automatic tracing of 3D neuron morphology based on hierarchical pruning of a gray-weighted image distance-tree". In: *Bioinformatics* 29.11, pp. 1448–1454. DOI: 10.1093/bioinformatics/btt170. URL: http://bioinformatics.oxfordjournals.org/cgi/doi/10.1093/bioinformatics/btt170.

54. Xie, Jun et al. (2010). "Automatic neuron tracing in volumetric microscopy images with anisotropic path searching". In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2010*. Springer, pp. 472–479. URL: http://link.springer.com/chapter/10.1007/978-3-642-15745-5_58.

55. — (2011). "Anisotropic path searching for automatic neuron reconstruction". In: *Medical image analysis* 15.5, pp. 680–689. URL: http://www.sciencedirect.com/science/article/pii/S1361841511000776.