DYNAMO: AN OPEN SOURCE PYTHON APPLICATION FOR COMPREHENSIVE STRUCTURAL AND FUNCTIONAL ANALYSIS OF DEVELOPING NEURONS.

by

Patrick Coleman

B.A., The University of British Columbia, 2018

B. Sc., The University of Adelaide, 2008

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Neuroscience)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

March 2021

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, a thesis entitled:

Dynamo: An open source python application for comprehensive structural and functional analysis of developing neurons.

submitted by Patrick Coleman in partial fulfillment of the requirements for

the degree of Master of Science

in Neuroscience

Examining Committee:

Dr. Kurt Haas, Department of Cellular & Physiological Sciences
Supervisor

Dr. Jason Snyder, Department of Psychology
Supervisory Committee Member

Dr. Timothy H. Murphy, Department of Psychiatry
Supervisory Committee Member

Abstract

Neurons are notoriously complex cells, generating high-frequency functional voltage dynamics inside a tree-like branching arbor structure. Developmental cellular neuroscience examines the processes underlying how these cells form their structure, the causal relationship between growth and firing dynamics, and how disruption of this system may result in developmental disorders such as Epilepsy and Autism Spectrum Disorder (Parenti et al., 2020). As more is discovered, it becomes clear that additional information is needed to paint the full picture of the environment and rules underlying development. Improvements have been made in terms of hardware (such as Light sheet or Two-photon microscopes), and biology (e.g. voltage- and calcium- sensors) which require similar improvements in the software and analysis tools to explore the raw data and find the information it contains. This research summarizes current software tools for developmental neuroscience, and combines these with new analysis and visualization features in an open-source python application called Dynamo. The features of Dynamo are explained, and the process of going from raw recordings to scientific results is shown using example neuron recordings from development of Xenopus laevis *in vivo* tectal neurons.

Lay Summary

Neurons are considered the building blocks of your brain, but unlike simple blocks, each neuron has a complicated tree-like structure, wiring itself to nearby neurons to pass around electrical signals that encode our thoughts. Developmental neuroscience studies how these structures first form, how each neuron establishes their role in the active system around them, and what can cause this to go wrong. A huge amount of data is required to paint a complete picture of everything going on during development, and analysis of that data is arduous. It is often restricted to only a small feature within the picture, and even automated approaches require manual cleanup before the results are accurate. This study implements Dynamo, an open-source python application that unifies a collection of existing isolated analysis approaches, offers faster editing, and some new techniques for analysis.

Preface

Chapters 1 and 3 are my own work, giving a background into areas of technological progress in developmental neuroscience that are relevant to this project (chapter 1) and a summary of the approach taken and future areas of research (chapter 3).

Chapter 2 is divided into three sections, corresponding to the aims of the thesis and three distinct phases of the overall body of work. All three sections cover work performed at the University of British Columbia's Centre for Brain Health, conducted under the supervision of Professor Kurt Haas. All animal experiments were performed in accordance with the University of British Columbia's Animal Care Committee (A19-0297). All procedures strictly adhere to the guidelines issued by the Canadian Council for Animal Care.

Section 2.1 covers work initially performed by former graduate students Kaspar Podgorski and Serhiy Opushnyev, who recorded 2-photon time-series imaging of *in vivo* neurons, and made initial analysis in MATLAB software. I rewrote the software to Python, reproducing the analysis for inclusion in (Podgorski et al., 2021).

Section 2.2 details work performed in collaboration with Peter Hogg (Ph. D. student), who performed all neuron preparation and imaging. I modified the Dynamo analysis software to add the required functionality, as well as expedite the reconstruction and cross-time point registration. Peter Hogg and I then joined efforts to produce the custom analysis required for that projection.

Section 2.3 presents research performed by Dr. Kelly Sakaki (post-doc), Tristan Dellazizzo Toth (Ph. D. student) and me, covering comprehensive structural and functional

dynamic morphometrics. Dr Sakaki built the original AOD microscope, and performed initial fluorescent construct testing and data visualization. Tristan optimized the fluorescent constructs, plus performed the neuron preparation and imaging, and I performed all analysis of the resulting data, translating the tree structures to Dynamo, incorporating the functional traces, and writing custom analysis code for that project.

Table of Contents

List of Tables

List of Figures

List of Supplementary Material

Glossary

Term	Definition
2P	Two photon microscopy
AOD	Acousto-optic deflectors, used to direct a light path to a particular point of 3D space
bAP	Back-propagating action potentials, depolarizing events travelling from the Soma back along dendrites.
Branch	A collection of connected POI, e.g. dendrites/axon/spine/filopodia
Continuous Probe (CP)	Stimulation protocol that does not induce plasticity.
Dynamic Morphometrics	The study of arbor metrics over time during development.
Filopodium (pl. Filopodia)	Small actin-filled protrusion, often transient, may contain synapses and transition into spines.
Monte Carlo (MC) Clustering	A measure of clustering of POI on a tree, by observing where it lies compared to multiple random POI locations.
Motility	Change in length of a branch/filopodium across time
NND (Monte Carlo)	Distance to nearest neighbor POI of a particular type.
NND (Trace filter)	Nonnegative deconvolution, fitting expected indicator dynamics to $\Delta F/F_0$ traces.
Point of Interest (POI)	A single point in an arbor tree, a position in 3D space connected to other POI along branches
Python	Programming language for the new Dynamo software.
Reconstruction	Creation of a tree (POI/branches/locations) from a stack.
Registration	Alignment across time of POI that represent the same point on the tree
Root	Base of the tree, usually located at the Soma.
Spaced Training (ST)	Stimulation protocol observed to induce LTP in neurons.
Stack	Single imaging volume – a 3D ‘stack’ of multiple 2D images.
Trace	High sample-rate data capturing functional dynamics at a POI
Tree	An acyclic collection of connected branches, representing the arbor.
Voxel	A single value in a 3D image – similar to a pixel in 2D.
Z-projection	Flattening a 3D stack into 2D, by projecting along the focal (Z) axis.

Acknowledgements

Firstly, I must thank my supervisor and mentor Dr Kurt Haas, for taking on someone with my lack of biological background, training me not only in neuroscience but also navigating academic life. I'll always be grateful for the freedoms you give your lab to investigate what they find interesting, sharing their excitement, and helping ensure high quality, relevant results can be found, and presented well.

A big thanks must also go to my two collaborators and friends in the imaging team: Peter Hogg and Tristan Dellazizzo Toth. Whether it be sharing that small office space, more recently virtual coffee mornings, or even the soccer matches, you've both played a large role in improving my time as a student - both in terms of productivity through sharing ideas/solutions, but also general enjoyment.

To the cellular side of the lab – Sin, Warren, and Fabian – I want to express my gratitude for everything I've learned over the last years. I remember struggling to follow along to the genetic/wet-lab side of things in early lab meetings, but you were all extremely patient with my beginner questions, and always seemed happy and skillful at helping me fill the holes in my knowledge.

Thanks go to my academic support crews: firstly, to Dr Tim Murphy and Dr Jason Snyder for assisting with the Masters process, as well as Dr Hilary Coleman and Dr Bodie Ashton for their experience with academia and giving any answers or tips on how best to make it through life as a grad student.

Final thanks must go to my parents Meredith and Mark, grandparents Diana and John, and most of all to Sri, for the continued love and support over the last years, for the constant chats and making sure everything outside research was running smoothly, even amidst a pandemic.

Dedication

I dedicate this work to my big sister and academic role model Dr. Hilary Coleman. She is the embodiment of good science: hard-working, unwaveringly diligent, and driven by a desire to find out truths about our world. I'm sure the next generation of Australia's chemists you are teaching will be inspired to produce quality research, as I have been here (hopefully…).

And to all the tadpoles who made this research possible – may their sacrifices not be just for understanding, but also for improving this world.

Chapter 1: Introduction

1.1 Developmental Neuroscience

Neurons are one of the most complicated cells, growing into rich, branching axonal and dendritic structural arbors, while also connecting themselves into a fast-acting functional network, balancing levels of ions, neurotransmitters, and internal and external secondary messenger dynamic systems with chaotic interactions. Developmental neuroscience studies this progression, investigating the rules that govern the establishment of the initial neurons and network, as well as how genetic or environmental interruptions during development can play a role in neurodevelopmental disorders such as Rett's syndrome, Epilepsy, Autism Spectrum Disorder, or Schizophrenia. (Johnston et al., 2005; Parenti et al., 2020). The field itself is large, with many avenues of research, however this study concentrates on the hardware and software tooling that assists in making developmental findings.

1.2 Full-arbor *in vivo* 2P imaging

Many results in developmental neuroscience have been accompanied by technological advances that were required to make them possible to observe. Flat 2D *in vitro* recordings were important for early axonal and dendritic guidance cues (Hedgecock et al., 1985), though these lack a few fundamental aspects of neuronal development that are required to paint the full picture of arbor development (Azari & Reynolds, 2016).

Firstly, the complete experiential background of a living organism is missing: both the firing history of the neuron being recorded (Borges & Berry, 1978), plus surrounding neurons in the network (Deppmann et al., 2008), as well as any nearby neurotransmitters or hormones (Chiu

& Cline, 2010) that are theorized to affect the observed growth. While natural conditions can be mimicked, the most representative way to capture a comprehensive developmental context is performing experiments *in vivo*.

Another important factor to consider is that neuronal arbors are complex, three dimensional structures that grow spatially in all directions. Some neurons such as the spiny stellate neurons in mice barrel cortex have biased directional growth (Nakazawa et al., 2018), and pyramidal cells will have a self-avoiding dendritic 'sampling' arbor that covers much of the space surrounding the soma (Stuart et al., 2012). It is important to fully capture this - any planar imaging technique limited to two dimensions will be missing one third of the available spatial information, and metrics such as branch distance, growth angles, or crossing counts will be incomplete.

To address these, all studies discussed in this thesis have utilized advances in comprehensive (i.e. full-neuron) *in vivo* two-photon (2P) imaging. By exciting fluorescent reporter proteins or dyes with a laser emitting photons at half their excitation energy (i.e. twice the wavelength), excitation only happens when two photons are simultaneously absorbed by the fluorescent target, which only occurs at the high photon density at the focal point. This allows limited out-of-focal-plane bleaching and phototoxicity, as well as deeper imaging that is less prone to scattering (Helmchen & Denk, 2005), at the cost of more photons required as simultaneous excitation probability is low, and a lower resolution. 2P imaging is used to study neurons by injecting them with fluorescent dyes such as Oregon Green BAPTA-1 AM (Stosiek et al., 2003), or through targeted expression of fluorescent proteins such as optimized variants of GCaMP (Nakai et al., 2001) through the use of single-cell electroporation (Haas et al., 2001).

By combining these, studies in dynamic morphometrics that examine purely structural development can fluoresce a single neuron *in vivo*, and perform volumetric imaging of the space encompassing it at desired intervals, anywhere down to the order of 5 minutes (section 2.2.1). By additionally utilizing random access (Sakaki et al., 2020) or projection microscopy (Kazemipour et al., 2019), higher temporal-resolution data such as voltage or calcium levels can be recorded along the arbor, for full structural and functional dynamic morphometrics.

1.3 Arbor Reconstruction

Every volumetric neuron recording is accompanied by the labor-intensive process of *reconstruction* - that is, converting the recorded data (a volume of per-voxel fluorescent intensities) into the meaningful, systematic information representing the connected arbor structure of the neuron, complete with soma position, axonal and dendritic branch locations, lengths and radii, as well as the parent-child tree connectivity between them. While the early arbor reconstructions of Cajal (Ramon y Cajal, 1911) were considered artistic and motivated many generations of neuroscience researchers, scientific manual reconstruction is timely and repetitive (van der Glaser & Loos, 1965), so understandably effort has been put towards automating the process. Recent success has been made using both statistical approaches as well as modern machine-learning-assisted image segmentation techniques, and progress continues to reduce remaining errors, as well as increase precision in noisy or densely populated neuronal environments (Acciai et al., 2016; Donohue & Ascoli, 2011). This is particularly important in developmental neuroscience, as reconstruction must be performed repeatedly at numerous times in development, and many of the features of interest are quite small so it must be accurate to a fine level of detail.

1.4 Registering multiple reconstructions

Unfortunately, while reconstruction of a single arbor is already a difficult problem, once solved there is still more work to do when examining developmental properties, as developmental neuroscience is interested with how the arbor changes over time. While single arbor reconstruction is a pre-requisite, work remains on techniques for *registration* – i.e. matching corresponding points across multiple reconstructions, to see how it evolves over time. Ideal for this is a recording technique which remembers an identifiable branch structure between times of recording, and stores these correspondences in a time-aware format such as SWCX (Nanda et al., 2018). When not available, post-hoc registration is required. Historically, this was done manually as it is a simple task for humans, albeit time-consuming. Methods exist to provide some automated registration assisted (Chalmers et al., 2016), however the recommended option when possible is to retain this information between recordings.

1.5 Dynamic Morphometrics

Once arbors are recorded, reconstructed, and registered, the next step in research is to describe the developmental arbor changes with objective measures, a process called *Dynamic Morphometrics.* Whether simple measures such as branch count or total dendritic branch length, or more complicated measures like fractal geometries (Ristanović et al., 2002), these can be calculated throughout development, and their progression analyzed under behavioural or genetic conditions. Dynamic morphometrics also includes metrics not just of individual trees, but the *changes* between them. Branches and filopodia or synapses have a developmental history outside of a single snapshot in time, and these addition/subtraction/extension/retraction rates and survival characteristics can also be measured and compared.

1.6 Functional Analysis

Structural morphometrics have been used to discover multiple principles behind the rules governing arbor development in neurons, but the technology for recording these precise structures has traditionally only allowed for time resolutions on the order of minutes to hours. This permits analysis of genetic or environmental effects on growth, but is too slow to capture any functional dynamics arising from the activity of the neuron or subsections of its arbor. Neurons are believed to have complicated compartmentalized calculations (London & Häusser, 2005), but only recently has technology been established to answer questions about how a neuron develops an arbor to process information in this way.

Through use of direct voltage measures, electrophysiology addressed this early by permitting direct measurement of activity at a few locations in an arbor (Jäckel et al., 2017), however performing this *in vivo* is very complicated in developing neurons for anything more than somatic recordings (Jouhanneau & Poulet, 2019). Dynamic fluorescent indicators such as GECIs or VSFPs (Zhang et al., 2011) afford non-invasive optical recording of activity, but comprehensive 3D scanning of localized activity across an entire dendritic arbor is still an area of active research. For planar or small volumes able to be scanned by single-photon microscopes, light sheet microscopy has been able to produce recordings at high temporal and spatial resolution (Haslehurst et al., 2018). For deeper 3D imaging, techniques such as random-access scanning (Sakaki et al., 2020) or linear projection (Kazemipour et al., 2019) have increased scanning rates to the kilohertz ranges, so it is anticipated that experiments containing both the full structural and functional development of a neuron will become more common. This results in large datasets (in the order of gigabytes of raw data per scan), which will soon outgrow traditional analysis tools that are currently aimed at smaller recordings of structural or functional

analysis in isolation. Feature sets of both will need to be combined before a more complete understanding of neuron development can be achieved.

1.7 Data Visualization and validation.

One under-appreciated area of neuro-developmental research is that of data representation and visualization. As discussed, comprehensive structural and functional recordings of even a single neuron during development will soon surpass tens to hundreds of gigabytes of data across all time points. Automated, algorithmic approaches still require much manual intervention (Acciai et al., 2016) which we have also found experimentally, so a researcher's visualization and interpretability of the results is also an important aim of this software. This helps increase confidence in findings, as otherwise errors in automated processing can easily pass unnoticed. The importance of interpretability is even higher when considering that some discoveries are serendipitous, where researchers manually explore and understand a dataset to find unexpected patterns.

The nature of comprehensive structural and functional recordings of arbor dynamics makes things particularly complicated here. Data recorded includes many modalities: three dimensions of real-world spatial locations, one dimension of color (e.g. arbor dye, plus fluorescent structural antibody, plus functional indicator), plus a multi-scaled time dimension (from kilohertz to per-day recordings), all overlaid on top a complicated dynamic tree structure. Work is underway improving the user interfaces, including utilizing virtual reality to better interact with the 3D environments (Wang et al., 2019), but there is still much room for improvement in designing software that is efficient and enjoyable to use while performing dynamic morphometrics research.

1.8 Research aims

Dynamic morphometrics is a fruitful avenue of research into developmental neuroscience, though improved automated algorithms, manual tooling, and analysis are required to help process the ever-increasing volumes of data being recorded for each project. The objective of this research project is to unify these into “Dynamo”, an open-source software project that is simple and free to use now and into the future for this genre of experiments. To assist in planning its creation, the research has been split into three successive aims, each covering both Dynamo development as well as validation with research data.

Aim 1: Migrate existing dynamic morphometrics software from MATLAB to python.

Previous lab data was collected, processed, and analyzed using a diverse range of tools, in MATLAB (primarily) and LabView. Due to licensing and version upgrades, these became difficult to manage, upgrade, and generate reproducible analysis pipelines. To address this, the first goal of the research was to unify and rewrite portions of the initial project into python. The application would be validated by comparing the resulting visualizations and analysis with existing structural morphology data and results from (Podgorski et al., 2021)**.**

Aim 2: Add new functionality to help with drawing, structural morphology tracking, and analysis.

Once the initial rewrite was shown to work, the intention was to use Dynamo for new research within the lab. After discussion with researchers who had used the earlier version, and those planning the new experiments, a number of desired features were identified, including: (a) Importing of machine-assisted reconstructed arbors, (b) simplified manual drawing and visualization, (c) adding automated and manual across-time point registration of branches, and

(d) exporting results for custom analysis. Discussion of these features and application to data sets will be used in two papers in preparation.

Aim 3: Support functional analysis.

Whereas traditional dynamic morphometric analysis was performed primarily on structural changes, Dynamo will be extended to add additional support for functional analysis. Advances in comprehensive *in vivo* imaging have allowed for both internal (e.g. calcium) and external (e.g. glutamate) functional activity measures, and new findings can be made by examining the interplay between this activity and existing structural morphology changes. The final goal of the research will support the storage and retrieval of these functional data series, as well as their visualization and analysis in the context of developing arbor structures. It will be applied to new data sets in (Podgorski et al., 2021; Sakaki et al., 2020)**.**

Separate to these aims, Dynamo is intended as a free, open-source python application. To achieve this, the final version must be made available through a GitHub repository (hosted at https://github.com/ubcbraincircuits/pyDynamo) and installable with pip.

Chapter 2: Dynamo

Situated in a lab that performs dynamic morphometrics on young sensory neurons to explore rules of neuronal development, the ability of our microscopes to collect data had outgrown the ability of our software to analyze the results. Additionally, the manual effort required made quantification of the metrics prohibitive at large scale, and larger & more complicated datasets were being planned, so this section of the thesis covers the development of Dynamo – initially a rewrite that needed to reproduce results on existing data, then supporting and improving both structural and functional analysis of newly recorded data.

2.1 Initial rewrite and application to existing data

2.1.1 Existing MATLAB software

The preliminary version of Dynamo itself was MATLAB software to control a 2P microscope and analyze the resulting recordings. Unfortunately, that microscope had been replaced by one controlled in LabView, the only researcher familiar with its execution had since left the lab, and sections of the code no longer worked in newer MATLAB versions. Due to this, it was decided that before any *new* analysis could be performed with Dynamo, it would first need to be rewritten in an open-source language, supporting old features and data sets but with a simpler interface.

2.1.2 Basic user interface

Due to its extensive support of scientific libraries and general simplicity and popularity in the scientific community (Muller et al., 2015), the language chosen for the new version of Dynamo was Python. To maintain similarity to the original, the user interface was written using PyQt5

and set up so that each time point's neuron volume was shown in its own window, with one Z index shown, synchronized between all windows. The user can then pan and zoom through all three dimensions, as well as manually draw the trees using the mouse. In the new version, a window was added to list all the volumes loaded, show/hide their corresponding images, and add/delete/reorder as necessary (see Figure 1).

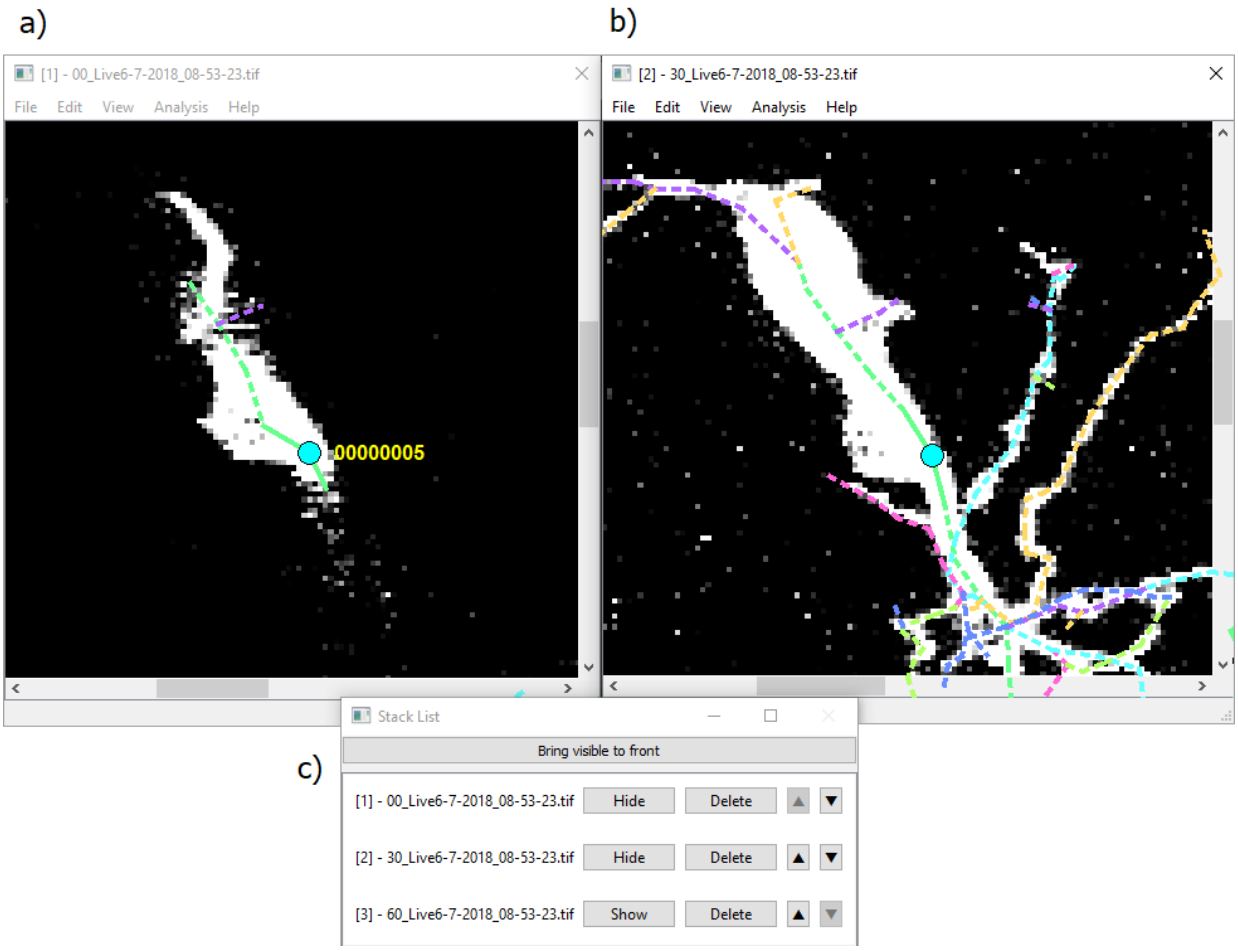

Figure 1: Example drawing interface, showing a) drawing on a regular volume showing one plane, b) a subsequent volume with Z projection, and c) the list of imaging volumes within this project.

2.1.3 Register-with-Adjust

The first complicated dynamic morphometrics algorithm that required implementation in Python was *registration with adjust* – that is, given an existing fully drawn arbor at a particular time point, using it to initialize the arbor at a later time point by finding positions within the volume that are a close visual match the initial time's positions.

This is achieved by repeating the following steps: first, the entire tree — all points of interest (POI), their locations plus parent-child interactions — is copied from the initial time point. Then starting at the arbor root (i.e. Soma):

- A small sub-volume around the point is extracted – 21x21x9 px, or roughly equivalent to a 13um cube at our default resolution. A similar sub-volume is extracted at the estimated position for the POI in the new volume.
- Both are flattened into 21x21 images by taking the maximum values along the Z axis.
- The scipy.optimize library (Virtanen et al., 2020) is used to find the optimal XY translation and rotation to align the two images.
- If they match within error thresholds, the estimated POI position is moved by the calculated XY offset. Finally, of the 9 Z planes considered, the POI is moved to that which results in the closest image. Note that the entire subtree at this point is translated by the same optimized XYZ offsets.
- If they don't match well, next a wider neighborhood is examined by repeating the same steps, now with a 51x51x9 px sub-volume. If no match is again found, this point is considered unable to be registered, and skipped.

All child POI are recursively aligned, until they are placed at their most similar positions. This approach is very good for initial placement of POI, as it will adjust for minor drift and rotation, and be much faster than re-drawing the tree by hand. Minor adjustments are expected though, particularly at distal dendrites which can be dim and/or thin compared to the soma region.

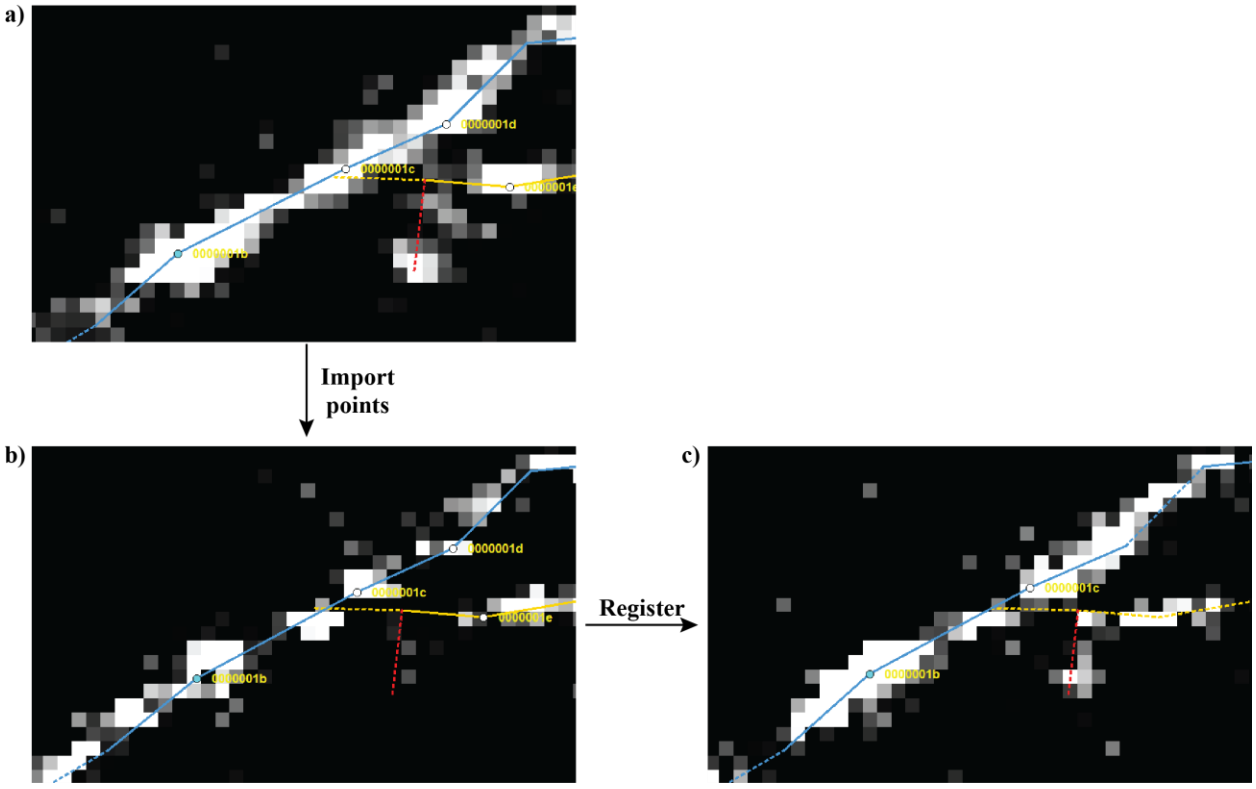

Figure 2: Registration with adjust. a) The original drawn time point. b) Importing the POI locations onto a subsequent volume. c) Location of points after registration, to positions closer matching the previous volume.

2.1.4 Simple metric analysis

The basic analysis performed by the original Dynamo was migrated to Python, and presented in a new analysis dialog which supports export to .csv, for external graphing and submission to journals. Some simple metrics were added, which can be calculated either per-tree or per-branch. The following summarize the entire set available:

Tree metrics	
TDBL	Total dendritic branch length (μm)
Point count	Number of POI in the tree.
Branch count	Number of branches, including filopodia.
Filopodia count	Number of filopodia (interstitial, terminal, and total)
Filopodia density	Filopodia count per μm (interstitial, terminal, and total)
Sholl critical value	Radius of maximal branch crossing (μm)
Sholl dendrite max	Number of crossings at the critical value.
Branches Added	Count of new branches at this time point
Branches Removed	Count of branches absent since last time point.
Branches Extended	Count of branches longer than last time point.
Branches Retracted	Count of branches shorter than last time point.

Table 1: Per-tree analysis metrics available in Dynamo

Branch metrics	
Length	Length of the branch (μm)
Type	Whether the branch is a filopodial (terminal/interstitial) or not.
Branch order	Both shaft and centrifugal branch orders.
Is axon?	Whether a point on the branch has been annotated ‘axon’.
Is basal?	Whether a point on the branch has been annotated ‘basal.
Parent Branch	ID of the branch this branch grows off.
Parent Point	ID of the POI at the base of this branch.

Table 2: Per-branch analysis metrics available in Dynamo

2.1.5 Motility measures

One of the most important tools essential to the domain of dynamic morphometrics is that of motility analysis: i.e. all the changes in arbor (axonal/dendritic branches, as well as filopodia/synapses), and quantifying the additions/subtractions/extensions and retractions caused over time during development.

Once all the POI (points, and as a result also branches) are registered (see 2.1.3), functionality was added to calculate the branch length in real-world units (μm) by summing up the length of all segments along it. Motility can then be calculated for each branch as follows, given some value M (μm) of minimal motility to be considered above noise.

Branch at t = -1	**Branch at t = 0**	**Type**	**Motility size**
Did not exist	Exists, length L_0	Added	$+ L_0$
Existed, length L_{-1}	Does not exist	Subtracted	$- L_{-1}$
"	$L_0 > L_{-1} + M$	Extended	$L_0 - L_{-1}$
"	$L_0 < L_{-1} - M$	Retracted	$L_0 - L_{-1}$
"	$L_{-1} - M < L_0 < L_{-1} + M$	Stable	$L_0 - L_{-1}$

Table 3: Classification guide for branches over time, including minimal motility M.

As with the earlier Dynamo, graphs were added to visualize these motility measures as dots on top of the 3D arbor structure, positioned at the branch tip. Dot color was used to indicate type, and radius to indicate (absolute) motility. As an optional extra, a second 2D version of this same motility plot was added, this time using a dendrogram to instead draw the arbor in a plane, with POI positioned by distance from soma. As with the simple analysis, motility calculations per branch per time point can also be exported for external processing.

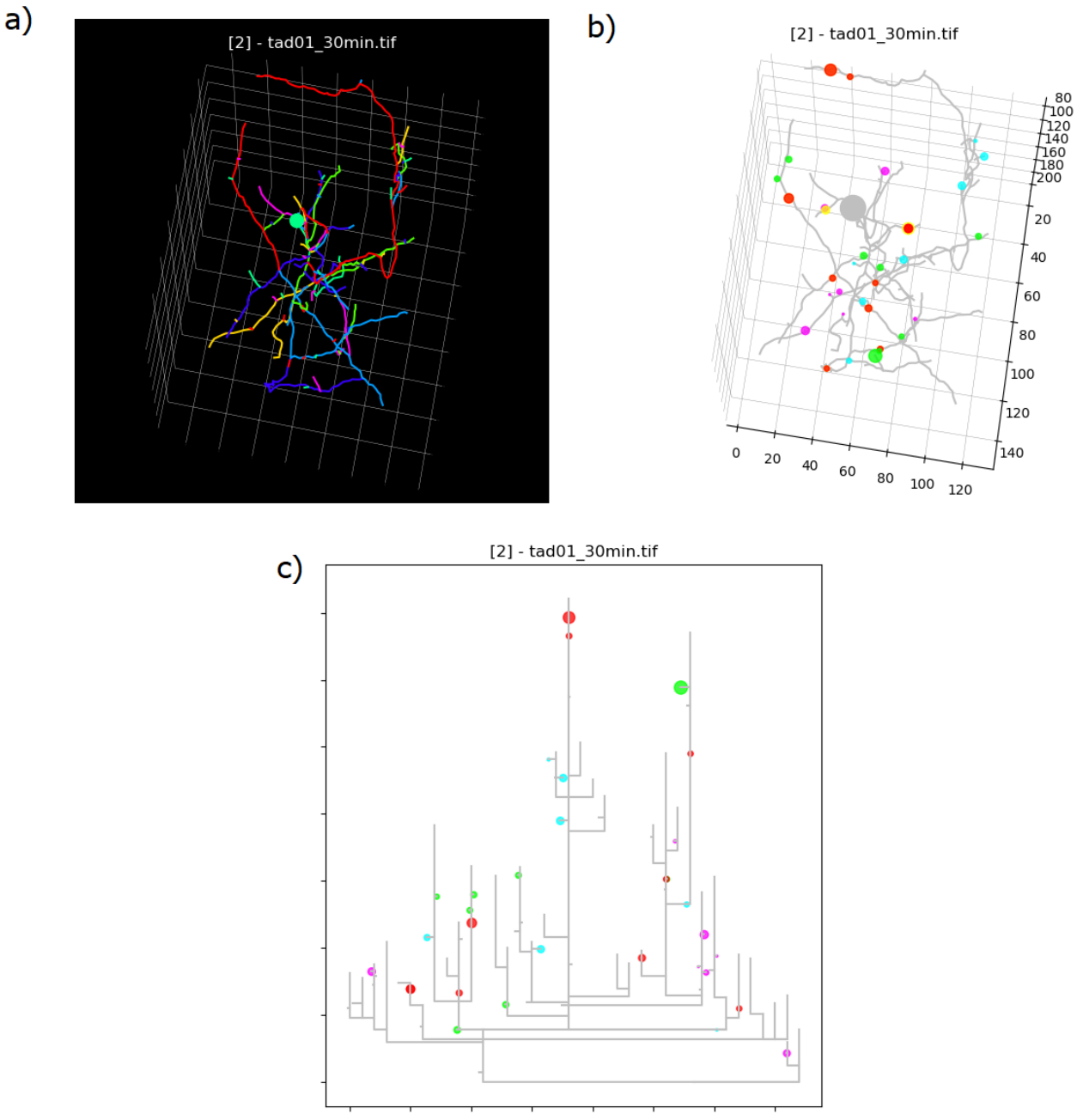

Figure 3: Motility plots, including a) Original 3D arbor, b) Overlaid motility from the previous time point, showing additions (green), subtractions (red), extensions (blue) and retractions (purple), with radius representing size change. c) Equivalent 2D dendrogram plot.

2.1.6 Monte Carlo clustering

One analysis technique of interest in dynamic morphometrics is in investigating whether *target* POI of interest (either structural changes or functional activity) are clustered or anti-clustered; that is, whether they happen significantly closer to other *source* POI of interest, or perhaps further away. This could be due to both extracellular cues (e.g. neurotransmitter dispersion) or intracellular (e.g. voltage, calcium, secondary messengers). As exact statistical modelling of location 'similarity' is difficult, one approach is to utilize Monte Carlo simulation (Raychaudhuri, 2008), i.e. repeated random sampling. First, the distance metric between the source and target sets is measured - a number of options for this are supported, but by default we calculate the median intracellular distance between a target POI and its nearest source neighbor POI ('median nearest neighbor distance (NND)'). Next, we simulate multiple runs of randomly distributing the source locations across the arbor, and measure the same metric, to estimate the random distribution that the original metric was pulled from. After enough runs, provided the random distribution can be estimated (in our case, by a skewed normal – see Figure 4b), the exact likelihood that metric was drawn from the random distribution (i.e. p-value) can be calculated.

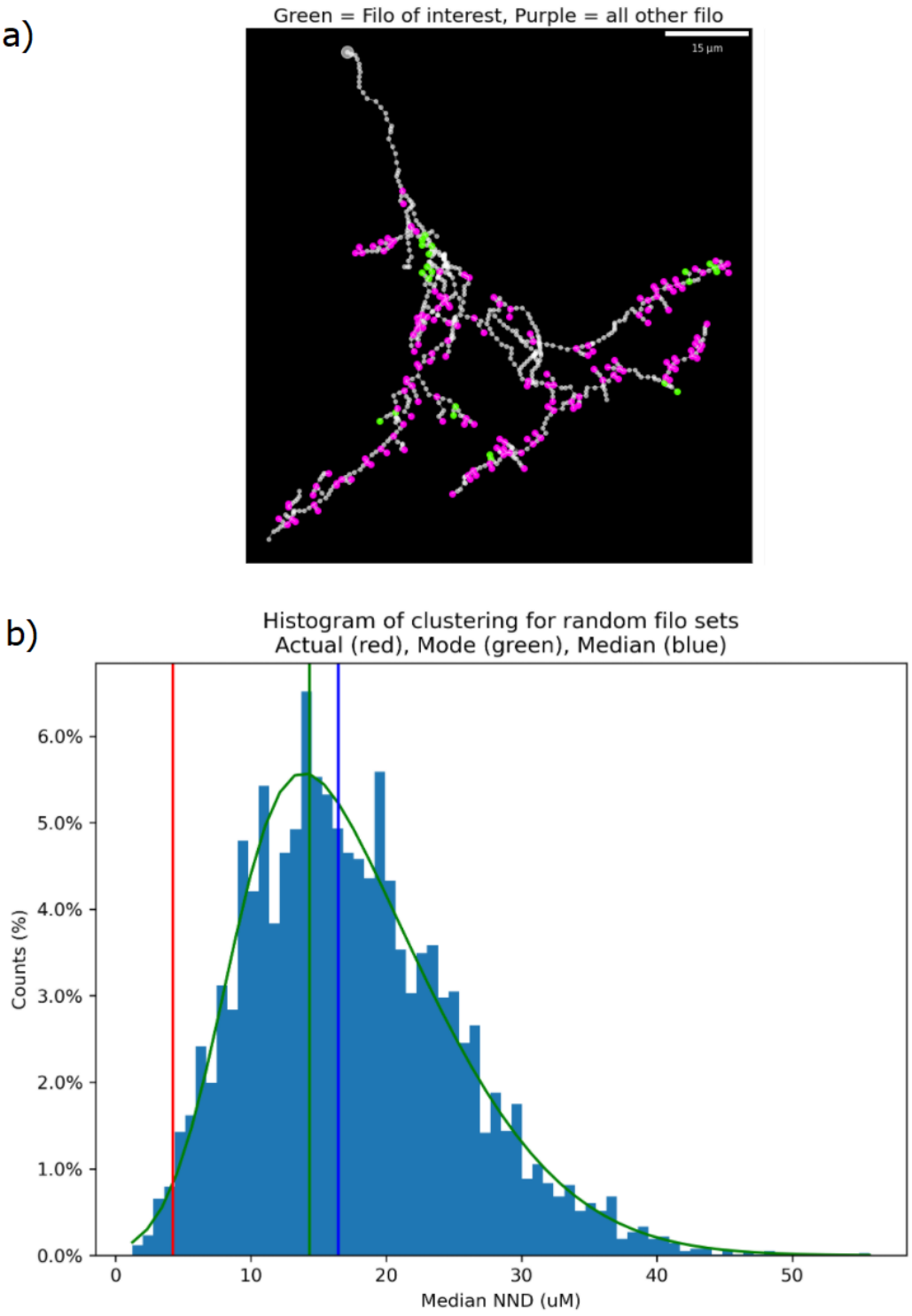

Figure 4: Monte Carlo clustering, showing a) Locations of additions (green) and other (magenta) filopodia, and b) Metric recorded from 10000 runs, with distribution modelled, showing mean (blue), median (green) and actual metric (red).

Numerous options are supported for the analysis: for example, using intra- vs extracellular distance, taking the mean or median nearest neighbor distance, or whether to include or exclude terminal filopodia in the analysis. The number of Monte Carlo runs can also be set – experimentally it was found 10000 iterations ran quickly (in the order of minutes) and estimated the distribution well using a skewed normal distribution, providing all possible pairwise POI-POI distances were precalculated once at the start. This can be seen in Figure 4 - where additions (green POI) were used as both source and target POI. Subfigure (b) shows the distribution of nearest neighbor distance metric across 10000 runs, as well as the modelled skewed normal distribution, allowing likelihood calculation for the measured metric – in this case, it is significantly lower than the mean, suggesting additions were clustered for this neuron.

2.1.7 Application to data: Filopodial clustering.

The first aim of the Dynamo project was to rewrite the software in python and use for existing data sets, so our first presented experiment analysis looks at analysing patterns in the structural changes of neurons undergoing a learning protocol. It is known that in mature neurons, dendritic spine additions will cluster after learning (Frank et al., 2018). Using our two-photon setup and dynamic morphometrics techniques, we can examine similar dynamics of filopodia *in vivo*.

2.1.7.1 Tadpole preparation

Freely-swimming, albino *Xenopus laevis* tadpoles were reared in 10% Steinberg's solution. Seven days post-fertilization, we perform single cell electroporation (Haas et al., 2001) in the left dorsolateral optic tectum, to express the GCaMP6m plasmid, a green calcium indicator for functional dynamics. Tadpoles were then screened for successful transfection visually under a

fluorescence stereomicroscope (Leica MZ16 F), and only tadpoles with successful expression in type 13b pyramidal neurons were used in the study.

Just prior to scanning, stage 50 tadpoles are paralyzed for 5 minutes in a bath of 2 mM panacronium dibromide (0693/50, Tocris) before being placed in an imaging chamber perfused with Steinberg's solution. The chamber itself stabilizes the head of the tadpole with a square sheet of cellulose acetate, and includes a red LED adjacent to the eye of the tadpole, which is turned on and off to provide visual stimulus.

2.1.7.2 Experimental protocol

This analysis of filopodial addition clustering was performed within a larger experiment also looking at somatic and filopodial responsiveness after LTP induction, so the protocol includes both regular volumetric scans of a neuron, as well as a stimulation procedure. Five three-dimensional imaging volumes surrounding the transfected neuron were rastered, at 30-minute intervals. Between the second and third volume, a ‘spaced training’ (ST) stimulation pattern is given, consisting of three 5-minute periods of high-frequency OFF stimuli (50ms each, at 0.3hz) spaced by 5 minutes of solid red light. Spaced training has been shown to induce LTP in these neurons, and can be verified through inspection of somatic traces. Between the other epochs, a neutral ‘continuous probe’ OFF stimulation is given (50ms, once per minute), shown to have no potentiation effect (Dunfield & Haas, 2010). Each volume then had arbors manually reconstructed in MATLAB by the researcher, and stored as .mat files.

2.1.7.3 Results: Filopodial additions and extensions are clustered post-training

The existing datasets of neuronal arbors undergoing training were captured with the MATLAB version of Dynamo, so first functionality was added to the python Dynamo to load these and

rewrite the data to its own internal format. The python Dynamo was then to first detect locations of all additions at each epoch. Next, Monte Carlo clustering was performed, measuring the distance of each addition immediately post-training to the nearest other addition. We can calculate the clustering index for each neuron, by calculating what percentage of changes (e.g. Additions, …) were close to other changes of the same type, compared to what percentage would be expected by chance (over 10,000 random placements). By calculating this across all the neurons, the locations of additions and extensions were found to significantly cluster to themselves. Subtractions and retractions found no such difference.

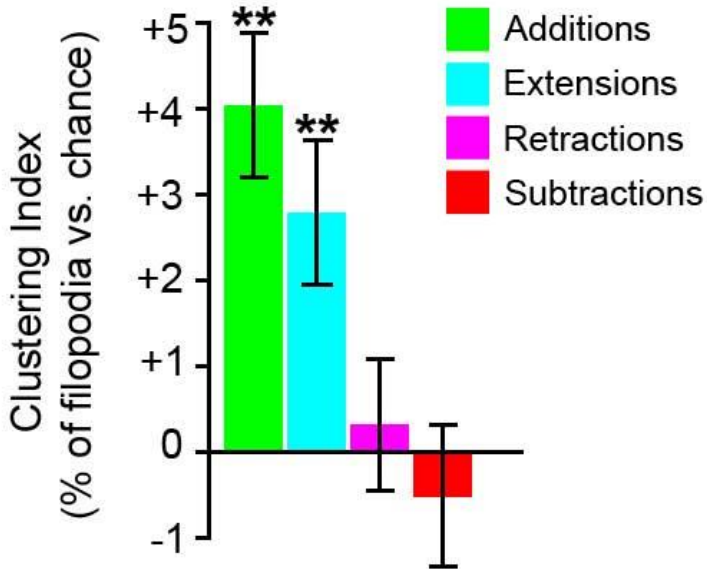

*Figure 5: Clutering of types of filopodia to their nearest neighbor, compared to average from 10,000 random runs. Additions and Extensions were found to cluster. **p<0.01, using t-test for n=1504 additions, 1102 subtractions, 1264 extensions, 1175 retractions in 21 neurons.*

2.2 New techniques for structural morphometrics

Once initial validation of the new Dynamo python software on existing datasets was performed, progress could be made towards aim 2: streamlining structural morphometric analysis for new research. Half of this effort was focussed on data creation – that is, making it faster and simpler to collect and validate data by increasing interoperability with existing tools, automating some error detection, and providing a simple user interface for a researcher. The second half was aimed at providing more analysis techniques to utilize the rich data sets of developing arbors, registered across time. This section describes both feature sets, and gives an example of using them for a data set collected in collaboration with doctoral student Peter Hogg.

2.2.1 Galvanometer 2P microscope

For purely structural morphology analysis, the primary data source is clean volumetric scans taken at a frequency high enough to capture events of interest, such as filopodia additions, subtractions, and motility. There is a natural trade-off between signal-to-noise ratio of the image, and spatiotemporal resolution, as many techniques to increase the former (e.g. reducing shot noise by averaging multiple images, or increasing dwell time) come at the expense of either fewer voxels being recorded, or slower times between volumes. The health of the specimen is also of concern due to *in vivo* imaging, as any damage to the neuron may invalidate results. Bleach dynamics due to over-exposure is also a problem, especially for any experiment where metrics are calculated from fluorescence intensity. For the data in this section, looking at structural changes alone, we found that a galvanometer-based 2P microscope comprising an Olympus FluoView 300V confocal scan box, coupled to a Chameleon Ultra II Ti:Sapphire laser (Coherent, Santa Clara, CA) gave sufficient images.

We could achieve raster scans of full 3D volumes at a resolution of 512x512x100 voxels across a real-world space of 150x150x150 µm, captured every 5 minutes, which was fast enough to cover most filopodial dynamics (Hossain et al., 2012; Portera-Cailliau et al., 2003). At this rate, the raw images were considered clean enough for accurate arbor reconstruction at the level of filopodia (see Figure 6) without excessive bleaching or damaging the neuron or specimen. Non-neuronal fluorescence is present (e.g. stationary melanocytes as well as moving macrophages) but are easy to remove due to the voxel distance from the neuron. Noise is also present, however not to the level which disrupted automatic reconstruction (section 2.2.3).

2.2.2 Image preprocessing

Dynamo supports importing imaging volumes ('stacks') into the application as .tif or .lsm files, using the format CZXY – i.e. the first (optional) dimension representing detector, often one per color, the second representing Z depth, and the final two being X and Y for each imaged plane. Dynamo itself does minimal image processing of these volumes, only applying gamma correction by normalizing intensities to [0, 1.0] and raising to the power of 0.8. Other processing is possible, but left to the researcher to apply before importing, as different experiments have different requirements, and we wish to avoid Dynamo modifying raw data unexpectedly.

Noise will always exist in the image, and automated techniques do exist for denoising these two-photon volumes such as CANDLE (Coupé et al., 2012) and CARE (Weigert et al., 2018), but when running these on our image experimentally, we found that filopodia would be smoothed away (see supplemental Figure 19). This would invalidate lifetime or subtraction metrics, so smoothing was not applied. That said, for other users of Dynamo, smoothing (or any custom pre-processing step for the images) can always be done outside of Dynamo, before importing the volume.

Another common pre-processing step that simplifies registration of arbors across time is image stabilization, to correct any 3D translational drift. This is achieved by finding the whole voxel XYZ offset that minimizes distance between the two images, and applying this to the later time point. By using stabilization, any subsequent volume can achieve a good initial reconstruction by simply copying the arbor POI from the previous volume, and applying the register-with-adjust algorithm from section 2.1.3. For the same reasons detailed above, this preprocessing is not provided within the Dynamo application, but software to run this on a collection of .tif files is available in the Dynamo source repository (see Appendix A: Source repository and documentation).

2.2.3 SWC Import

With an efficient drawing interface, manual construction of an entire arbor (in the order of 1000 POI on 100 branches) is still a time intensive operation, taking multiple hours. Arbor reconstruction on subsequent imaging volumes is faster, thanks to the register-with-adjust algorithm (section 2.1.3), so implementation of new features began with trying to reduce the time to get the first reconstruction.

Thankfully, arbor reconstruction from volumes is an active area of research (see section 1.3), and many of these techniques are freely available to run on our volumes, so the approach was to utilize these instead of attempting to write our own. By using the Vaa3D software package (Peng et al., 2010), automated reconstructions can be produced within a few minutes, and exported to the open source .swc file format (Cannon et al., 1998). This format includes all the information needed for Dynamo to load the estimated tree structure: a list of POI, their 3D locations in XYZ voxel space, their approximate radius (also in voxels), and their parent POI. Code was added to Dynamo to enable import of an .swc file into an existing imaging volume,

and using that to initialize the tree. The one remaining task was to clean up data missing from the format, most noticeably primary/secondary branch information at bifurcation points. For POI with multiple children, Dynamo maintains which child is on the same (primary) branch as its parent, used for branch length and order calculations. By default, Dynamo will assume the first child of a POI is on the primary branch. If incorrect, the primary branch can be manually set at each bifurcation point using the drawing interface (Ctrl-B). An algorithm is provided to perform whole-tree primary branch estimation, by examining each bifurcation POI, calculating the length of all the child branches, and setting the primary branch to the longest.

Once the .swc is loaded into Dynamo and primary branches are set, the tree is displayed along with the imaging volume, with all POI marked in a distinct color: light blue, rather than the usual white. This is used to indicate to a researcher that the points were automatically assigned, and require manual inspection before results of morphometric quantifications should be trusted. As discussed earlier, initial reconstructions almost always require some manual correction, especially when research is focussed on fine details such as filopodial or synaptic protrusions, which automatic algorithms can estimate incorrectly, or miss completely. This color marking will be cleared on selection of each POI, with options available to mark and unmark a POI or its entire subtree, for convenience. Note that these POI marking options can be used outside of SWC import as well, whenever it is useful to track progress of manual tree inspection.

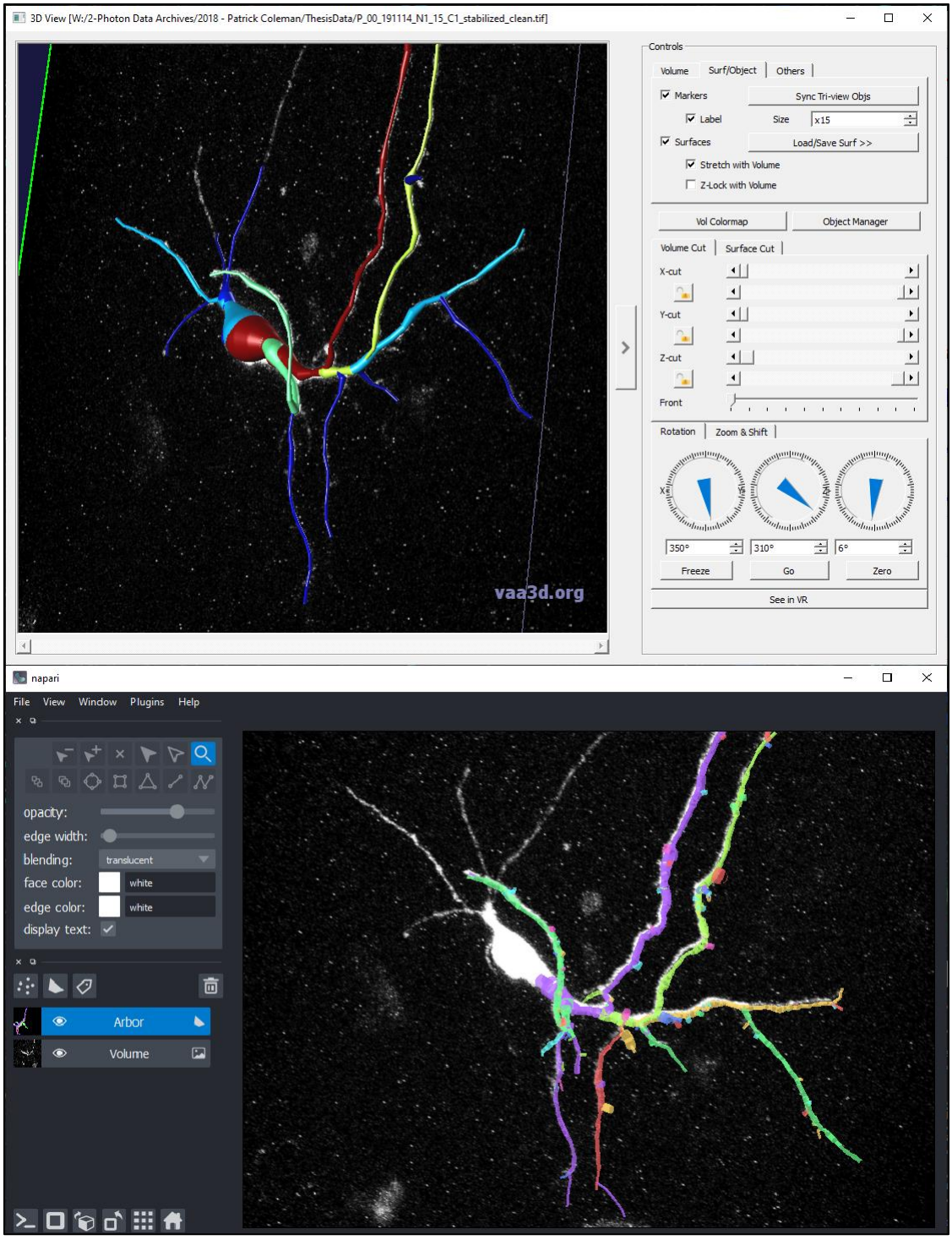

Figure 6: Integration with automated reconstruction algorithms, a) Automatic Reconstruction in Vaa3D, taking under 10 minutes but only capturing large branches. b) After import into Dynamo and manual correction to capture fine detail, the improved 3D arbor is shown in the software (using Napari).

By using the approach of automatic reconstruction, SWC import, primary branch estimation, and manual cleanup, the updated approach now takes on the order of 30-60 minutes for initial tree drawing. The majority of this is in POI cleanup (primarily: adding missed sections, fixing protrusions, and cleaning branch-dense locations) depends on the accuracy of the automated algorithm used, so the time taken is expected to reduce either with improved, cleaner imaging techniques, or as the reconstruction algorithms themselves are enhanced over time. For completeness, export from dynamo to the .swc format was also included, so Dynamo itself can be used for arbor reconstructions. The resulting trees can then be exported to other tools for analysis like NEURON (Carnevale & Hines, 2006) or archives of open data sources such as NeuroMorpho (Ascoli et al., 2007).

2.2.4 Radius and 3D Visualization overlay

A second improvement, not existing in the original MATLAB Dynamo but followed from .swc import, was utilizing knowledge of the radius at each POI. This is vital for internal propagation dynamics (Anwar et al., 2014), but also useful for minor corrections to protrusion length calculations, as these are usually performed from the edge of the parent branch, rather than its center (Feldman, 2009). Radius values are also useful for researchers analyzing spiny neurons, as spine neck and head sizes can be represented, although all neurons analyzed in this paper are non-spiny. A feature was added to Dynamo to draw the POI circles at the given radius, so they can be aligned with the visual neuron image (in the X/Y plane).

As is a theme in this research, automatically generated metrics (e.g. radius) almost always require manual adjustment before they can be trusted for findings, so Dynamo provides options for increasing/decreasing radius, as well as setting to a specific value by clicking on the outside of the POI in the drawing interface. A second option of automatic setting is provided for those

not using .swc – implemented by Peter Hogg, this sets the radius by applying a Roberts edge detection to the plane, followed by gaussian smoothing, then finally growing the POI's radius until the average intensity is lower than a threshold. As with other automatic options, this can be run against an entire arbor, then corrected manually in the few places it produces erroneous results.

One benefit of per-POI radius information is that, for the first time in Dynamo, 3D arbor reconstructions can now fully visually represent the arbor volume. Rather than branch segments being represented by cylinders of a fixed radius, they can have the end radii set to the correct sizes, using the stored parent and child radii. To fully appreciate this, Dynamo utilized a new performant scientific visualization toolkit (Napari Contributors, 2019). Napari is similar to a python-based ImageJ, offering several basic visualization tools (such as 3D volumes and shapes), and allowing programmatic access, which Dynamo can use these to construct visualisations. Dynamo combines Napari with the radius information to draw the complete, correctly sized arbor on top of the imaging volume in 3D, which can then be rotated/zoomed by an operator to try to find any remaining errors in the reconstruction (shown in Figure 6b).

2.2.5 Manual verification

Due to the excessive amounts of tedious, manual drawing, as well as less visible but still important structural details such as primary branch structures and across-time point registration, dynamic morphometrics research has many possible sources of error which could invalidate results from hundreds of hours of data collection and preparation. To assist with this, a few automated arbor validation checks have been included, with the ability to easily insert more as deemed useful. At the press of a button, these are run across the currently selected arbor, and any

potential sources of error printed, allowing a researcher to examine and potentially correct these manually before analysis is performed.

The first check catches a common error source, when a primary branch changes between two points in time. It is not expected that a POI will 'change' branch over time, so the ID of its branch should stay consistent during the existing of the POI. This check finds all POI, and if the ID of their branch ever changes, this is marked as a probable error.

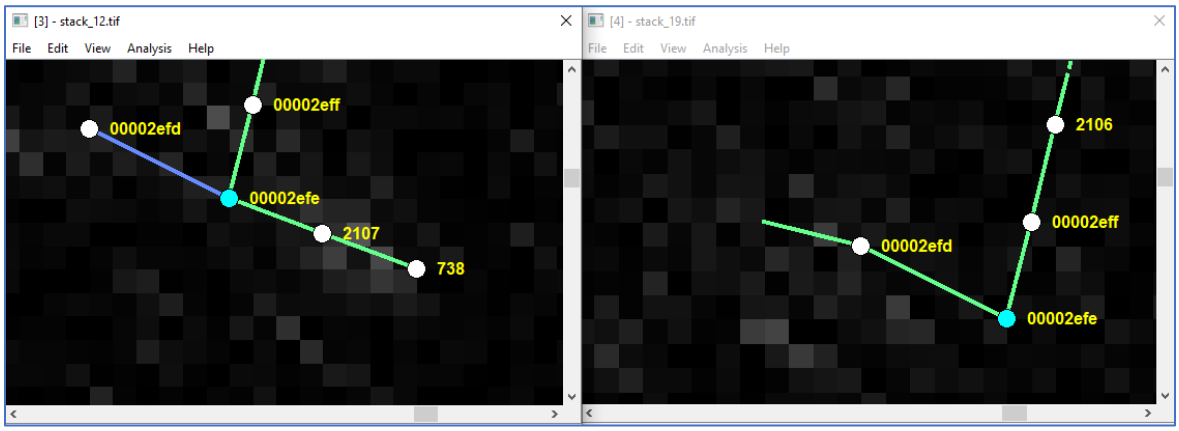

Figure 7: A discovered branch change – the terminal filopodia to POI 738 is subtracted, and 00002efd incorrectly becomes considered the primary branch. This is fixed by keeping 00002efd on its own branch.

The second check is for unusually tight angles that do not appear to be biologically feasible, and are more likely to be due to drawing error. By finding every triplet of grandparent-parent-child POI, and calculating the internal angle, we compare this to a threshold picked based on experimental data. This threshold is 126^{o} (70% of 180^{o}) for when all three are on the same branch, and 153^{o} (85% of 180^{o}) when the child is on a separate branch, as these are expected to be at a greater angle. As with above, any angle found to be above threshold results in a printed warning, which allows a user to expect and manually intervene if required.

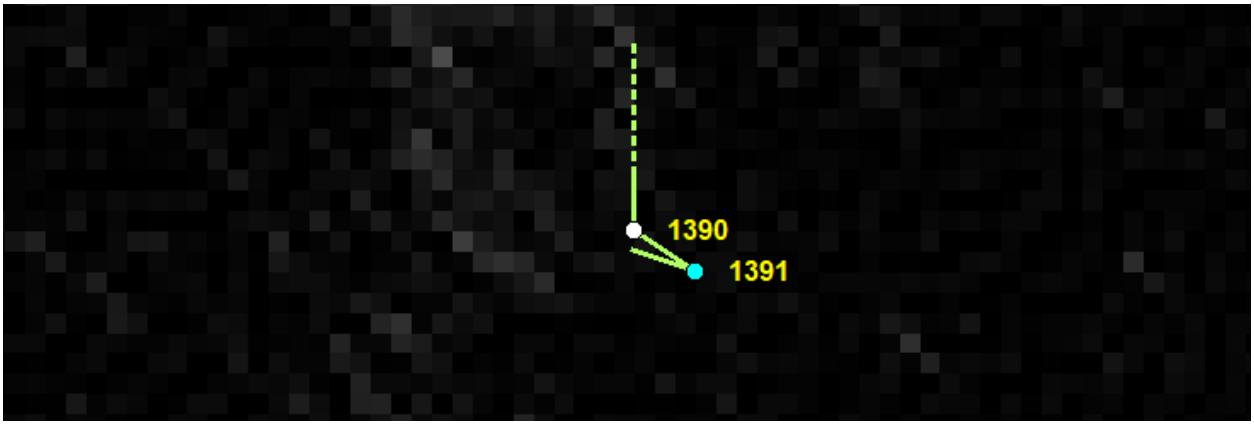

Figure 8: Problematic angle that will be flagged; 1391 is the true branch end, the following point was a mis-click and was deleted.

2.2.6 Simplified drawing improvements

A large focus on development of Dynamo was on usability for a researcher. Any study concerning dynamic morphometrics can result in hours of using Dynamo *per neuron*, and using an increased number of neurons should be incentivized to improve scientific validity. Because of this, small improvements can save a lot of time across the course of an experiment, and any minor inconvenience can quickly turn into a major pain point for a researcher. Multiple drawing improvements were added to the new version of Dynamo to help with this – the list is too long to be included exhaustively, but we cover some here.

The first most obvious example is that of supporting Undo (Ctrl+Z) – that is, after making an erroneous drawing, allowing a user to revert to the previous state. The earlier MATLAB version of Dynamo did not support this; instead, the arbor state was saved every three minutes, and in the event of major data entry error (e.g. accidental deletion of an entire branch), it could revert to the latest save, losing at most a few minutes of work. Autosave was kept in the

new Dynamo, although a recent history of all changes is also maintained within the application, and when a mistake is made, the user can instantly revert to the previous state.

A second feature added after discussion with users of the older Dynamo was the ability to add new POI mid-branch. The MATLAB version did not support this, which made it problematic to keep sparse POI along branches, in case filopodia or branches appeared at a place with no prior POI, so their base would not be positioned correctly. It had been common practice therefore to have extremely dense POI placement along branches, which increased drawing time. The new python Dynamo allows drawing a new POI between two existing POI on a branch. This allows manual drawing to have a much faster sparse initial placement, and be progressively refined to a higher density only to the level of precision required for the research question.

One final example of a feature added to the new python Dynamo that sped up initial arbor drawing and registration significantly is that of keyboard shortcuts. For example, selecting a POI by scrolling through Z planes until its ID is found, then and clicking on it, requires slow precise mouse movement. Instead, the key combination for Find (Ctrl-F) enables a user to select any identified POI they are looking for, and the view will automatically move to show the POI. Once selected, navigation to neighboring POI is also faster, by either walking up or down the branch (Ctrl-< and Ctrl->) or into a sub-branch (Ctrl-/), again avoiding the need for targeted mouse clicks.

2.2.7 Sholl analysis

Sholl analysis is a popular quantitative technique for measuring the complexity of neuronal arbors (Sholl, 1953). Traditionally performed in 2D, the analysis is performed by expanding a circle outward from the soma, and recording the number of branches crossing that circle at

specific distances. The resulting plots (of crossings vs distance) have characteristic shapes that capture branch counts and densities, and how they change based on distance from the soma. As an objective measure of dendritic field, it has been used to quantify neural developmental disruption in conditions such as the presence of nicotine (Powell et al., 2016) and deficiency of iron (Bastian et al., 2016).

As Dynamo contains the full reconstructed arbors, calculation of Sholl crossings for a given radius is can be automated by measuring the distance from the soma to each pair of joined POI in the arbor. This can be calculated for any desired range of radii, either in the original 2D (by discarding Z values) or in full 3D. Dynamo supports the calculation and export of these values for all time points through its analysis window, in addition to plotting the Sholl diagrams from within the application (Figure 9).

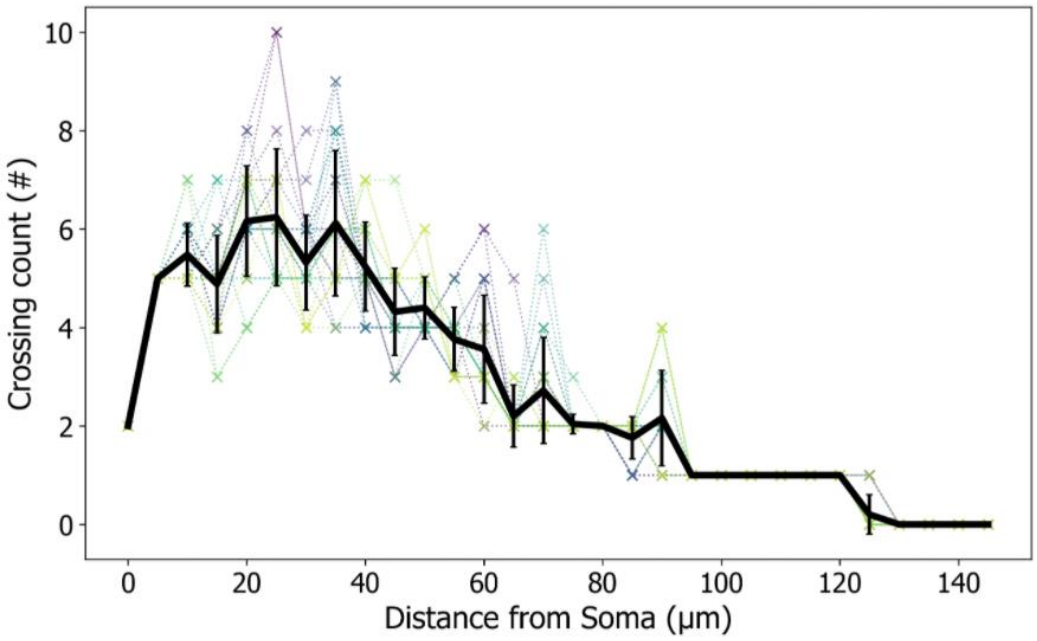

Figure 9: Sholl analysis within Dynamo, showing histograms of 3D branch crossings by soma distance. Colours indicate time points, with black showing mean ± std.

2.2.8 New modality: Puncta quantification

One extra modality of data that can be captured for a developing neuron, and useful to layer on top of contextual underlying structural morphologies, is that of localized puncta. That is, specific real-world 3D positions that identify the location and intensity of specific markers of interest. A common usage for this is in the quantification of synaptic markers such as PSD-95 (Niell et al., 2004), but can also be applied to other intracellular mechanisms of interest such as mitochondria (Brusco & Haas, 2015).

Dynamo supports these avenues of analysis by providing a simplified puncta mode, where identified circles can be drawn at their respective XYZ position within each volume, and moved/resized/deleted in volumes at subsequent time points. Analysis options are then presented, to count the total number of puncta, measure their sizes (as radii), and calculate the sum intensity in any imaging channel over the area enclosed by the puncta. For simplicity, puncta currently only support circular shapes as they reuse the same representation as arbor POI, containing XYZ position and radius, but these can be made more complicated in the future as required. An example of drawing puncta within Dynamo is given in Figure 10.

Puncta metrics

Size	Area of each puncta (μm^2)
Total Intensity	Sum of fluorescence within the puncta
Average Intensity	Average fluorescence within the puncta (Total Intensity / Size)
Nearest Point	POI on the arbor closest to centre of the puncta.

Table 4: Per-puncta analysis metrics available in Dynamo

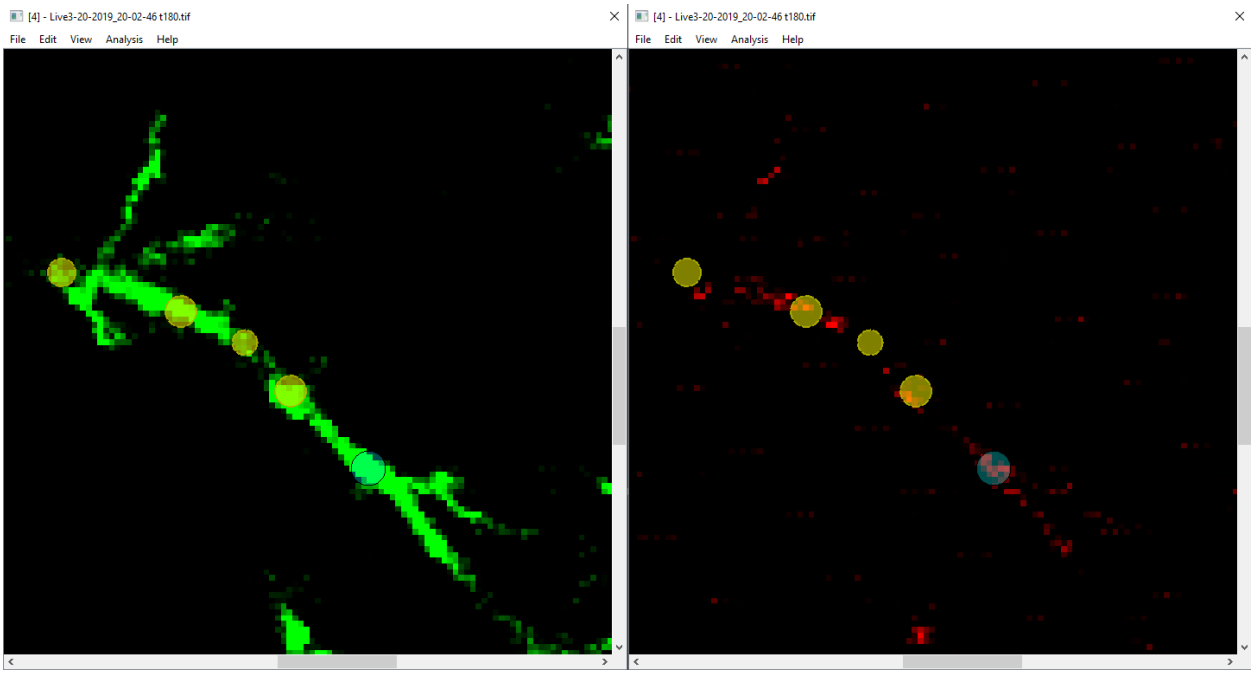

Figure 10: Example of puncta drawing, showing a) GFP space filler for arbor reconstruction, and b) PSD95 marker, with one puncta drawn on this plane (bottom right, blue) and four indicated on other nearby Z planes.

2.2.9 Application to data: The role of TRPC6 in development.

After implementation in Dynamo of the above techniques for structural dynamic morphometrics, the second aim of this research can only be achieved by applying these techniques to real-world neuron recordings. The data and analysis described here was done in conjunction with graduate student Peter Hogg, as part of his work looking at the role of the TRPC6 receptor on synaptotrophic growth in neurons. Dynamic morphometrics was used to characterize the growth of his neurons, in terms of branch sizes, filopodial densities, and addition/subtraction rates. Work is still ongoing, and the full study is planned to include multiple pharmacological conditions (control, plus a blocker and activator of TRPC6, as well as an upstream receptor mGluR1), with multiple neurons per condition and multiple time points per neuron, so this project guided much

of the process of optimizing the workflow within Dynamo. Here we present a subset of the data, describing the collection and analysis of filopodial dynamics in the presence of TRCP6 blocker GSK417651A and activator GSK1702934A.

2.2.9.1 Experimental protocol

Similar to section 2.1.7.1, albino *Xenopus laevis* tadpoles were raised, this time to stages 47-48 in order to capture earlier growth dynamics. Single neurons were labelled through injection of the dye Alex488-Dextran. Once filled, a galvanometer 2P microscope (section 2.2.1) was used to image twelve volumes at 5-minute intervals. Each imaging plane within the volumes is rastered twice then averaged, to reduce noise. This initial hour of recording forms an internal control for each neuron.

Following this, TRPC6 receptors were bathed in 0.5% DMSO Steinberg's solution, mixed with either 40nM of the blocker GSK417651A, or 440nM of the activator GSK1702934A, for the remainder of the experiment, to pharmacologically alter TRPC6 receptor activity. Twelve more volumes were then recorded at 5-minute intervals with continued drug application.

2.2.9.2 Results 1: TRPC6 inhibition decreases stability.

The volumes were converted into reconstructed arbor time series by first using Vaa3D to automatically approximate the initial reconstruction, exported to .swc and imported into Dynamo along with the first .tif, which was then manually inspected and cleaned within Dynamo. One at an acceptable level to the researcher, subsequent time point .tif volumes were imported, and Register-with-Adjust used to make small adjustments where the neuron structure had changed, then these were again manually inspected and corrected where necessary. It was estimated that

each volume resulted in under an hour of manual work, so the full 24 volumes per neuron could be done in a week, covering 2 hours of neuronal development at 5-minute intervals.

Once drawn, the neurons were inspected within Dynamo, but the final analysis was written in custom python, using Dynamo's export-metrics feature (section 0). By loading a .csv produced by Dynamo that included the length of all filopodia, Peter Hogg was able to visualize this as an intensity plot, with rows representing filopodia, columns representing points in time, and intensities on a scale to representing growth (green) to retraction (red). This type of visualization was generated for each neuron by changing which file Dynamo used to generate the analysis results. From these, more rows can be observed when TRPC6 is blocked, indicating higher filopodial turnover. Similarly, rows are shorter, with fewer stable (white) periods, showing a decrease in stability compared to control (Figure 11). These will be converted to single metrics and analyzed numerically in Dynamo once data collection is complete.

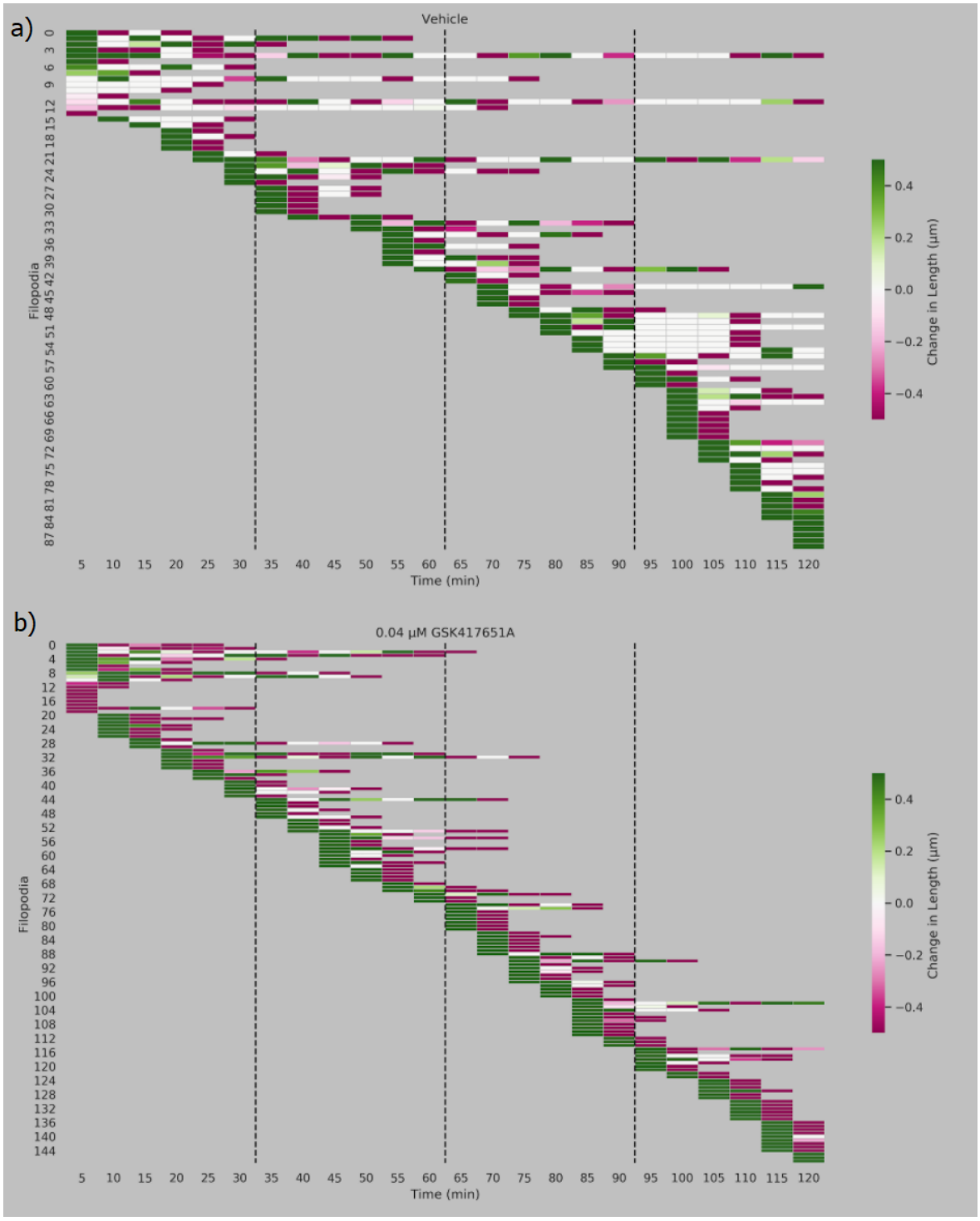

Figure 11: Waterfall plots of motility for a) Control and b) TRPC6 blocker GSK417651A. Rows represent filopodia, columns time, and color the change in length

2.2.9.3 Results 2: TRPC6 activation increases filopodial density.

One of the ways Dynamo was used to quantify the differences due to activation of TRPC6, was by calculating the filopodial densities at each time point. Dynamo's metrics can calculate both the total dendritic branch length, and counts of filopodia – both interstitial, terminal as well as total. By dividing the counts by TDBL, average filopodial density across the pre- and post-application volumes can be calculated for each neuron. Next, the change in interstitial filopodia density is normalized by dividing be the pre-application density. It was found that for the first three drawn neurons in the activator condition, there was a significant increase in filopodial density after activation by GSK1702934A. This increase was not observed in control neurons, which had 0.5% DMSO Steinberg's Solution applied rather than the activator (Figure 12). The datasets are still being expanded, and can be automatically processed in the same way, as with other experimental conditions such as the inhibitor used above.

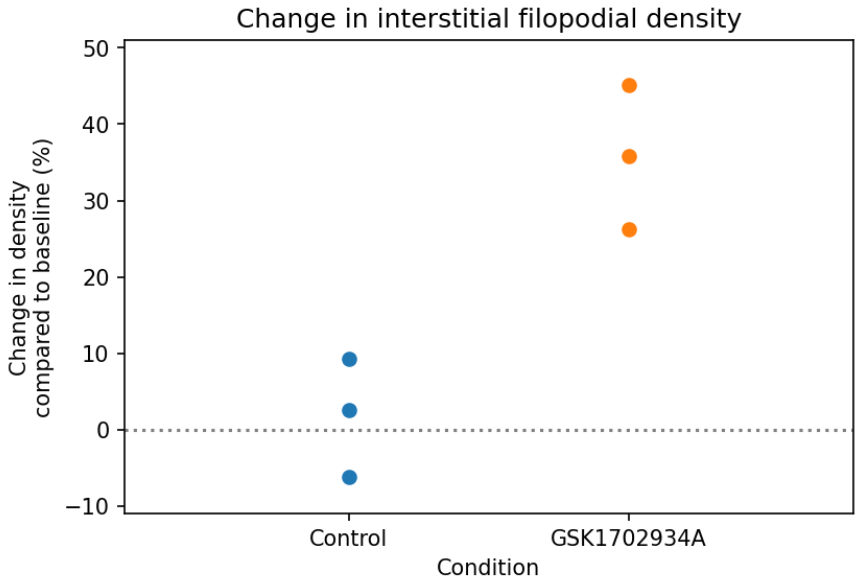

Figure 12: TRPC6 Activator GSK1702934A increases filopodial density

2.3 Functional analysis

2.3.1 AOD Microscope v2, with LabView software

The third and most recent iteration of *in vivo* imaging techniques used to produce the data analyzed by Dynamo was an acousto-optic deflector (AOD)-based random access microscope. By combining a piezo-actuator for focal plane (Z) positioning, with two OAD1121-XY acousto-optic deflectors for random access (XY) within the plane, this scope can record pre-drawn POI up to 200Hz in 2D, and up to around 6Hz for a full 3D arbor (Sakaki et al., 2020). Paired with a 2.5W Ti:Sapphire laser (Coherent Chameleon Vision II) tuned to 910nm, and two output PMTs (Hamamatsu H7422-40) with red and green emission filters, this was used to record two channels from 2P emission. Unlike the previous microscopes, this was implemented using LabView controlling software. LabView's drag-drop visual interface is very good for interfacing with hardware in time-sensitive applications, however analysis is much more complicated, so effort was made to export the raw data (arbor drawings, 3D structural volumes, and high-frequency recorded traces) into text formats for later processing in Dynamo's python. As a minor improvement to earlier techniques, the tadpole chamber also included a screen perpendicular to the eye of the tadpole, onto which a small projector (ShowWX+/PicoP, Microvision) displayed the stimulus. This supports a wider variety of stimuli, although for this experiment, full red is still used at different intensities.

2.3.2 POI registration of fixed trees across time

The first feature addition to the Dynamo software required for analyzing the data coming off this AOD 2P microscope was related to a change in the across-time point POI registration algorithm. The earlier 'Register-with-Adjust' algorithm (section 2.1.3) had permitted minor XYZ location

adjustments of the POI, moving them to the position that most closely matched their surroundings from an earlier time point. This is ideal when the registration and movement can happen before imaging, however this newer generation of AOD microscope enforced fixed locations, as these were used to record the functional calcium traces, so their location cannot be changed afterwards. The microscope also did not maintain a memory of POI identifiers between scans, so a second registration method was required: one that was not allowed to move POI in 3D, however could use their fixed locations and underlying arbor structure to best register POI between time points.

Due to the dynamic nature of the arbors that are growing between recordings, a few more complications had to be supported. Large dendrites can grow or shrink (or even totally appear or disappear at large time scales of imaging), so a changing number of POI per branch must be supported. Smaller branches (e.g. filopodia, synapses, or growth cones) are highly motile (Hossain et al., 2012; Portera-Cailliau et al., 2003), so the registration must also allow for a changing number of branches in the tree. Even for less motile mature neurons, translational drift and small rotations of the tadpole or imaging chamber are expected, so exact XYZ location matches cannot be used for registration, and the algorithm must be more forgiving of minor POI relocation. By default, the AOD simply identifies POI by their order in the tree, but this causes problems when the number of POI changes (see Figure 13).

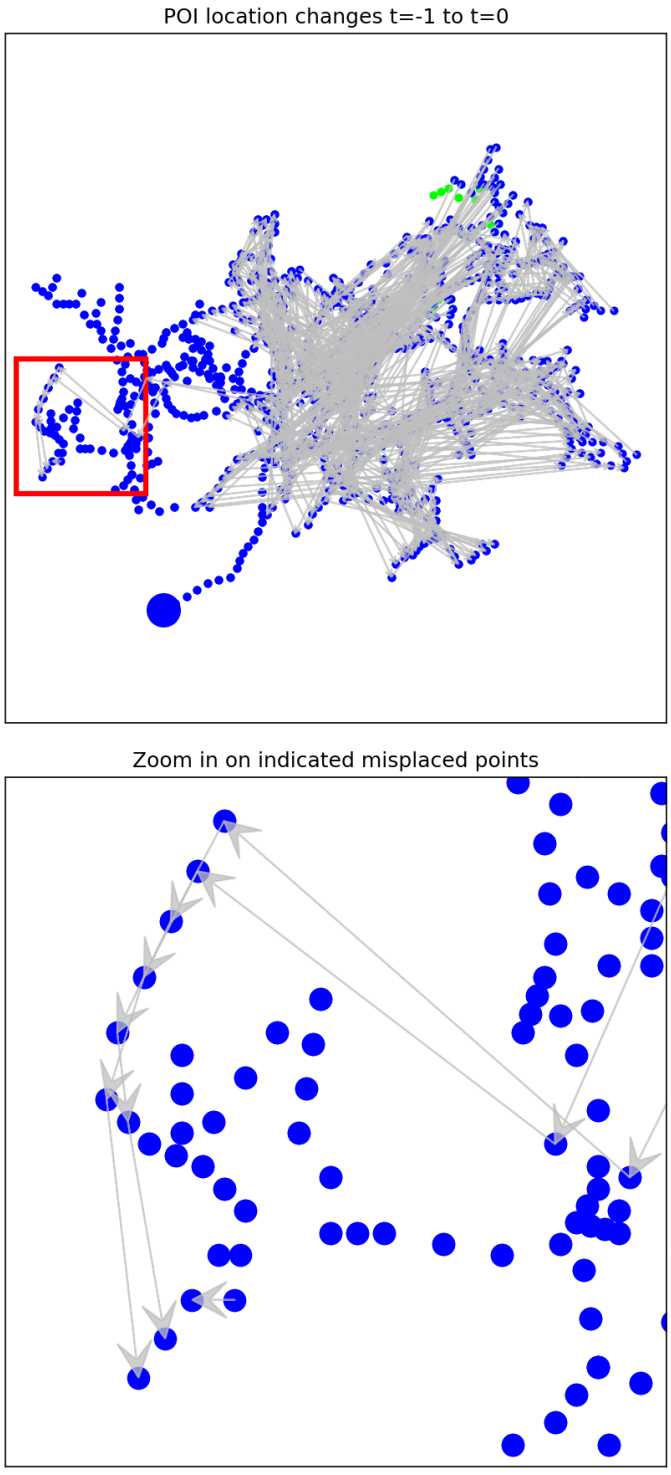

Figure 13: POI registration with microscope-assigned IDs that were not designed to be stable over time. Arrows indicate the location changes of a single identifier between consecutive time points, showing large jumps. POI lacking arrows are correctly registered.

Due to the nature of the tree structures, a recursive cost-based approach was taken, where ID alignment of POIs between two trees (or subtrees) can be given a ‘cost’, and the registration algorithm matches POIs in such a way to minimize that cost. The approach is similar to earlier work, minimizing a cost-based matching between axonal arbors (Chalmers et al., 2016), but now extended to support full trees.

The cost for whole-tree registration was a combination of two separate costs: *matched* and *unmatched*. To calculate the *matched* cost of a registration, the algorithm considers POI that exist on both time points, and adds up the Euclidean distances between their positions (relative to their respective parents, to allow for drift). The lower the matched cost, the closer the POI are to their original location. This can be thought of as the lengths of the arrows in Figure 13, so smaller is better.

The *unmatched* cost is much simpler: it is the total number of POI that remain unmatched by the registration in either time point, scaled by a constant ‘unmatched penalty’. This constant can be set by a user, and represents how far in their experimental setup a POI is allowed move before it should be considered ‘new’. For example, if the penalty is set to zero, only perfect XYZ location matches will be registered to each other, everything else will be set unmatched. A penalty too low will result in a registration that errs on the side of not matching POI, whereas a penalty too high will tend to match POI even if the locations have changed considerably. It is recommended any user trial a few values and pick the best - in Dynamo we found a good unmatched penalty of 10 (i.e. 10 voxels of shift allowed), so that is default in the software.

Finding the registration between time points that has minimal cost (i.e. a weighted mix of smallest relative distance change, and fewest unmatched POI) is initially a prohibitively slow process, as too many possible matchings must be checked. To speed it up, one innovation

introduced was the concept of a *maximum skipped* number - that is, the number of POI in a row in a registered tree that are able to be left as unmatched, before the entire subtree is considered unmatched. By setting this to a low value (3 is the default), this allows for a few POI to be added mid-branch, but means that completely new branches (or branches in the wrong direction) will be disregarded much faster.

Using the *maximum skipped* value, the final algorithm is achieved by defining a 'cost' function that takes two POI (one from each tree), as well as the number skipped previously in each, and returns the best registration between the subtrees rooted at the POI, as well as the best registration matching between them that achieves the cost. To calculate these, a few options are considered:

- Skip the current POI in the first time point's tree if we haven't skipped too many already. Find the lowest cost from matching one of its children to the POI in the second tree (and leaving the others unpaired). Or,
- The same as above, but this time by skipping the POI in the second tree, and finding the lowest cost of matching its children to the POI in the first. Or,
- Register the POI to each other. The cost of this is the matching cost for the POI (i.e. distance between their locations, relative to their parents), plus the cost of the best matchings between their children.

By allowing the large-scale branch skipping, and storing intermediate results to prevent recalculations, this algorithm now performs registration in under an hour for two arbors of the order of 1000 POI per arbor.

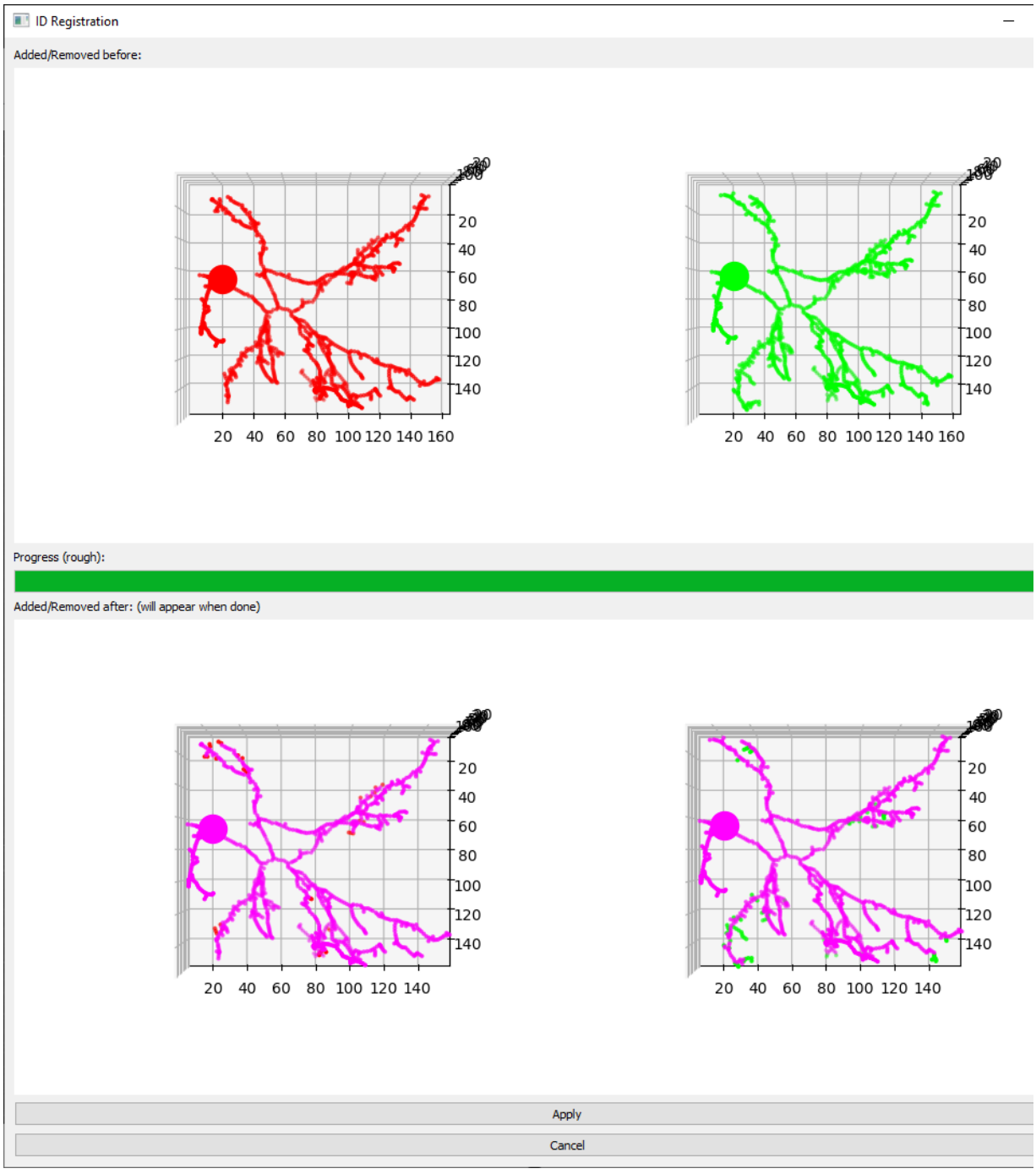

Figure 14: The results of within-Dynamo automatic registration. Top row: Previous registration, showing previous and current time point. Bottom row: Best registration. Red shows a removed POI, Green an added POI, and magenta a registered POI.

The final registration has been observed to generally be very accurate, usually with no or minor mistakes between arbors. As with all automated approaches, an option for manual registration is also provided in Dynamo, where an operator can select a point for each arbor, and instruct Dynamo to register them to the same ID. It is possible to use the manual registration technique for the entire registration needs, however a combination of automatic with manual corrections is recommended.

2.3.3 Loading traces into Dynamo

After fixed-tree registration aligns point IDs across time, the subsequent step allowing functional analysis in Dynamo is to load trace timeseries recorded for each POI at each time step. To support this, the Neurodata Without Borders format was selected (Teeters et al., 2015), as it is an open format supporting imaging time series (in addition to many other formats). By providing the user a way to import one .nwb file per epoch, Dynamo will associate any included time series to the POI in that epoch with corresponding identifier. Once loaded, traces for a selected POI can be displayed within Dynamo – multiple can be viewed simultaneously, across POI and across time points, either together or overlaid. A collection of filters has been implemented that can be applied to the traces, as follows:

2.3.3.1 $\Delta F/F_0$ filter

A core part of Calcium signal processing is converting raw fluorescent intensity readings to incremental ‘delta’ fluorescence, by subtracting a baseline F_0 then dividing by it to normalize whatever basal calcium is in the region recorded. As no pre-existing python implementation could be found, this has been implemented following the specification in (Jia et al., 2011).

2.3.3.2 Okada filter

A second filter migrated to python for cleaning up functional traces was the Okada filter, used to reduce shot noise on noisy activity data by flattening out samples that are abnormally higher or lower than the samples around them (Okada et al., 2016). Due to the low signal-to-noise ratio of the AOD scope data, this was applied to all raw traces before $\Delta F/F_0$ filtering is performed.

2.3.3.3 Smoothing filters

The final step of pre-processing neural activity indicators was to fit traces to the expected exponential-decay dynamics of ΔF/F0 signals. A python library was already available for the popular, fast OASIS deconvolution algorithm (Friedrich et al., 2017), and a second python implementation was written for our own non-negative deconvolution ('NND') approach used in prior research (Podgorski & Haas, 2013).

Figure 15 contains an example showing somatic traces at two consecutive time points visualized within Dynamo, with changing sets of filters applied.

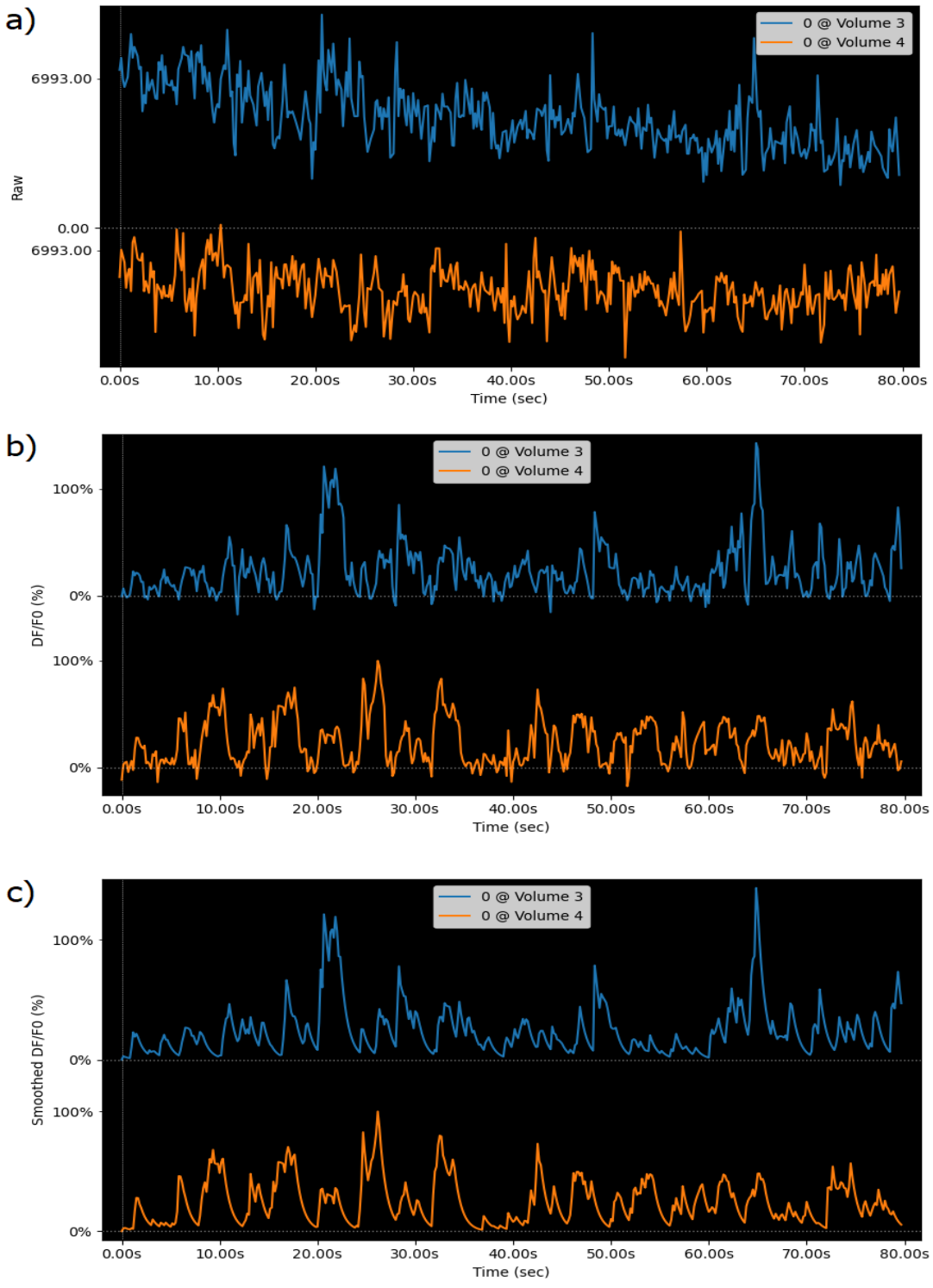

Figure 15: Somatic traces displayed in Dynamo for two time points, showing a) Raw intensity, b) DF/F0, and c) DF/F0 after smoothing through non-negative deconvolution using indicator dynamics

2.3.4 Application to data: Characterizing back-propagating action potentials

To demonstrate the utility of the new functional analysis tools, we first analyzed existing data sets to relate function to structure in a familiar context: that of back-propagating action potentials (bAPs). It is known that after somatic action potentials, bAPs can propagate backwards along the dendritic arbor, depolarizing it and affecting localized dendritic calcium levels (Ali & Kwan, 2019) that can be picked up by GECIs. In our own recordings we have observed large global increases in calcium $\Delta F/F_0$ during somatic events (Figure 16) which can complicate identification of synaptic events.

These bAPs are believed to be important for synaptic plasticity (Sjöström & Häusser, 2006), so their dynamics are of interest in developing cells. In this section, we describe analysis techniques that can be used to model bAP strength as an exponentially decreasing function over distance from the soma, quantify the decay rate, and use Dynamo to calculate the progress of this metric over time for a neuron by combining structural and functional recordings.

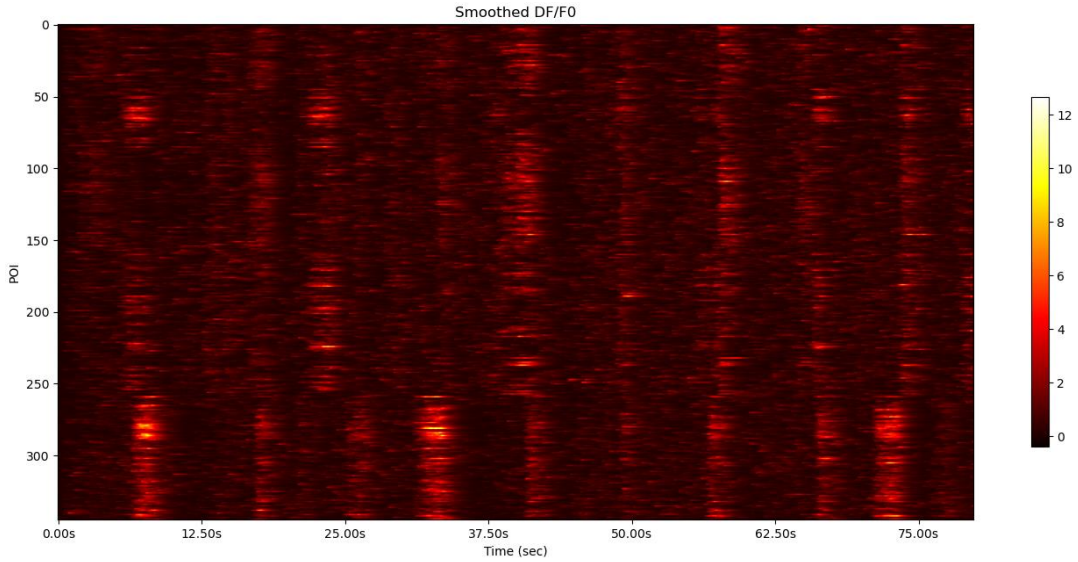

Figure 16: Intensity plot within Dynamo showing DF/F0 across time for all POI. All filters (Okada, DF/F0, NND) were enabled for this plot.

2.3.4.1 Experimental protocol

Similar to the initial experiment in section 2.1.7, albino *Xenopus laevis* tadpoles were used, this time performing single-cell electroporation of a combination plasmid mCyRFP-P2A-jGCaMP7s-farn, to express both an updated farnesylated green calcium indicator alongside a red indicator for cleaner structural arbor marking. Once placed inside the AOD microscope from section 2.3.1, a three-dimensional volume scan is taken every 30 minutes, covering an area of 112x112x105 µm, or 512x512x70 voxels. Using this, an operator manually draws the arbor structure using the LabView interface, and these locations are used to perform the functional random-access scans, capturing the GCaMP7s activity at 6Hz during a series of four OFF stimuli – where the projector shows a solid red image, which is turned off for 50ms. These are separated by a pseudo-random interval of 8 to 12 seconds, to reduce any learned timings (Sumbre et al., 2008). This pattern – of 3D structural volume followed by per-POI time series recordings – is repeated for four total epochs. As before, an LTP-inducing 'spaced training' stimulation paradigm was given between the second and third scanning epoch.

2.3.4.2 Results

Recorded traces for each epoch are loaded into Dynamo, cleaned using the Okada filter, converted into $\Delta F/F_0$, and finally smoothed to responses using non-negative deconvolution. By inspection of the somatic traces, stimuli that invoked action potentials are selected. Each POI has its per-stimulus response calculated as the maximal $\Delta F/F_0$ from 0 to 2 seconds after the stimulus, and finally the bAP strength is defined by the mean per-stimulus response across all stimuli that invoked action potentials. Finally, by comparing each POI's bAP strength against its intracellular distance from the soma, we characterize the bAP decay as it propagates backwards, by modelling it as exponential decay and calculating the distance constant (Perez-Alvarez et al., 2020).

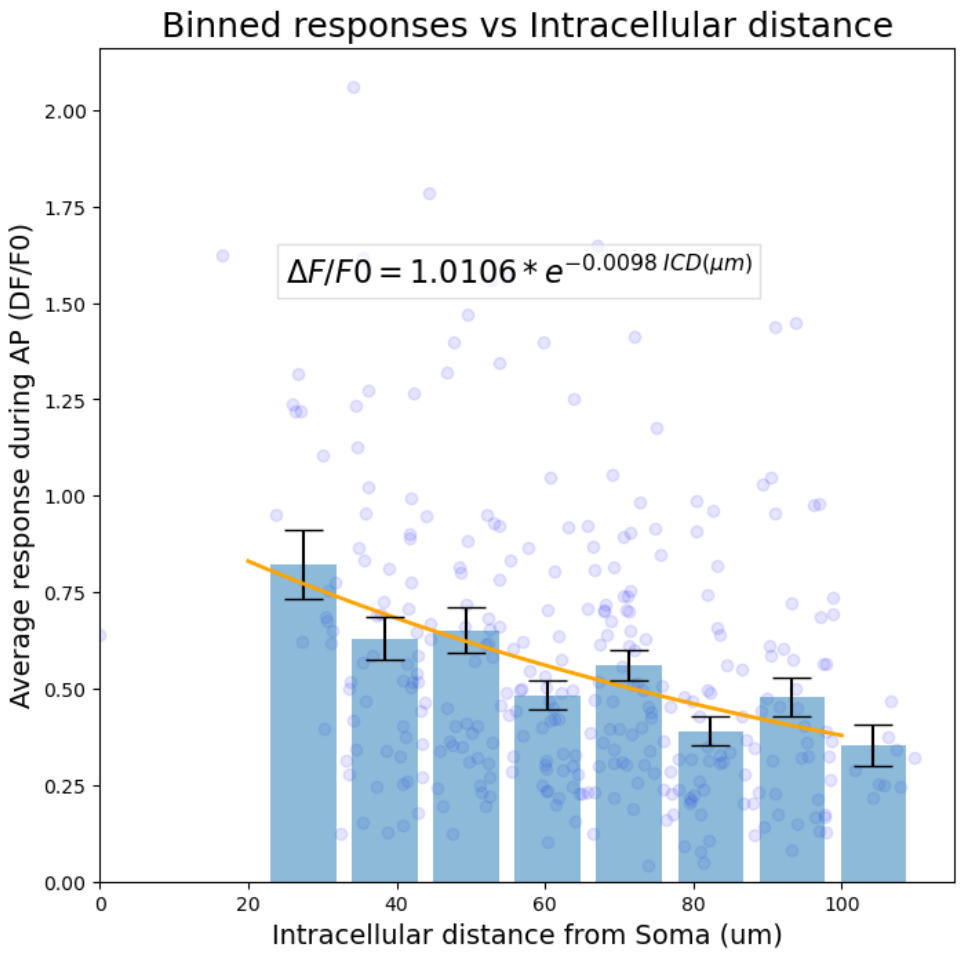

Figure 17: DF/F0 response vs intracellular distance. Fitted exponential decay added, showing decay constant (-0.0098).

By calculating the decay constant across recordings, we can observe how it changes during development. No significant changes were found across the 2hr of recordings.

2.3.5 Application to data: Functional-Structural clustering.

For the final project of this thesis, we expand our structural-functional analysis to use the changes in morphology data handled by Dynamo, and combine the structural filopodia additions over time to proxies for functional activity at filopodia. It has been shown that in mature cells, added synapses do cluster (Kastellakis et al., 2015) in particular during learning (Frank et al., 2018), and synaptic responses are also clustered (Kleindienst et al., 2011), but the interplay

between structural and functional clustering of filopodia in a developing neuron is unclear. This section describes an experiment to address this issue, and how Dynamo is used to quantify both patterns filopodia responses around clustered additions, as well as the structural and behavioural patterns of the added filopodia.

2.3.5.1 Experimental protocol

As above, we again electroporate a GCaMP7s-P2A-mCyRFP plasmid, and screen for expression in type 13b pyramidal neurons in the dorsolateral tectum of stage 50 albino *Xenopus laevis* tadpoles. One protocol change made was to extend the stimulation performed during the functional recordings – starting with the four OFF stimuli, then transitioning from a solid red to solid black baseline via a linear darkening over 8 seconds. This is followed by four ON stimuli, where the projector shows red stimulation is shown for 50ms, again separated by pseudorandom intervals of 8 to 12 seconds.

2.3.5.2 Identifying responsive interstitial filopodia

The outlined protocol results in four epochs of functional scans of calcium activity per POI scanned: 30 minutes pre-training, as well as immediately, 30 minutes and 60 minutes post training. At each of these epochs, filopodia are identified by including branches of length under 10µm (Li et al., 2011). Terminal filopodia were identified as those which originate within 10µm of the end of their parent branch; these were excluded from analysis, as their developmental behaviour is known to be more distinct from traditional connective synapses (Hossain et al., 2012). The remaining, interstitial filopodia are then classified as responsive or unresponsive by examining the calcium responses at both their base origination point, as well as their tip. First, the raw intensity for both tip and base POI are converted to $\Delta F/F_0$ to better capture transient

increases due to activity. Next, noise levels for each are calculated by measuring the median standard deviation at times when no stimuli are presented. Finally, a filopodia is classed as OFF-responsive if, after an OFF stimulus was presented, the $\Delta F/F_0$ at the tip was greater than the base by at least the noise level of that trace (the standard deviation of the $\Delta F/F_0$ for 3 seconds pre-stimulus). This was done to avoid false-positives from back-propagating action potentials (bAP), where the filopodium is inactive but calcium levels increase across the arbor, decreasing in intensity as it travels distally. More details are available in the methods section of (Podgorski et al., 2021).

2.3.5.3 Results: Post-training additions: clustered, more stable, and responsive

Across the experiment, we now have locations of interstitial filopodia, as well as a classification of those that are responsive, as well as those that first were added to the arbor during the experiment. By using the Monte Carlo clustering technique (section 0), those interstitial filopodia added during the spaced-training are observed to be clustered to responsive filopodia - that is, their clustering value is significantly higher ($p<0.05$ using t-test for n=14 neurons)**.** As a control, performing the same analysis with interstitial filopodia added before the LTP-inducing training, no clustering is observed (Figure 18a)

Limiting ourselves to only the interstitial filopodia added after training, we next classify them as 'clustered' if they grew within 10um of a responsive filopodia (and the rest 'unclustered'). Those that are clustered have a significantly higher chance of surviving an hour compared to unclustered ($p < 0.01$, χ^2 test of n=118 additions on the 14 neurons), and those that survive are more likely to be classed as responsive ($p < 0.05$, χ^2 test of n=95 surviving additions) (Figure 18 b & c). This supports earlier findings from synapses, that added filopodia and active filopodia

cluster, and suggests that this clustering is centralized around existing seed filopodia that were already responsive to the entrained stimulus.

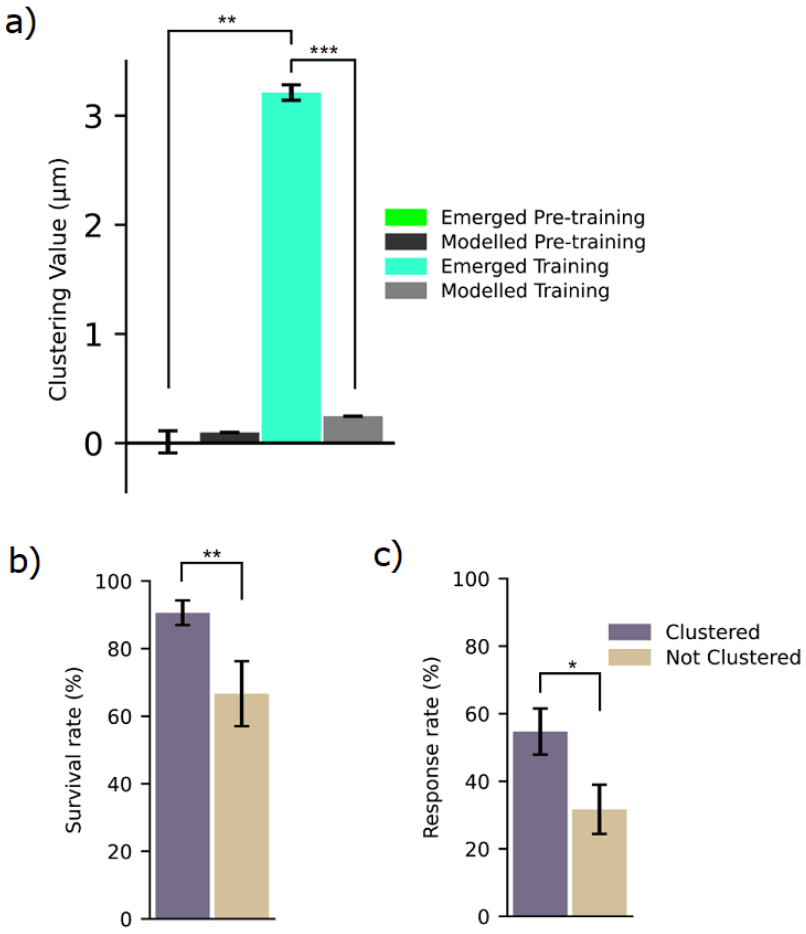

Figure 18: Structural-functional dynamic morphometric results. a) Filopodia added after LTP-inducing training cluster to existing responsive filopodia, and those clustered are more likely to b) survive and c) become responsive.

Chapter 3: Conclusion

3.1 Summary of Aims

Revisiting the aims in section 1.8, this thesis has achieved them through Dynamo; the open-source python library successfully runs across all three major operating systems (Windows, Mac and Linux), and has been applied to multiple research projects. Tooling and analysis have been provided for making discoveries on the dynamic structural morphometrics of developing neurons, as well as the interaction between these and their functional dynamics. Section 2.1 covers the initial migration, and validates the approach by loading and analysing existing MATLAB datasets. Section 2.2 then expands on the options available for structural morphology reconstruction and registration, giving examples from a study which utilized information from hundreds of reconstructions (8 neurons x 24 time points per neuron). By combining automated and manual methods, precise morphologies could be produced in a time-efficient manner, and an example is given that uses dynamic morphometrics of the reconstructions to show that TRPC6 activation increases filopodial density. Finally, section 2.3 presented analysis tools for functional, activity-based metrics, and could use these in conjunction with full-arbor comprehensive two-photon scanning to provide novel findings linking the structural and functional environments during development.

3.2 Strength and limitations

One primary strength of Dynamo is its flexibility for multi-modal analysis of neurons. Many traditional techniques look at a single data source – for example, only functional calcium dynamics, or only 2D structural arbors at a single time-point, whereas the formats supported by

Dynamo allow these to remain, but also be extended to 3D, multi-color images, at both millisecond and minute/hour time scales. The open nature of the code allows simple modification as new modalities are required, and this flexibility has been proven by its use in multiple studies by multiple researchers. This extensibility does come at the cost of making it more difficult to decide what features go in Dynamo, and which remain up to the researcher to develop on their own. Inclusion in the software by default makes it a more appealing and simpler package to use, but comes at the cost of bundling in a large amount of options that may never be used (the approach used by ImageJ (Collins, 2007)). Conversely, removing uncommon analysis from Dynamo will make maintenance and future changes easier, minimize the learning curve of usage when starting, but does put more burden on researchers to implement code for analysis specific to their project. Currently, the only features added are either ones required for dynamic morphometrics (drawing and registration tools, motility analysis) or very simple (3D viewer, TDBL and branch counts), though this may be updated as more researchers use the tool.

A second strength that guided much development was the combination of automated techniques (e.g. reconstruction and registration) with manual intervention. Throughout the research, it was found experimentally that a purely manual approach was too slow, whereas a purely automated approach produced results that were too erroneous to use for scientific findings. The proposed software was designed to reach a middle-ground, of efficient automatic processing followed by minor manual interventions to correct errors. As recording technology improves, and even more raw data can be recorded from a developing neuron, this combination will be even more necessary, as automated analysis is unlikely to reach 100% accuracy.

In contrast, a clear limitation of Dynamo is the POI and puncta representation, characterizing them just by XYZ location and radius. This simplified approach was chosen as it aligns with the existing SWC format, but it is clear that both POI and puncta cannot be perfectly modelled as circles. Improvements could be made to enable slightly more accurate approximations – e.g. a separate x and y radius, plus orientation, would permit angled ellipses – but both POI and puncta may have arbitrarily complex segmentations. For this, full volumetric masks would be required. Adding these to Dynamo was considered, but omitted, as the benefits gained were not worth the complexity of calculating the initial segmentation, as well as storing each mask, and analysis changes to calculations of length, intersections etc. If future experiments do require complete non-circular POI shapes, these may be added in the future.

3.3 Future work

A primary goal of future development on Dynamo will always be to enrich the in-built analysis available. In a similar vein to L-Measure (Scorcioni et al., 2008) or NeuroM (The Blue Brain Project, 2020), having publicly accessible algorithms to produce morphometrics from common data formats will allow researchers using Dynamo to investigate even more properties of their imaged neurons, with minimal overhead. To assist in this, an understanding of useful metrics to include will be best gained by using Dynamo for more studies, and by more researchers. Only through exposure to a varied assortment of research questions and experiments can we gain a better understanding of what features should be put in Dynamo itself, and what is better left to each individual research project.

A second simple task that would benefit both Dynamo and dynamic morphometrics research in general would be to codify a file format that supports this time series of developing

morphologies. SWCX started on this path, but did not support the full the full richness required by Dynamo, such as dynamic XYZ locations and radii for POI, or the richer puncta or trace information (Nanda et al., 2018). One possibility would be to publish a new, dynamic-SWC format, based off the information currently used in Dynamo. Instead, the recommendation would be to work all that data into the existing Neurodata without Borders format (Teeters et al., 2015). NWB is commonly used for both structural imaging as well as functional recordings and stimulation, and is already utilized within Dynamo, however it is lacking any support for the tree structures required for arbor reconstruction. Adding these would allow Dynamo to reuse this documented, open format, but also assist the numerous other researchers already using NWB files in combination with separate SWC.

Finally, whenever prioritizing work spent improving Dynamo, the primary question to ask was always: what will make future research easier and faster to do. For example, SWC import (allowing automated reconstruction) meant initial reconstructions could be done in minutes rather than hours. Automatic registration similarly reduced the time spent on each subsequent arbor, and keyboard shortcuts roughly halved the remaining time needed to manually verify the automated suggestions. To further reduce the time from data collection to results creation, the next area for improvements would be to improve the automated steps, so verification time is reduced. Dynamo itself cannot help much in this area, but instead improvements can be made from a few approaches. The first is by reducing imaging noise, as we found high levels of imaging noise to be a cause of many of the automated reconstruction issues. Improvements to both microscopy hardware and fluorescent proteins will assist with this, but another avenue would be even more improvements to denoising algorithms that preserve fine details of synapses and filopodia. The second approach that could quickly improve automated

reconstruction would be to design algorithms whose accuracy can be increased by providing them with initial estimates. Dynamo (or time series reconstruction in general) has an advantage that, other than for the original time point, subsequent reconstruction can be performed with access to the previous time point's arbor, which will only have minor modifications when the imaging interval is small.

Once the automated steps of reconstruction and registration can be improved further, and manual intervention of under an hour per neuron can be reached, this increased throughput will allow many more studies in dynamic morphometrics to take place, as well as increased sample size validity in those already taking place. Supporting this, as well as extra modalities such as functional or puncta tracking, plus an open design that facilitates new modalities, the Dynamo software will help the field of dynamic morphometric research uncover many rules behind how neurons grow.

Bibliography

Acciai, L., Soda, P., & Iannello, G. (2016). Automated Neuron Tracing Methods: An Updated Account. *Neuroinformatics*, *14*(4), 353–367. https://doi.org/10.1007/s12021-016-9310-0

Ali, F., & Kwan, A. C. (2019). Interpreting in vivo calcium signals from neuronal cell bodies, axons, and dendrites: a review. *Neurophotonics*, *7*(01), 1. https://doi.org/10.1117/1.nph.7.1.011402

Anwar, H., Roome, C. J., Nedelescu, H., Chen, W., Kuhn, B., & de Schutter, E. (2014). Dendritic diameters affect the spatial variability of intracellular calcium dynamics in computer models. *Frontiers in Cellular Neuroscience*, *8*(JULY), 1–14. https://doi.org/10.3389/fncel.2014.00168

Ascoli, G. A., Donohue, D. E., & Halavi, M. (2007). NeuroMorpho.Org: A central resource for neuronal morphologies. *Journal of Neuroscience*, *27*(35), 9247–9251. https://doi.org/10.1523/JNEUROSCI.2055-07.2007

Azari, H., & Reynolds, B. A. (2016). In vitro models for neurogenesis. *Cold Spring Harbor Perspectives in Biology*, *8*(6), 1–11. https://doi.org/10.1101/cshperspect.a021279

Bastian, T. W., von Hohenberg, W. C., Mickelson, D. J., Lanier, L. M., & Georgieff, M. K. (2016). Iron Deficiency Impairs Developing Hippocampal Neuron Gene Expression, Energy Metabolism, and Dendrite Complexity. *Developmental Neuroscience*, *38*(4), 264–276. https://doi.org/10.1159/000448514

Borges, S., & Berry, M. (1978). The effects of dark rearing on the development of the visual cortex of the rat. *The Journal of Comparative Neurology*, *180*(2), 277–300. https://doi.org/10.1002/cne.901800207

Brusco, J., & Haas, K. (2015). Interactions between mitochondria and the transcription factor myocyte enhancer factor 2 (MEF2) regulate neuronal structural and functional plasticity and metaplasticity. *Journal of Physiology*, *593*(16), 3471–3481. https://doi.org/10.1113/jphysiol.2014.282459

Cannon, R. C., Turner, D. A., Pyapali, G. K., & Wheal, H. v. (1998). An on-line archive of reconstructed hippocampal neurons. *Journal of Neuroscience Methods*, *84*(1–2), 49–54. https://doi.org/10.1016/S0165-0270(98)00091-0

Carnevale, N. T., & Hines, M. L. (2006). *The NEURON book*. Cambridge University Press.

Chalmers, K., Kita, E. M., Scott, E. K., & Goodhill, G. J. (2016). *Quantitative Analysis of Axonal Branch Dynamics in the Developing Nervous System*. 1–25. https://doi.org/10.1371/journal.pcbi.1004813

Chiu, S. L., & Cline, H. T. (2010). Insulin receptor signaling in the development of neuronal structure and function. *Neural Development*, *5*(1), 1–18. https://doi.org/10.1186/1749-8104-5-7

Collins, T. J. (2007). ImageJ for microscopy. *Biotechniques*, *43*(S1), S25--S30.

Coupé, P., Munz, M., Manjón, J. v., Ruthazer, E. S., & Louis Collins, D. (2012). A CANDLE for a deeper in vivo insight. *Medical Image Analysis*, *16*(4), 849–864. https://doi.org/10.1016/j.media.2012.01.002

Deppmann, C. D., Mihalas, S., Sharma, N., Lonze, B. E., Niebur, E., & Ginty, D. D. (2008). A model for neuronal competition during development. *Science*, *320*(5874), 369–373. https://doi.org/10.1126/science.1152677

Donohue, D. E., & Ascoli, G. A. (2011). Automated reconstruction of neuronal morphology: An overview. *Brain Research Reviews*, *67*(1–2), 94–102. https://doi.org/10.1016/j.brainresrev.2010.11.003

Dunfield, D., & Haas, K. (2010). In vivo single-cell excitability probing of neuronal ensembles in the intact and awake developing Xenopus brain. *Nature Protocols*, *5*(5), 849–856. https://doi.org/10.1038/nprot.2010.10

Feldman, D. E. (2009). *Developmental Synaptic Plasticity: LTP, LTD, and Synapse Formation and Elimination*.

Frank, A. C., Huang, S., Zhou, M., Gdalyahu, A., Kastellakis, G., Silva, T. K., Lu, E., Wen, X., Poirazi, P., Trachtenberg, J. T., & Silva, A. J. (2018). Hotspots of dendritic spine turnover facilitate clustered spine addition and learning and memory. *Nature Communications*, *9*(1), 1–11. https://doi.org/10.1038/s41467-017-02751-2

Friedrich, J., Zhou, P., & Paninski, L. (2017). *Fast online deconvolution of calcium imaging data*. 1–26.

Haas, K., Sin, W.-C., Javaherian, A., Li, Z., & Cline, H. T. (2001). Single-Cell Electroporation Neurotechnique for Gene Transfer In Vivo. *Neuron*, *29*, 583–591.

Haslehurst, P., Yang, Z., Dholakia, K., & Emptage, N. (2018). Fast volume-scanning light sheet microscopy reveals transient neuronal events. *Biomedical Optics Express*, *9*(5), 2154. https://doi.org/10.1364/boe.9.002154

Hedgecock, E. M., Culotti, J. G., Thomson, J. N., & Perkins, L. A. (1985). Axonal guidance mutants of Caenorhabditis elegans identified by filling sensory neurons with fluorescein dyes. *Developmental Biology*, *111*(1), 158–170. https://doi.org/10.1016/0012-1606(85)90443-9

Helmchen, F., & Denk, W. (2005). Deep tissue two-photon microscopy. *Nature Methods*, *2*(12), 932–940. https://doi.org/10.1038/nmeth818

Hossain, S., Sesath Hewapathirane, D., & Haas, K. (2012). Dynamic morphometrics reveals contributions of dendritic growth cones and filopodia to dendritogenesis in the intact and awake embryonic brain. *Developmental Neurobiology*, *72*(4), 615–627. https://doi.org/10.1002/dneu.20959

Jäckel, D., Bakkum, D. J., Russell, T. L., Müller, J., Radivojevic, M., Frey, U., Franke, F., & Hierlemann, A. (2017). Combination of High-density Microelectrode Array and Patch Clamp Recordings to Enable Studies of Multisynaptic Integration. *Scientific Reports*, *7*(1), 1–17. https://doi.org/10.1038/s41598-017-00981-4

Jia, H., Rochefort, N. L., Chen, X., & Konnerth, A. (2011). In vivo two-photon imaging of sensory-evoked dendritic calcium signals in cortical neurons. *Nature Protocols*, *6*(1), 28–35. https://doi.org/10.1038/nprot.2010.169

Johnston, M. v., Blue, M. E., & Naidu, S. (2005). Rett syndrome and neuronal development. *Journal of Child Neurology*, *20*(9), 759–763. https://doi.org/10.1177/08830738050200091101

Jouhanneau, J. S., & Poulet, J. F. A. (2019). Multiple Two-Photon Targeted Whole-Cell Patch-Clamp Recordings from Monosynaptically Connected Neurons in vivo. *Frontiers in Synaptic Neuroscience*, *11*(MAY), 1–13. https://doi.org/10.3389/fnsyn.2019.00015

Kastellakis, G., Cai, D. J., Mednick, S. C., Silva, A. J., & Poirazi, P. (2015). Synaptic clustering within dendrites: An emerging theory of memory formation. *Progress in Neurobiology*, *126*, 19–35. https://doi.org/10.1016/j.pneurobio.2014.12.002

Kazemipour, A., Novak, O., Flickinger, D., Marvin, J. S., Abdelfattah, A. S., King, J., Borden, P. M., Kim, J. J., Al-Abdullatif, S. H., Deal, P. E., Miller, E. W., Schreiter, E. R., Druckmann, S., Svoboda, K., Looger, L. L., & Podgorski, K. (2019). Kilohertz frame-rate two-photon tomography. *Nature Methods*, *16*(8), 778–786. https://doi.org/10.1038/s41592-019-0493-9

Kleindienst, T., Winnubst, J., Roth-Alpermann, C., Bonhoeffer, T., & Lohmann, C. (2011). Activity-dependent clustering of functional synaptic inputs on developing hippocampal dendrites. *Neuron*, *72*(6), 1012–1024. https://doi.org/10.1016/j.neuron.2011.10.015

Li, J., Erisir, A., & Cline, H. (2011). In Vivo Time-Lapse Imaging and Serial Section Electron Microscopy Reveal Developmental Synaptic Rearrangements. *Neuron*, *69*(2), 273–286. https://doi.org/10.1016/j.neuron.2010.12.022

London, M., & Häusser, M. (2005). Dendritic computation. *Annual Review of Neuroscience*, *28*, 503–532. https://doi.org/10.1146/annurev.neuro.28.061604.135703

Muller, E., Bednar, J. A., Diesmann, M., Gewaltig, M., Hines, M., & Davison, A. P. (2015). *Python in neuroscience*. *9*(April), 14–17. https://doi.org/10.3389/fninf.2015.00011

Nakai, J., Ohkura, M., & Imoto, K. (2001). A high signal-to-noise ca2+ probe composed of a single green fluorescent protein. *Nature Biotechnology*, *19*(2), 137–141. https://doi.org/10.1038/84397

Nakazawa, S., Mizuno, H., & Iwasato, T. (2018). Differential dynamics of cortical neuron dendritic trees revealed by long-term in vivo imaging in neonates. *Nature Communications*, *9*(1). https://doi.org/10.1038/s41467-018-05563-0

Nanda, S., Chen, H., Das, R., Bhattacharjee, S., Cuntz, H., Torben-Nielsen, B., Peng, H., Cox, D. N., Schutter, E. de, & Ascoli, G. A. (2018). Design and implementation of multi-signal and time-varying neural reconstructions. *Scientific Data*, *5*, 1–10. https://doi.org/10.1038/sdata.2017.207

Napari Contributors. (2019). *Napari: a fast, interactive, multi-dimensional image viewer for python*. https://doi.org/10.5281/zenodo.3555620

Niell, C. M., Meyer, M. P., & Smith, S. J. (2004). In vivo imaging of synapse formation on a growing dendritic arbor. *Nature Neuroscience*, *7*(3), 254–260. https://doi.org/10.1038/nn1191

Okada, M., Ishikawa, T., & Ikegaya, Y. (2016). A computationally efficient filter for reducing shot noise in low S/N data. *PLoS ONE*, *11*(6), 1–18. https://doi.org/10.1371/journal.pone.0157595

Parenti, I., Rabaneda, L. G., Schoen, H., & Novarino, G. (2020). Neurodevelopmental Disorders: From Genetics to Functional Pathways. *Trends in Neurosciences*, *43*(8), 608–621. https://doi.org/10.1016/j.tins.2020.05.004

Peng, H., Ruan, Z., Long, F., Simpson, J. H., & Myers, E. W. (2010). V3D enables real-time 3D visualization and quantitative analysis of large-scale biological image data sets. *Nature Biotechnology*, *28*(4), 348–353. https://doi.org/10.1038/nbt.1612

Perez-Alvarez, A., Fearey, B. C., O'Toole, R. J., Yang, W., Arganda-Carreras, I., Lamothe-Molina, P. J., Moeyaert, B., Mohr, M. A., Panzera, L. C., Schulze, C., Schreiter, E. R., Wiegert, J. S., Gee, C. E., Hoppa, M. B., & Oertner, T. G. (2020). Freeze-frame imaging of synaptic activity using SynTagMA. *Nature Communications*, *11*(1). https://doi.org/10.1038/s41467-020-16315-4

Podgorski, K., & Haas, K. (2013). Fast non-negative temporal deconvolution for laser scanning microscopy. *Journal of Biophotonics*, *6*(2), 153–162. https://doi.org/10.1002/jbio.201100133

Podgorski, K., Opushnyev, S., Brusco, J., Kim, D. H., Haas, K., Sciences, P., & Haas, K. (2021). *Comprehensive imaging of synaptic activity reveals dendritic growth rules that cluster inputs*. *5*(604), 1–60.

Portera-Cailliau, C., Pan, D. T., & Yuste, R. (2003). Activity-Regulated Dynamic Behavior of Early Dendritic Protrusions: Evidence for Different Types of Dendritic Filopodia. *The Journal of Neuroscience*, *23*(18), 7129–7142. https://doi.org/10.1523/JNEUROSCI.23-18-07129.2003

Powell, G. L., Gaddy, J., Xu, F., Fregosi, R. F., & Levine, R. B. (2016). Developmental nicotine exposure disrupts dendritic arborization patterns of hypoglossal motoneurons in the neonatal rat. *Developmental Neurobiology*, *76*(10), 1125–1137. https://doi.org/10.1002/dneu.22379

Ramon y Cajal, S. (1911). Histologie du système nerveux de l'homme et des vertébrés. *Maloine, Paris*, *2*, 153–173.

Raychaudhuri, S. (2008). Introduction to monte carlo simulation. *2008 Winter Simulation Conference*, 91–100.

Ristanović, D., Nedeljkov, V., Stefanović, B. D., Milošević, N. T., Grgurević, M., & Štulić, V. (2002). Fractal and nonfractal analysis of cell images: comparison and application to neuronal dendritic arborization. *Biological Cybernetics*, *87*(4), 278–288. https://doi.org/10.1007/s00422-002-0342-1

Sakaki, K. D. R., Podgorski, K., Dellazizzo Toth, T. A., Coleman, P., & Haas, K. (2020). Comprehensive Imaging of Sensory-Evoked Activity of Entire Neurons Within the Awake Developing Brain Using Ultrafast AOD-Based Random-Access Two-Photon Microscopy. *Frontiers in Neural Circuits*, *14*(June), 1–18. https://doi.org/10.3389/fncir.2020.00033

Scorcioni, R., Polavaram, S., & Ascoli, G. A. (2008). L-Measure: A web-accessible tool for the analysis, comparison and search of digital reconstructions of neuronal morphologies. *Nature Protocols*, *3*(5), 866–876. https://doi.org/10.1038/nprot.2008.51

Sholl, D. A. (1953). Dendritic organization in the neurons of the visual and motor cortices of the cat. *Journal of Anatomy*, *87*(4), 387–406. http://www.ncbi.nlm.nih.gov/pubmed/13117757

Sjöström, P. J., & Häusser, M. (2006). A Cooperative Switch Determines the Sign of Synaptic Plasticity in Distal Dendrites of Neocortical Pyramidal Neurons. *Neuron*, *51*(2), 227–238. https://doi.org/10.1016/j.neuron.2006.06.017

Stosiek, C., Garaschuk, O., Holthoff, K., & Konnerth, A. (2003). In vivo two-photon calcium imaging of neuronal networks. *Proceedings of the National Academy of Sciences of the United States of America*, *100*(12), 7319–7324. https://doi.org/10.1073/pnas.1232232100

Stuart, G., Spruston, N., & Häusser, M. (2012). Dendrites. *Dendrites*, 1–576. https://doi.org/10.1093/acprof:oso/9780198566564.001.0001

Sumbre, G., Muto, A., Baier, H., & Poo, M. (2008). Entrained rhythmic activities of neuronal ensembles as perceptual memory of time interval. *Nature*, *456*(7218), 102–106. https://doi.org/10.1038/nature07351

Teeters, J. L., Godfrey, K., Young, R., Dang, C., Friedsam, C., Wark, B., Asari, H., Peron, S., Li, N., Peyrache, A., Denisov, G., Siegle, J. H., Olsen, S. R., Martin, C., Chun, M., Tripathy, S., Blanche, T. J., Harris, K., Buzsáki, G., … Sommer, F. T. (2015). Neurodata Without Borders: Creating a Common Data Format for Neurophysiology. *Neuron*, *88*(4), 629–634. https://doi.org/10.1016/j.neuron.2015.10.025

The Blue Brain Project. (2020). NeuroM. In *GitHub repository*. GitHub. https://github.com/BlueBrain/NeuroM

van der Glaser, E. M., & Loos, H. v.d. (1965). A Semi-Automatic Computer-Microscope for the Analysis of Neuronal Morphology. *IEEE Transactions on Biomedical Engineering*, *BME-12*(1), 22–31. https://doi.org/10.1109/TBME.1965.4502337

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … Vázquez-Baeza, Y. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

Wang, Y., Li, Q., Liu, L., Zhou, Z., Ruan, Z., Kong, L., Li, Y., Wang, Y., Zhong, N., Chai, R., Luo, X., Guo, Y., Hawrylycz, M., Luo, Q., Gu, Z., Xie, W., Zeng, H., & Peng, H. (2019). TeraVR empowers precise reconstruction of complete 3-D neuronal morphology in the whole brain. *Nature Communications*, *10*(1), 1–9. https://doi.org/10.1038/s41467-019-11443-y

Weigert, M., Schmidt, U., Boothe, T., Müller, A., Dibrov, A., Jain, A., Wilhelm, B., Schmidt, D., Broaddus, C., Culley, S., Rocha-Martins, M., Segovia-Miranda, F., Norden, C., Henriques, R., Zerial, M., Solimena, M., Rink, J., Tomancak, P., Royer, L., … Myers, E. W. (2018). Content-aware image restoration: pushing the limits of fluorescence microscopy. *Nature Methods*, *15*(12), 1090–1097. https://doi.org/10.1038/s41592-018-0216-7

Zhang, J., Ackman, J. B., Dhande, O. S., & Crair, M. C. (2011). Visualization and Manipulation of Neural Activity in the Developing Vertebrate Nervous System. *Frontiers in Molecular Neuroscience*, *4*(November), 1–9. https://doi.org/10.3389/fnmol.2011.00043

Appendix

Appendix A: Source repository and documentation

The Dynamo software, as well as instructions for installation through the `pip` python packaging system can be found under the GitHub page of the Dynamic Brain Circuits in Health and Disease at UBC: https://github.com/ubcbraincircuits/pyDynamo

Documentation pages for usage are hosted in a similar location:

https://ubcbraincircuits.github.io/pyDynamo/

An archived version of Dynamo at the time of submission is available as the zip file '`Master's Thesis Supplemental Code.zip`', attached to this thesis.

Supplemental figures:

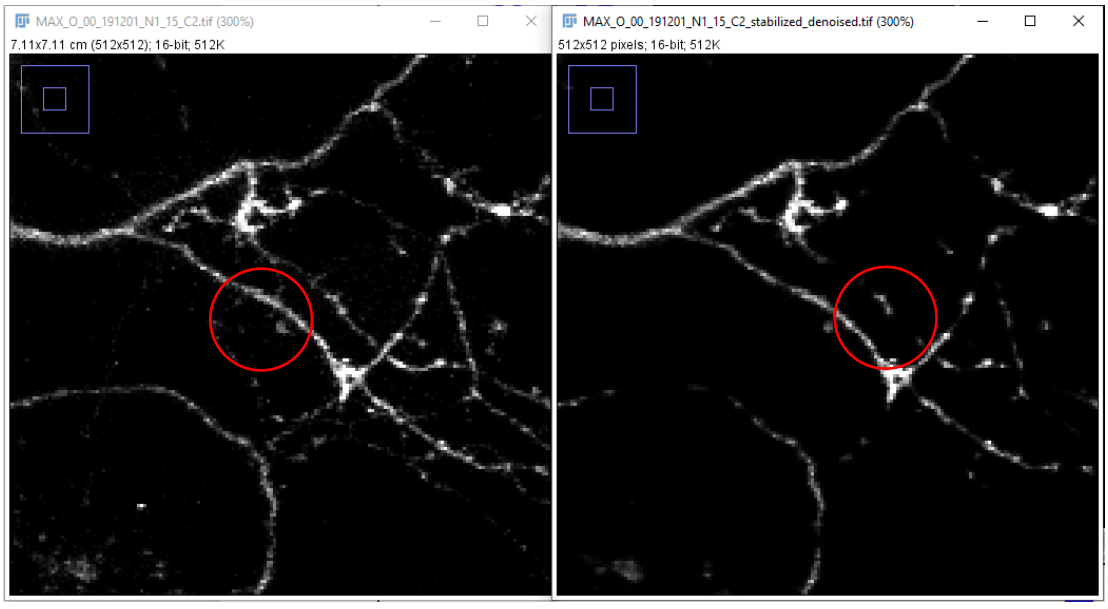

Figure 19 : An example of problematic denoising. Left: raw volume. Right: After CANDLE denoise. Both using maximal intensity Z projection. Missing filopodia indicated.