

QUANTITATIVE DESCRIPTION OF MICROTUBULE DISORGANISATION IN NEURODEGENERATIVE DISEASES: SOFTWARE DEVELOPMENT AND IMAGE ANALYSIS

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF BIOLOGY, MEDICINE AND HEALTH

2021

Beatriz Costa-Gomes

School of Biological Sciences

Contents

Abstract	9
Declaration	10
Copyright	11
Acknowledgements	12
1 Introduction	14
1.1 The nervous system, neurons and axons	14
1.2 The importance of microtubules for axonal physiology	15
1.2.1 Biochemical and physical properties of microtubules	15
1.2.2 Roles of axonal microtubules	16
1.2.3 Regulation of axonal microtubules	17
1.2.4 Key Problem and <i>Drosophila</i> as a solution	19
1.3 The model of local axon homoeostasis	20
1.3.1 Key Aims	22
1.4 Available strategies and software packages to analyse axonal MT phe- notypes	23
1.5 Aim	25
2 Languages, external resources and images used	27
2.1 MATLAB	27
2.1.1 MATLAB Incorporated Functions	27
2.1.2 Functions from Other Sources	27
2.1.3 Deep Learning Toolbox	28
2.2 Outsource Software	28
2.2.1 Ilastik	28
2.3 Biological Images	29
2.3.1 Manual Analysis	30
3 Image Processing	31
3.1 Background and Rationale	31
3.2 Shape Extraction from Images	35

3.2.1	Identifying a suitable filter	36
3.2.2	Obtaining binary shapes - Skeletonisation	40
3.3	Identify paths within the shape patterns	44
3.3.1	Existing Software and Methods	44
3.3.2	Implementation	44
4	Quantitative analysis and description of the binary shape	58
4.1	Background and Rationale	58
4.2	Length and Straight Segments	59
4.2.1	Length	59
4.2.2	Straight Segments	59
4.3	Curvature	68
4.3.1	Testing	73
5	ALFRED: the pipeline and the software	78
5.1	Rationale	78
5.2	Strategic decisions for software design	79
6	Biological Image Analysis	84
6.1	Description of the Images and Aim	84
6.2	ALFRED Analysis	84
7	Machine Learning	97
7.1	Background and Rationale	97
7.2	Methods	98
7.2.1	Data Clean-Up and Augmentation	98
7.2.2	Classification	101
7.3	Classification Results	105
8	Conclusions, discussion and future	107
8.1	Main Outcomes	107
8.1.1	Aim of the Thesis	107
8.1.2	Image Processing and Analysis	107
8.1.3	ALFRED	112
8.1.4	Biological Interpretations from ALFRED	113
8.1.5	Machine Learning	116
8.1.6	Final Remarks	117
A	ALFRED: Software Implementation and User Manual	128
A.0.1	Loading Window layout and functionality	128
A.0.2	Main Window layout and functionality	129
A.0.3	ROI Window	131

A.0.4	Microscope Specifications Window	133
A.0.5	Analysis and Calculations	133

Word Count:32917

List of Tables

1.1	Software Comparison	24
3.1	Skeletonisation: Time performance	42
4.1	Straight Segments: Testing	76
4.2	Theoretical Curvatures	77
4.3	Root Mean Square Error of Radius Calculations	77
6.1	Average Radius Measured	88
7.1	Data Augmentation Ranges	100
7.2	Network Parameters Explored	106

List of Figures

1.1	Simplified Neuron	14
1.2	Representation of a microtubule end	16
1.3	Neuronal microtubule organisation	16
1.4	Diagram of an axon swelling.	17
1.5	Primary <i>Drosophila</i> neurons stained for microtubules.	21
1.6	Microtubule disorganisation in <i>Drosophila</i> primary neurons.	22
1.7	CellProfiler Analysis of Circles	25
3.1	Initial Image vs Computer-Readable Format	31
3.2	Concept Illustration	33
3.3	Filter Explanation	34
3.4	Image transformation flowchart.	35
3.5	Applying mask without any filters	36
3.6	Examples of image processing into binary images	37
3.7	Vesselness and mask applied to channel	49
3.8	Comparison of masks	50
3.9	Vesselness of Disorganisation	50
3.10	Neighbourhood definition	50
3.11	Medial Axis Definition	51
3.12	Fast Marching Method: First steps	51
3.13	Skeletonisation of Sample Image	52
3.14	Skeletonisation: Boxplot of timed performance	53
3.15	Skeletonisation of Images	54
3.16	Path Finding: From Binary to Graph	54
3.17	Path Finding: From Skeleton to Path	55
3.18	Path Finding	56
3.19	Path Finding: Adding Points	57
3.20	Path Finding: Gaps	57
4.1	Parameter extraction from neuronal images.	58
4.2	Straight Segments: Euclidean Difference	60
4.3	Straight Segments: Example	61
4.4	Hough Transform: ρ and θ definition.	61

4.5	Hough Transform: Application.	62
4.6	Hough Transform: Plot of ρ, θ	63
4.7	Hough Transform: Implementation Flowchart	64
4.8	Hough Transform: Overlap Example	65
4.9	Hough Transform: Testing Iteration	65
4.10	Hough Transform: Testing Images	67
4.11	Curvature: Mathematical Definition	68
4.12	Curvature: Pixel Angles	69
4.13	Curvature: Pattern Fitting	70
4.14	Curvature: Function Fitting	70
4.15	Curvature: Gaussian Windows	72
4.16	Curvature: Smoothing Spline Fitting	72
4.17	Curvature Testing examples	73
4.18	Curvature Comparison	74
4.19	Curvature: Skeleton treatment	75
5.1	Experiment diagram	80
5.2	ALFRED Pipeline	81
5.3	Example of overlapping bounding boxes.	82
6.1	Manual axon length measurement	85
6.2	Axonal Length Comparison	89
6.3	Relative Axonal Length - Analysis	90
6.4	Manual disorganisation measurement	91
6.5	Axonal Disorganisation Area Comparison	92
6.6	Relative Disorganisation Area Analysis	93
6.7	Microtubule Disorganisation Index Comparison	94
6.8	Manual approximate radius measurements. Red circles denote the ap- parent loops, with the respective radius in nm in white. Adapted from Dr. Simon Pearce's work [99].	94
6.9	Curvature Comparison	95
6.10	Straightness Comparison	96
7.1	Data Clean-Up Pipeline	98
7.2	Microtubule Channel without clean-up	99
7.3	Image Processing with Ilastik	100
7.4	Example of an incomplete cleaning of the image.	101
7.5	Classification Architecture	102
7.6	Neural Network Architecture	103
7.7	Deep Neural Network Architecture	103
7.8	VGG16 Architecture	104
7.9	ResNet18 Building Block	104

7.10 Deep Learning Toolbox Interface	105
A.1 ALFRED: Load Window	136
A.2 ALFRED: Main Image Window	137
A.3 ALFRED: Channel Options	138
A.4 Bounding box tools	138
A.5 ALFRED: Highlighted Regions	139
A.6 ALFRED: ROI Window	140
A.7 ALFRED: Region Detection	141
A.8 ALFRED: Microscope Specifications Window	142
A.9 Scaling Factors	142
A.10 Saving options	142
A.11 Image Labelling	143

Abstract

Axons are the long, cable-like processes of neurons which form the nerves that wire our bodies. The maintenance of these delicate structures requires continuous parallel bundles of filamentous polymers called microtubules, which run all along the axonal core. In ageing and neurodegeneration, axonal microtubules often become curled and disorganised, causing detrimental accumulations of organelles and vesicles. The hypothesis of Local Axon Homeostasis suggests that the disorganisation of axon bundles is caused by the constant mechanical stress from the activity of motor proteins performing life-sustaining axonal cargo transport. In healthy axons, microtubule-regulating proteins prevent this from happening. If such proteins are dysfunctional, microtubules have a heightened probability of becoming disorganised. In this scenario, we would expect that microtubule phenotypes should appear similar, regardless of which microtubule regulatory proteins are absent; their images seem to suggest this. Therefore, my task was to develop objective parametric analyses to describe and compare microtubule phenotypes of different genetic conditions.

First, I tested a number of existing software packages (Imaris, CellProfiler, NeuronJ, FiberApp), none of which proved suitable for a number of reasons. Especially curvature descriptions of the curled microtubules were impossible with any of these software packages. I therefore investigated different algorithms that would allow such analyses, and chose the Smoothing Spline Fitting method, which I turned into computational code in MATLAB. Tests of these algorithms using spirals and ellipses of defined parameters revealed a fairly stable reproduction of their curvatures. In addition, I used Hough Transforms to establish tools to identify and analyse straight lines. In my validation tests using sample images, these performed better than classic metrics.

In the next step, I designed ALFRED (Advanced Labelling, Fitting, Recognition and Enhancement of Data) as user-friendly software in which the above algorithms can be applied. The development of ALFRED included strategies for image processing (from reduction of background noise to skeletonisation), ROI selection (with the help of MATLAB user interface tools) and path recognition (skeletons transformed into graphs and the shortest path algorithm is applied).

When using ALFRED to analyse biological images of neurons, length and area measurements were matching closely with manual analyses. Curvature and straightness analyses of neurons of different genotype revealed a clear distinction between normal and mutant neurons, and different mutant conditions revealed similar values, as is consistent with the hypothesis. The software is now available for wider application.

As a complementary approach, I attempted to compare different phenotypes using machine learning. Data clean-up of neuron images was performed with Ilastik, but no accurate outcomes were achieved in the time available. Two pre-trained classification networks (VGG16 and ResNet18) were used to classify different genotypes, but results were inconclusive, likely also due to the insufficient data clean-up of images.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

- i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the Copyright) and s/he has given the University of Manchester certain rights to use such Copyright, including for administrative purposes.
- ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made **only** in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.
- iii. The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.
- iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see <http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=24420>), in any relevant Thesis restriction declarations deposited in the University Library, the University Librarys regulations (see <http://www.library.manchester.ac.uk/about/regulations/>) and in the Universitys policy on Presentation of Theses.

Acknowledgements

If I were to mention every person that has contributed to this PhD, that would put the size of the thesis to shame. As such, I will try to make it brief:

First and foremost, I want to thank the people without whom there would have been no project, my supervisors Andreas and Matthias. Thank you for letting me derail so much from the original plan and still staying with me through it all!

I want to thank Andre, the best postdoc any student could ask for, and who made sure I got to the end. Thank you for everything, really - there was no way I'd get here without your help.

In terms of collaboration, I want to thank both Nuno, who helped with the curvature despite everything, and Iain, who supervised me through the Turing times! A very important acknowledgement goes to Rita, who drew the ALFRED logo.

Now, on a more personal and mental health level: thank you Ines for being the role model, Cristina for being the best dementor, Jill for being the best deskmate and Jo and Alex for the irl and virtual coffee breaks! To Liz and Rob for the wonderful breakfasts. To Alessio for all the games and incredibly horrible-yet-magnificent movies. To all the guys at the Turing for a great 6 months (plus extra covid time!) during my final year, as well as my London family who was stuck with me for a while! And to Mauricio, who's always been close even at a distance.

A very important acknowledgement from my heart, to the people that helped me at the most difficult of times: to Basilio and Paula, for making me feel part of the family when I was the most isolated in the world.

To the best mate with the worst face in the world, Jordie, thank you for the immense amount of patience and coffee.

Thank you, mum and dad (Isabel and Manuel, in case there are any questions), and everyone in my family who has been with me, for making me who I am, and helping me get here!

Maria, Piolhovski, Piolhoshka, thank you for being an inspiration and such a bright light in my life. You are my most favourite person in the whole world.

Chapter 1

Introduction

1.1 The nervous system, neurons and axons

Animals are complex organisms with complex behaviours that have to be coordinated and adapt to the needs of the body in response to the environment. This coordination is performed by the nervous system. Much work is dedicated to the study of its development, maintenance and disorders. This requires investigations at all levels of complexity, including the cellular and subcellular levels.

The key cell types contributing to nervous system function are glia cells and **neurons**. Neurons are very special cells. First of all, their shape is singular: Tree-branch like extensions extending from the cell body act as receivers of information (Dendrites, Figure 1.1). Long slender cable-like processes (**Axons**, Figure 1.1) are responsible for the transmission of information in form of action potentials along the body, either from the periphery to the brain or in reverse.

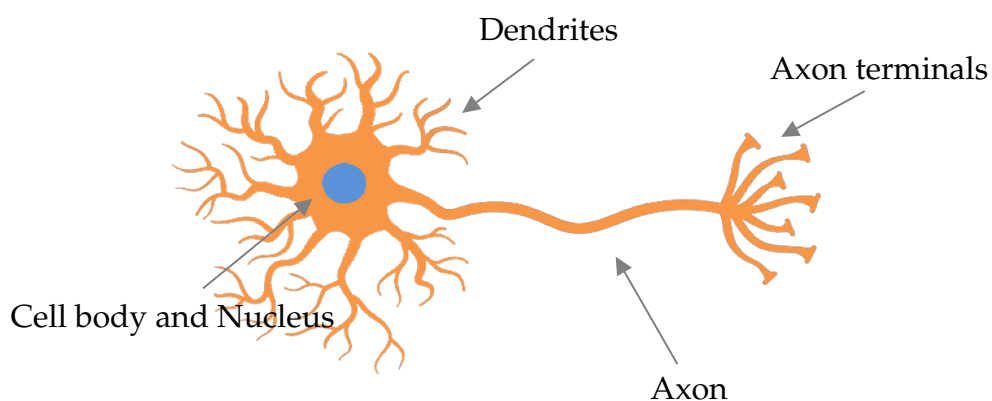


Figure 1.1: A simplified schematic of a neuron. The main parts are indicated by arrows: cell body, dendrites and axon, with the nucleus shown in blue.

Secondly, neurons are very delicate structures. To wire the human body, axons can be up to 1m long and only up to $15\text{ }\mu\text{m}$ in diameter [1].

Finally, neurons usually do not undergo turnover and have to survive for a lifetime, i.e. a century in humans: if neurons die or lose their axons irretrievably (e.g.

upon axon degeneration), their connections have to be taken over by other neurons in their surroundings; otherwise these connections are irreparably lost.

Even during healthy ageing, the human brain can lose up to 50% of axons [2], and still has to maintain its normal functionality. This axon decay is further enhanced in the presence of neurodegenerative diseases [3], particularly the group of premature axon decay - **axonopathies** [4].

These diseases can be of different types: either acquired, such as trauma [5, 6] or inherited, such as Charcot-Marie-Tooth [7] or Amyotrophic Lateral Sclerosis (ALS) [8]. Axon decay can either be a primary driver of neurodegenerative diseases or a symptom, i.e. dysfunction of axons can be potential cause or consequence of neurodegeneration.

Given their particular shape and lifetime, some questions can be posed to further understand axons:

- How can axons develop and then be maintained over such long periods of time?
- What mechanisms underlie axon maintenance?
- How do these mechanisms go wrong in ageing and disease?

1.2 The importance of microtubules for axonal physiology

An essential factor underlying their growth and maintenance of axons is the cytoskeleton (as the name indicates, skeleton of the cell). It comprises actin, intermediate filaments and microtubules. The focus of this work is on **microtubules**.

1.2.1 Biochemical and physical properties of microtubules

Microtubules are polymers formed by heterodimers of α - and β -tubulin (Figure 1.2). These heterodimers align in a head-to-tail conformation into protofilaments, so that microtubules have polarity: one end has α -tubulin (minus end, blue in Figure 1.2) while the other has β -tubulin (plus end, orange in Figure 1.2) [9]. Usually, 13 protofilaments align in parallel, forming a sheath that closes into a hollow tube of about 25 nm outer diameter ([10, 11]). In vitro, microtubules polymerise in the absence of aiding proteins. Polymerisation can happen also at the minus end, but at a much slower rate than at the plus end.

Microtubules are not stable structures, particularly at the plus end, where they are in a constant state of dynamic instability [12]. Balancing between two reactions (endothermic polymerisation and exothermic depolymerisation), there are three different conversions: **pause**, when the reaction happening comes to a halt; **catastrophe**,

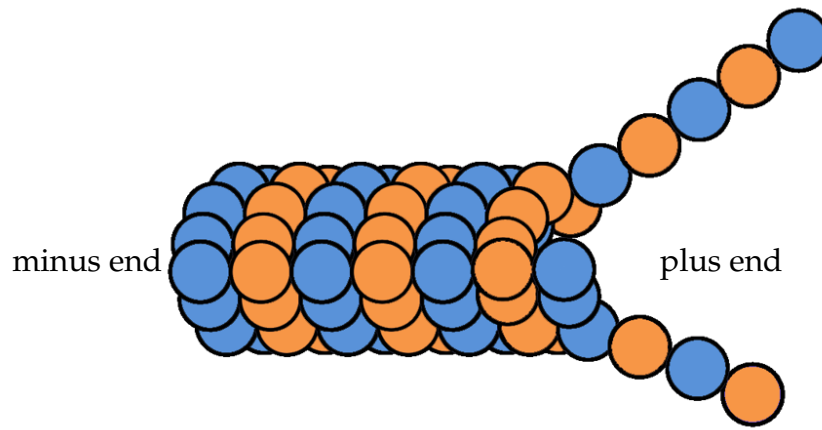


Figure 1.2: Representation of a microtubule end, with α - (blue) and β - (orange) tubulin. The plus end is the polymerisation site, where new tubulin dimers are added.

when the polymerisation or the pause turn into depolymerisation; **rescue**, when the reaction goes from depolymerisation to one of the other states. The dynamics are tightly controlled by different classes of microtubule binding proteins (MTBPs) [13, 14]. Although there is some knowledge regarding their molecular functions, it is little understood how the different MTBPs are orchestrated to regulate microtubule dynamics in meaningful ways. It is, however, important to note that many have genetic links to developmental or neurodegenerative diseases [15].

1.2.2 Roles of axonal microtubules

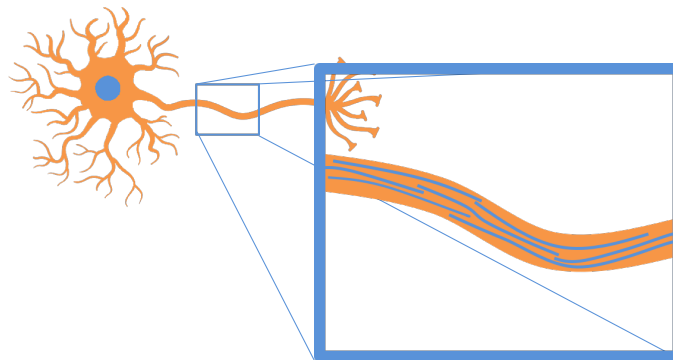


Figure 1.3: Neuronal microtubules organised in a discontinued, parallel bundle along the axon.

Microtubules act as a structural backbone in axons. Thus, microtubules (Figure 1.3) are organised in parallel bundles in axons, in that all microtubules have their plus-end towards the axon terminals. In contrast, microtubule bundles in dendrites are anti-parallel, i.e. have mixed orientation. Microtubule orientation is a vital feature for one of their most important functions in neurons: axonal and dendritic transport,

driven by molecular motors that have a tendency to move towards or against the plus end.

Microtubule bundles have a significant role throughout the lifetime of the axon: through extension at the leading tips or forming off-track daughter bundles they drive axon growth and branching, and their destabilisation correlates with axon retraction [16]. As the axon becomes mature, the functional importance of microtubules is not reduced: they are crucial architectural components and fundamental for axonal transport, sustaining axonal physiology and providing materials and signalling processes essential for axonal maintenance [4].

Microtubules are essential during axon regeneration. It was shown that the stabilization of microtubules promotes regeneration *in vivo* [17] as shown by studies of microtubule stabilizing drugs in spinal cord injury of rodents [8, 18]. Axonal transport is also fundamental for a proper regeneration. For example, in *Drosophila*, it was shown that molecular motor kinesin-1 is involved [19].

Notably, in both ageing and some forms of neurodegeneration, axons can develop pathological swellings. In these swellings, microtubule bundles have been reported to become disorganised (Figure 1.4) which can correlate with the functional deficiency of certain MTBPs [20–22].

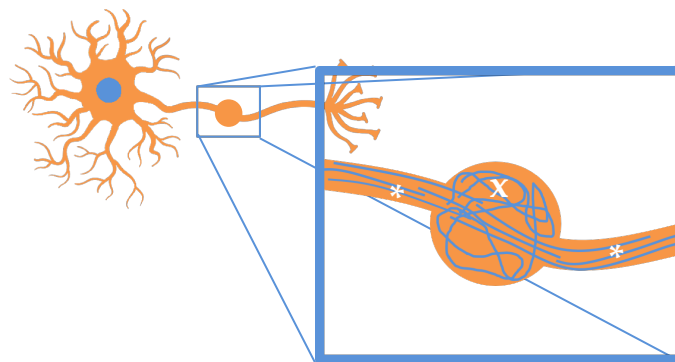


Figure 1.4: Diagram of an axon swelling. The zoomed image shows microtubules (blue) in bundled conformation in healthy areas of the axon (asterisk), and taking on disorganisation within a swelling (cross).

As will be explained in the next section, there are new and improved means to explain such disorders, and with the molecular knowledge we have about these proteins, we can have a deeper understanding of the machinery that regulates the microtubule network in neurons, explaining also how areas of disorganisation can form.

1.2.3 Regulation of axonal microtubules

In vitro, microtubules are stiff rods with a very high persistence length in the order of millimetres. However, when observing microtubule in gliding assays *in vitro*,

where microtubules move freely on carpets formed by microtubule-associated proteins, there are conditions (e.g. high microtubule density), where they show a frequent tendency to curl with diameters in the lower micrometre range [23]. These *in vitro* conditions might provide mechanistic explanations as to why microtubules might fail to keep their straight bundle formation in axons: high densities of kinesins might impose a constant bias for microtubules to take on curled conformations.

In vivo, however, there are different types of MTBPs that bind microtubules in different places, regulating microtubules in ways that prevent curling. As it would be extensive to cover all of them, this section will focus on three examples that are best studied and relevant to the content of this thesis.

Even though microtubules are very dynamic all throughout their structure, the main focus of dynamics is at the tip where microtubules extend through polymerisation and where this extension needs to be regulated with respect to speed and directionality. As such, it is a large field to study, as many proteins are involved in the processes occurring. Plus end proteins, or +tips, have a self-explanatory name: they are the proteins that track and attach to the plus end of microtubules. They belong to a complicated machinery responsible for different aspects in a microtubule's life, either by promoting or inhibiting their polymerisation or guidance. The main idea of the Polymerisation Chaperone Hypothesis [24] is the existence of functional redundancy within a network of proteins that interact with each other to form a protein superstructure that regulates the plus end. Due to the redundancy within this superstructure, the loss of single proteins does not necessarily have much impact [14]. However, there are some types of proteins which have a more central role in the machinery than others, such as end-binding proteins (EB), spectraplakins or XMAP215.

Of these, XMAP215 acts as a polymerase localising to the microtubule plus end, where it promotes polymerisation [5]. Through driving polymerisation, XMAP215 facilitates the binding of EB proteins ([25] and references therein). EB proteins in turn recruit large cytoskeleton-interacting proteins called spectraplakins [26]. Whilst binding to EB proteins at the tip of polymerising microtubules with one end, spectraplakins can bind actin at the axonal surface with the other. In that way, they can guide extending microtubules in parallel to the axonal surface into well-aligned bundles [26, 27].

Furthermore, microtubules and the bundles they form need to be stabilised. According to Voelzmann et al [14], there are different types of microtubule stabilisation that might occur in the axons. One mode is to cross-link microtubules into bundles through proteins such as tau, MAP2 or MAP1B (reviewed in [1, 11]), another is to stabilise them by protecting against the action of depolymerases or microtubule severing proteins [28]. These comprise severers such as katanin, spastin or fidgetin [29], or they can be specialised molecular motor that can drive depolymerisation, including proteins from the kinesin-13, kinesin-8 or kinesin-14 families [30, 31]. But there

are also kinesins, such as members of the kinesin-4 family, which decrease the microtubule turnover at the plus end.

Another important function of axonal microtubule bundles, is to provide the highway for transport inside the cell. Molecular motors are the proteins responsible for the transport along axonal microtubules, which can happen over long distances. There are two families of microtubule-binding molecular motors, kinesins and dynein/dynactin, whereas myosins drive transport or movement along actin filaments [32]. Dyneins move along microtubules in the retrograde direction (from the plus to the minus end, i.e. from the axon terminals to the cell body), whereas kinesins move anterogradely. Their functions in axons are not only about the transport of cargo. For example, dyneins also bind to the membrane cytoskeleton, and might be responsible for microtubule movement along axons [32].

The kinesin superfamily consists of fourteen subfamilies [15]. Kinesin families have several interesting properties and functions in axonal microtubules that make this molecular motor a particular focus of study to understand their regulation throughout the lifetime of the cell. Members of the kinesin-1, -2, -3 and -4 families are involved in axonal transport towards the plus end of microtubules, whereas others play roles in microtubule stabilisation or depolymerisation (see above). During this thesis, I will use functional data relating to kinesin-1. The kinesin-1 family is of importance since its dysfunction leads to different inherited pathologies such as Charcot-Marie-Tooth disease or spastic paraplegia type 10 [33]. Mutations in its motor domains inhibit microtubule-dependent transport [34], revealing one potential mechanism underpinning its roles in pathology.

Kinesin-1 also plays roles during axonal growth, and their impairment correlates with reduced growth [35, 36]. Since they have the important task of transporting essential cellular components to the growing tips, this likely interferes with the provision of materials (e.g. tubulins or membrane lipids) or important mitochondria providing the energy to drive growth processes.

1.2.4 Key Problem and *Drosophila* as a solution

It is rather clear that understanding how these mechanisms jointly regulate microtubules is an important but difficult task, given how complicated it is to study intracellular components. An important strategy is provided through using the model organism *Drosophila melanogaster* for studies of axonal microtubule regulation.

Work with *Drosophila* is cost-effective, it is easy to keep the flies, and they have a short generation cycle, speeding up project work. Their efficiency is uncontested, consequently having a rapid research progression [37–39]. Compared to higher organisms, *Drosophila* genes are there in lower copy numbers, it is highly efficient to

combine mutations or transgenic constructs in the same flies to study functional overlap and cooperation (referred to as combinatorial genetics) [40], and there is the possibility of performing unbiased screens for genes that are involved in certain biological processes. Notably, 75% of human genes that are connected to diseases are believed to have a fly homologue [41, 42], so that biological processes studied in the fly often translate into understanding in higher organisms including humans (thus explaining that 9, arguably 10, Nobel laureates were awarded for their work in *Drosophila* [39]).

Importantly, there is already a large number of available resources and knowledge, which is organized in large databases. With these resources, it becomes easy to genetically manipulate the organisms, as almost every gene can be mutated and fly lines are already largely available that can be used to manipulate almost every *Drosophila* gene. The experimental studies in *Drosophila* can be done both *in vivo* and within cell cultures, which allows complementary experimental approaches [27].

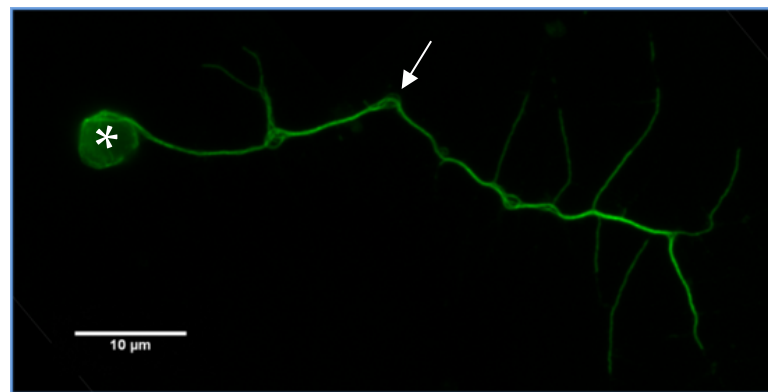
All of the aforementioned characteristics make *Drosophila* a useful research tool, especially for neuronal studies. For example, due to their genetic manipulations and high sequence conservation of the microtubule binding proteins with their mammalian genes, some important mechanisms were uncovered regarding guidance, nerve branching and axonal transport, among others [11]. Importantly, the genetic redundancy of some of the MTBPs mentioned above is particularly low, which facilitates the study and understanding of these proteins [6]. These studies can either be done via systemic readouts such as axonal length achieved over a certain period, or with live imaging of the cytoskeletal dynamics which is particularly facilitated by the cell cultures mentioned above [28]. In the following section I will discuss important concepts that were deduced from work with *Drosophila*.

1.3 The model of local axon homoeostasis

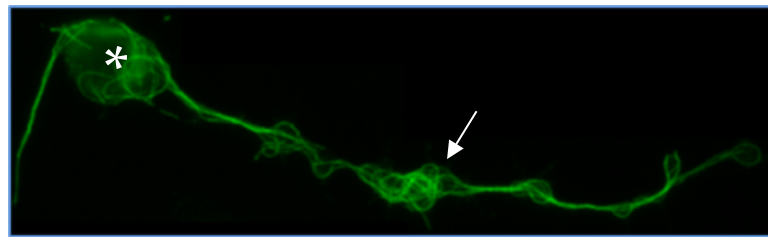
As mentioned before, the parallel microtubule bundles in axons can often be disorganised (Figure 1.5). This disorganisation is likely to cause traffic jam of cellular components (such as mitochondria), but may also influence the proper propagation of action potentials, as the diameter changes ([43, 44] and references within).

Using *Drosophila* as a model, my host group has found many MTBPs that, when mutated, can cause microtubule disorganisation in the axons of primary neurons [28, 45, 46]. Based on a wealth of data, the group has come up with a hypothesis to explain the occurrence: the model of **Local Axon Homoeostasis** [11, 14], which will be briefly explained here.

As described before, microtubules are highly dynamic polymers and suffer constant turnover by displaying constant events of polymerisation, depolymerisation and severing.



(a) wild-type



(b) Mutant

Figure 1.5: Primary *Drosophila* neurons stained for microtubules, showing a higher level of disorganisation in the mutant. (a) wild-type neuron, (b) mutant neuron. Asterisk denotes the cell body, arrows point to one of the disorganised areas in each neuron. Shown area of width $24.6\mu\text{m}$.

As mentioned previously, *in vitro* studies show that, under certain conditions, dynamic microtubules can curl up rather than form the usual straight rods [23]. Axons display conditions that favour microtubule curling, such as high density of microtubules, high density of kinesins, and the presence of physical barriers in the confined space of axons. This might explain why microtubule disorganisation is found in axons but far less frequent in other cell types. The assumption therefore is that, by default, microtubules in axons tend to obtain a disorganised state. In order to keep the organised bundles in axons, microtubules rely on their regulators.

The taming of microtubules into bundles can happen by several mechanisms, for example through cross-linking them via MAPs or tau, or guiding their extension via EB proteins and spectraplakins (see Section 1.2.3). In the absence of this function, as is the case in the *shot*³ mutation, microtubule curling is observed (Figure 1.6a). Another mechanism is the elimination of off-track microtubules that leave the bundle and approach the axonal surface. Such a surface associated microtubule inhibitor is *efa6*^{ko} which also causes curling when absent (Figure 1.6b). Surprisingly, also loss of the transport motor kinesin-1 (called Kinesin heavy chain/Khc in *Drosophila*) shows the curling phenotype (*khc*^{27/27} in Figure 1.6c). Although kinesin-1 is expected to be one of the motors damaging and curling microtubules through its transport activity, its absence does surprisingly not relieve microtubules but causes damage. To explain this, experiments suggest that loss of kinesin-1-mediated transport depletes axons

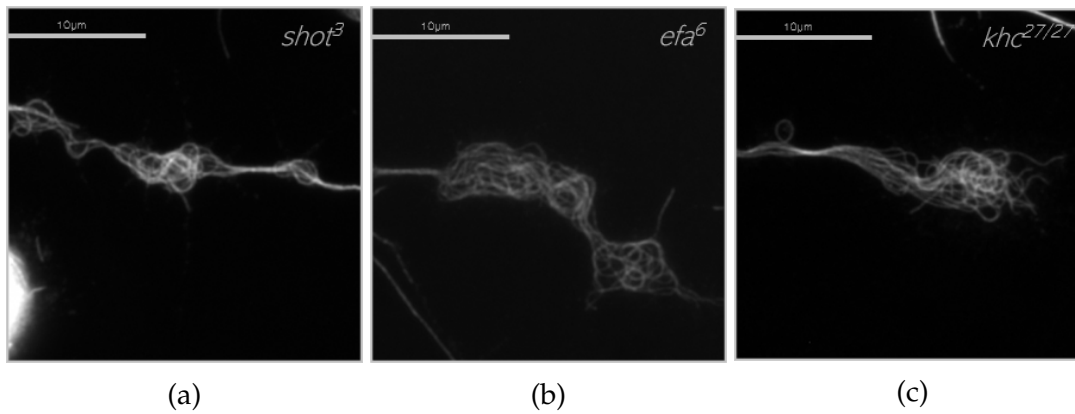


Figure 1.6: Microtubule disorganisation in *Drosophila* primary neurons of three genotypes with mutations on different proteins: (a) *shot*³ (a spectraplaklin), (b) *efa*6^{ko} (a surface associated microtubule inhibitor) and (c) *khc*²⁷ (kinesin-1). Albeit the mutations occur on different pathways, the resulting disorganisation appears to be similar between them. Staining for microtubules. Scale bar 10µm.

of important materials and physiological conditions that are needed to maintain microtubule bundles, whilst other kinesins are still performing transport activities that continue to damage and curl the bundles (for details see [4]). Although these three mechanisms are independent from each other, mutations to each of their genes in *Drosophila* causes a similar disorganisation phenotype (Figure 1.6). If the disorganised microtubules are objectively similar in all phenotypes, then it is a strong indication that the curling is driven by the microtubules in response to the specific axonal environment. It is not caused by deregulation of MTBPs, which merely function to antagonise microtubule curling.

1.3.1 Key Aims

My aim was to test the hypothesis that biased behaviours of axonal microtubules are the common cause for their disorganisation across the various mutant conditions of MTBPs identified so far to display this phenotype. I would predict that the patterns of microtubule curling are qualitatively similar between these mutant conditions. My aim was therefore to obtain unbiased descriptions of axonal microtubule disorganisation in neurons using unbiased image analysis strategies and a range of descriptive parameters. Several parameters are being considered: axon length, disorganisation area and the shapes that microtubules form in these areas for example regarding the shape of their curvature. I focused on the following key questions:

- Can the image analysis become automated, i.e. more objective and less user-dependent?
- Can we retrieve important parameters from the images, such as straightness or curvature of axons and their microtubules?

- Are the phenotypes qualitatively similar between the mutant conditions of the different MTBPs?

1.4 Available strategies and software packages to analyse axonal MT phenotypes

When analysing the images, there are some basic observed properties for the disorganisation: there is the presence of loops, with occasionally straight lines within. However, microtubules are not individually distinguishable, only in bundles. Furthermore, resolution is quite low and due to the nature of the fluorescence imaging, there is no 3D information provided.

As such, microtubule bundles have to be described as two-dimensional line objects in an image.

Current Approach

The analysis previously done within our group has been based on two key parameters analysed: the axonal length and the absolute areas of disorganised regions. The main metric to compare phenotypes is the ratio between the combined areas of disorganisation in an axon per its length.

The measurements are performed manually using Fiji/ImageJ [47], with personalised macros to ease and uniform the analysis by all users in the group. The conversions and calculations are then performed on a spreadsheet and statistically analysed with Graphpad Prism [48]. The full methodology is detailed in Section 2.3.1.

This analysis does not provide, however, any insights into shape characteristics of the disorganised regions - whether they have a higher or lower incidence of loops, or straight segments.

State of the Art Software

There are a plethora of image analysis software available with different purposes and goals. Table 1.1 is the overview of bioimage analysis software for 2D fluorescence images, available for users at the School of Biological Sciences, University of Manchester.

Imaris is a multifaceted software for image analysis. Particularly, there are different packages available for the users, according to the type of analysis needed. The one mentioned here is Imaris for Neuroscientists, as it allows specific features for neurons. The software is very good for videos and measurements of processes along the neurons, particularly with the automated filament recognition. The software has

Table 1.1: Feature comparison between the most widely used bioimage analysis software. Checkmark indicates presence of the feature. Question mark refers to not enough information available to determine presence of feature. Availability corresponds to where can the user analyse their images, either on their individual computer, at the bioimaging facility local computers or both. When the software is not free, the price widely varies with the needed modules.

Software	Features				Availability		Free	Open Source
	Length	Curvature	Straightness	Automated Detection	Own Computer	Imaging Facility		
Imaris [49]	✓	(?)	✓	✓		✓		
CellProfiler [50]	✓	(?)	(?)	(?)	(?)	✓	✓	✓
FiberApp [51]	✓	✓	(?)	(?)	✓	(?)	✓	✓
Fiji/ImageJ [47]	✓			(?)	✓	✓	✓	✓
Fiji/NeuronJ [52]	✓			✓		✓	✓	✓

a straightness calculation for each segment recognised (axons or other processes of the cell). In addition, it finds and selects closed circular loops that approach a perfect circle in filaments but does not provide any type of curvature. This is, however, a paid software - and only recently provided a conditional satellite license for users (i.e., allow individual users to access the software from their computers rather than using the ones provided by the local bioimaging facility). Furthermore, an extra package is necessary if one intends to add their own analysis in programming languages like MATLAB or Python.

CellProfiler allows the user to build pipelines to analyse images in bulk. However, it was designed for volumetric cells, or with a recognisable geometrical shape. In the case of filaments, it underperforms and does not identify correctly the relevant parts of the image. While it does not compute curvature, it provides two shape descriptors: eccentricity (distance of the shape to a circle) and form factor (calculated as $\frac{4\pi \text{Area}}{\text{Perimeter}^2}$). Each of these show the proximity of the shapes to a circle. Going through the pipeline using digitally created circles (Figure 1.7a) the program detected 98 shapes in the image, rather than the 16 circles. Furthermore, it provides largely divergent values than the expected - 0 for eccentricity, 1 for form factor - as seen in Figure 1.7b.

The software that seemed most suitable for the intended analyses was FiberApp. This software identifies fibres and calculates morphological features, which includes curvature. However, the detection of each fibre is either done manually (the user has to click all along the fibre, including all the small curves) or it will lose information. Our images require fine calculation and the inclusion of small curvatures that would otherwise be ignored by FiberApp. As it was the only option that retrieved curvature, the results from the analysis are further discussed in Section 4.3.

Finally, the current software used by our group also provides a plug-in built for neuronal images, NeuronJ. However, the path recognition (even though automated)

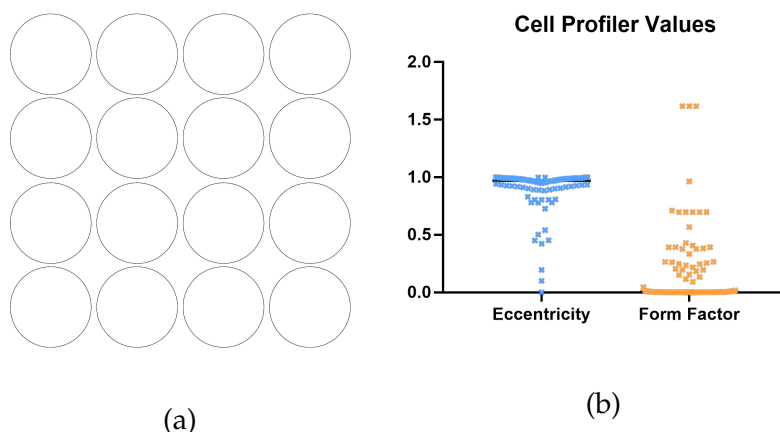


Figure 1.7: Testing Cellprofiler software on digitally created circles (a) and their measurements of eccentricity and form factor (b). The number of sections analysed was 98. The circles should have 1 section each, form factor of 1 and eccentricity of 0.

is time consuming and needs user supervision to guarantee a successful result. Furthermore, it does not provide any curvature calculations or recognition of dense regions as disorganised microtubules.

1.5 Aim

As is suggested by the model of local axon homeostasis, understanding the nature of microtubule disorganisation will contribute to our understanding of how axons are maintained throughout the lifetime of an organism. Furthermore, insights into how similar the regions are between different genotypes provide a better idea as to whether microtubules are the culprits of the disorganisation, or whether the malfunctioning of the MTBP networks regulating them actively impose these non-customary shapes within the axons.

My task was then to develop ways to neutrally describe the microtubule disorganisation phenotypes and compare them. Furthermore, I aimed to develop more efficient ways to speed up data analysis as a strategy to further obtain a larger data pool on the basis of which to refine the hypothesis.

On this project I developed a user-friendly software package to efficiently import and analyse images of microtubule disorganisation phenotypes, allowing a neutral parametric description of neurons and, particularly, the microtubule disorganisation regions. Additionally, I aimed to assess whether classification across different genetic and experimental conditions is possible with the available datasets already acquired, either using parameter retrieval-based approaches or machine learning.

Before continuing, it is important to highlight the overall structure of the thesis. In the next chapter (Chapter 2), I begin by describing all software versions used and the existing methods of acquisition and analysis of the biological images. It is important

to first process the images, as described in Chapter 3. The next chapter (Chapter 4) describes how to retrieve the quantitative information from the processed images. The software that wraps all the previous processing and analysis is described in Chapter 5. Next, using the software, the biological images are analysed in Chapter 6. In the following chapter (Chapter 7), machine learning methods are applied to a biological image dataset. Finally, in Chapter 8, the final discussion and conclusions are explored.

Chapter 2

Languages, external resources and images used

Before starting the description of the processing and analysis, it is important to describe the common methods for all the proceeding chapters, both software and biological images.

2.1 MATLAB

While the first versions of the software were developed with MATLAB version R2017a, the final adjustments and deep learning were done with version R2019b. All tests were done on Windows.

2.1.1 MATLAB Incorporated Functions

The user interface (UI) was developed using MATLAB GUIDE [53], an integrated development environment (IDE) for user interfaces that automatically creates functions for each of the visual components, such as the buttons or sliders present in the UI.

Whenever there is a performance assessment, time is measured with `tic toc` [54, 55].

2.1.2 Functions from Other Sources

Bioformat

Version used: [56]

This is a standalone Java library that allows importing a large number of common image format acquired by biology experimentation devices. In addition to loading the files, it can interpret and handles the metadata available with the images, i.e., the information that arrives associated with the image definition at acquisition point, such as pixel resolution among other important features.

Image Graphs

Retrieved from [57]

In order to do most of the calculations to retrieve parameters from images, it was necessary to convert them into graphs. This function converts a binary image into a graph using neighbour relationships, i.e., each pixel of value 1 is a node and the edges are connections to the 8 neighbours that it possesses.

Other functions from the same package were used, particularly **checkConnectivity**, that finds all of the connected components in a graph.

Vesselness 2D

Version used: [58]

Key function for the recognition of structural features in images. Allows a very quick recognition of the structures in the images, which we found to be particularly relevant in neuronal microtubule stainings, for example. With the application of a series of filters within the function, a monoscale image is retrieved which can be quickly converted into binary using a simple threshold. Further explanation is done in Section 3.2.1.

2.1.3 Deep Learning Toolbox

The methods performed in Chapter 7 were done using the Deep Learning Toolbox, from MATLAB [59].

Image augmentation was done using the function `augmentedImageDatastore`, with `imageDataAugmenter`.

The overall training of the different neural network algorithms was done using `trainNetwork`, personalised with `trainingOptions` for the exploratory data analysis. Each network used (VGG16 and ResNet18) was pre-trained with the ImageNet dataset[60]. The classification layers were trained on the microscopy dataset obtained within the group. Further explanation in Chapter 7.

2.2 Outsource Software

2.2.1 Ilastik

The software Ilastik[61] was used to perform the segmentation of the images 7. The aim was to select the relevant sections (neurons of interest) from the background and other cells.

The Pixel Classification workflow [62, 63] allows an interactive segmentation training, with results shown in real-time. As such, it allows the tuning of the training as each new image is manually labelled.

Feature Selection was done with edge and texture 2D options selected - it is time consuming but the intricacy of the microscopy images demanded an attention to all details. The colour option was overlooked as the images of interest were grayscale. After the training with a few images and manual selection of the regions of interest, batch processing was performed to the remaining images.

2.3 Biological Images

Fly Stocks

Fly stocks were maintained on standard food vials with yeast supplementation [64]. The Oregon R strain was used as the wild-type control to all mutant strains.

The following mutants were used:

- khc^8 [65]
- khc^{27} [66]
- $efa6^{ko}$ [67]
- $shot^3$ [68]

Blue genotype was used only for the Machine learning, red genotypes were used for the ALFRED vs Manual comparison and orange genotype was used in both.

Cell culture protocol and Imaging

Primary neurons were generated following procedures described in [40, 64, 69]. Neuronal cell cultures were generated from stage 11 *Drosophila* embryos, and incubated at 26°C for varying time points depending on the experiment.

For the images analysed in Section 6.2, the cultures were grown for 1 day and then fixed in 4% Paraformaldehyde in 0.1 M sodiumphosphate buffer pH 7.2. Cells were subsequently washed three times with 0.3% PBT (TX-100) and incubate for 1 hour with mouse-derived anti-tubulin antibody (DM1A, 1:1000) at room temperature. After 3 washes with 0.3% PBT (TX-100), cells were incubated with anti-mouse Alexa 488 antibodies (1:200) and Phalloidin-TRITC (1:1000) to label microtubules with Alexa 488 and Actin with Phalloidin-TRITC.

Alexa488-signal (microtubules) were saved in the green image channel, Phalloidin-TRITC (Actin) signal in the red image channel.

The experiments were performed and images acquired by Dr. André Voelzmann.

For the images used in the machine learning methods, primary neurons were cultured for 5 days. Cells were fixed in 4% Paraformaldehyde in 0.1 M sodiumphosphate buffer and stained for alpha-tubulin (blue image channel) and nuclear neuron

specific protein elav (green channel). The images were acquired using an AxioCam MR5m, 100x objective, 1 optovar and 2x2 binning. Experiments and imaging were performed by Dr. Yuting Liew.

2.3.1 Manual Analysis

Up to date, image analyses and measurements were performed using FIJI/ImageJ [47]. The two values retrieved from each neuron are axon length and disorganisation area.

Axon length is measured by longest neuronal protrusion, measuring microtubules from the edge of the cell body to the most distal tip of axonal microtubules. This is done in FIJI using the segmented line tool by clicking along the axon and calculating the sum of straight distances between each point. Furthermore, FIJI allows the conversion from pixels to micrometres.

The disorganisation area is calculated as the sum of all areas of disorganisation along the axon, manually marked with the freehand selection tool.

Statistical analyses were performed using GraphPad Prism software, version 9.0.0 [48]. The data of measured parameters was non-parametric (i.e., a model was not defined *a priori* and is determined from the data), and did not follow a normal distribution, so the Dunn's Multiple Comparison test (in a Kruskal Wallis test) was applied. The null hypothesis states that, for any pair of values retrieved from two sets respectively, there is the same probability of either being the larger value. A low p-value indicates that there is a higher chance of one of the datasets having numbers larger than the other.

In the dataset used in this project, the p-value is the probability of obtaining the observed difference in the outcome measure given that no difference exists between treatments in the population. In other words, low p-value indicates that the observed difference does not come from cell populations which do not have any difference due to their treatment but is rather due to a real difference between the control and the mutation datasets.

Chapter 3

Image Processing: The steps from acquisition to binary shape

3.1 Background and Rationale

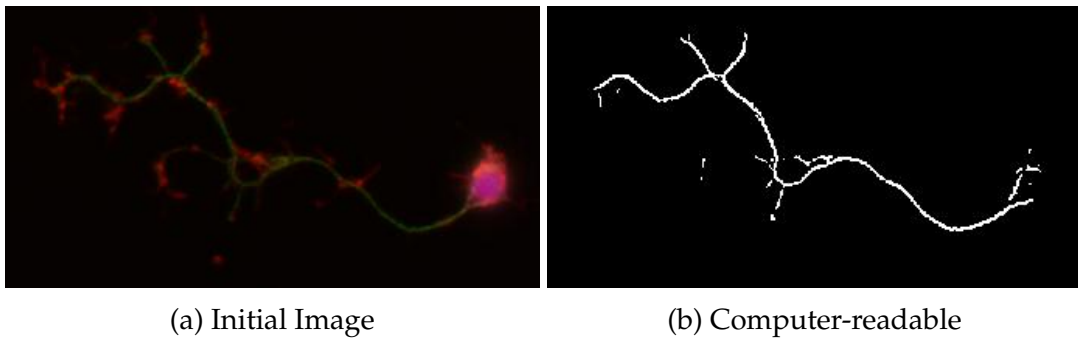


Figure 3.1: Example of a processed image of a neuron. (a) RGB image of a *Drosophila* primary neuron acquired on a fluorescence microscope, stained for actin (red), tubulin (green) and nuclear Elav (blue). (b) mask of the green channel, i.e., of the microtubule networks in this neuron. Image acquired by Dr. Yu Ting Liew. Shown area of width $43.6\mu m$.

The first step towards computer-based image analysis is the translation of the original images (Figure 3.1a) into formats that allow a user-independent quantification of parameters (Figure 3.1b). The aim of this chapter is to outline the methods used to obtain a final image that represents the experimentally obtained images as accurately as possible, to be used as a template for computational interpretation and quantification.

Before delving into the algorithms, it is important to briefly review some concepts.

The term **image** in this thesis usually refers to fluorescent microscopy images of primary neurons of the fruit fly *Drosophila* in cell culture (Figure 3.1a). Images acquired in the group are either monoscale, with only one channel (Figure 3.2a) or have multiple channels. The most common type of multi channel images is **RGB** (Figure 3.2b), composed of three channels (red, green and blue, Figure 3.2 c-e).

In computational terms, this means there are three overlapping matrices of size $M \times N$ that represent what can be actually seen, i.e., an image is an $M \times N \times 3$ matrix of double precision values that give intensity of the colour in that spacial position.

Each **pixel** of an image is its basic element of spacial precision, whose intensity is stored in a matrix with $M \times N$ pixels. In the most common type, an 8 bit image, this means that any pixel will have an intensity value between 0 and 255 for each of the channels¹ (Figure 3.2 f-h). For example, the middle pixel on Figure 3.2b (highlighted in yellow) has the RGB value of (0, 0, 255), which corresponds to null intensity in the red and green channels, and maximum in the blue channel, producing the colour blue.

The way biological images are acquired using fluorescence microscopy usually implies that each colour channel corresponds to a different component of the cell. For example, the microscopic image presented in Figure 3.1 shows an antibody staining for tubulin highlighting the neuronal microtubule networks (one particular cytoskeleton component of neurons) in the green channel, an antibody staining for the cell nucleus in blue and a third staining for the F-actin network (a further cytoskeleton component of neurons) in red. Although images are represented of three superimposed matrices, we only need to focus on one at a time (Figure 3.2 b versus c-e). As one complication, images will frequently show some level of background noise and artefacts that could potentially skew the results of measurements.

When an image is described as **binary**, it is composed of only one matrix and the pixels have one of two values, generally 0 or 1. Two types of binary images will be used:

A **mask** (Figure 3.2, Binary: Mask) is a binary image where each pixel is logical: either belongs to the shape we are interested in or not. For example, getting a mask for the neuronal images would mean each pixel would either be a neuronal component (`True` or 1) or background (`False` or 0). One way to approach the background problem is to apply a threshold, where all pixels of the image with a value above the threshold will be considered 1, whilst everything else will be 0. For the masks in Figure 3.2, the threshold applied was 100.

A **shape** will be the pixel-wide contour, or **skeleton**, of the objects in images (Figure 3.2, Binary: Skeleton, where the skeletons were obtained considering 4 neighbours). In the skeleton, each pixel will only have a maximum of two neighbours in their vicinity. Whether considering 4-neighbour or 8-neighbour depends on the type of image and the algorithm. Furthermore, the decision on which pixel to choose is also an important point of the algorithm. This considerations will be taken into account further along in section 3.2.2.

The first step therefore relies on finding suitable algorithms that will not only enhance the important features of a channel for meaningful translation into a binary

¹In an 8 bit image, each pixel is represented by one byte, i.e., 8-bits. This means a range of 2^8 numbers, i.e., from 0 to 255.

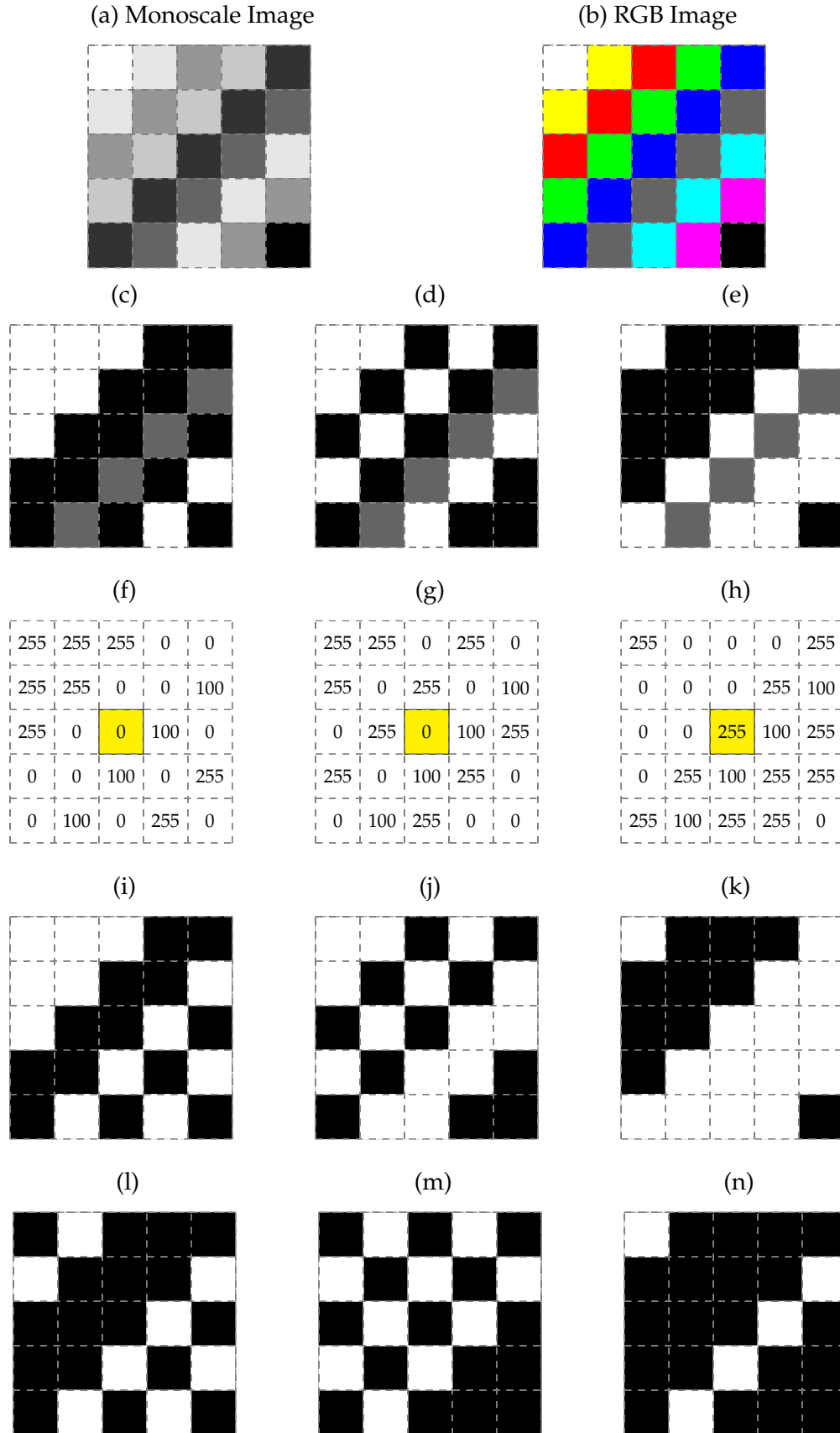


Figure 3.2: Schematics of digital images. Images are either monoscale (a) or colourful/RGB (b). The latter can be split into three channels (c red, d green, e blue), also shown as numerical matrices (f, g, h), where the highlighted pixel has the RGB value of (0,0,255) and appears blue in the composite image (b). There are the corresponding binary masks obtained with a threshold of ≥ 100 (i, j, k) and a possible skeleton for these (l, m, n). In (i-n), the relevant pixels are white on black background. The dashed line represents the separation of pixels.

mask, but potentially reduce any noise without losing information as judged by the experimentalist. These will be further explained in Section 3.2.

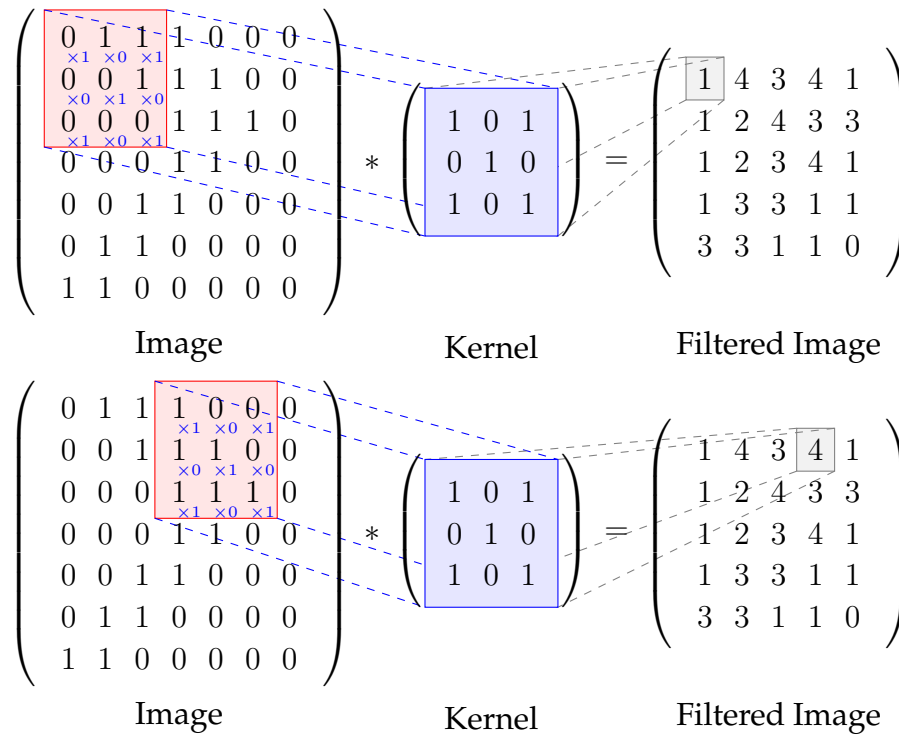


Figure 3.3: Example of a smoothing filter applied to a binary image, in matrix form. The red square on the image matrix corresponds to the pixels that will be multiplied by the filter matrix (or kernel). The sum of these multiplications will give the new value in the filtered image.

In addition to setting a threshold, other methods of filtering can be applied, for example smoothing filters (Figure 3.3). These filters apply a simple computational convolution: addition of each element of the image to its local neighbours, weighted (or multiplied) by the kernel. I.e., each element on the kernel will multiply each element of the image section, and the sum of all these values is the new filtered value.

On the top example, the new value would be:

$$(0 \times 1) + (1 \times 0) + (1 \times 1) + (0 \times 0) + (0 \times 1) + (1 \times 0) + (0 \times 1) + (0 \times 0) + (0 \times 1) = 1 \quad (3.1)$$

The filter acts as a window that will go through all of the pixels in which all the image neighbours contribute to the multiplication. Note that the neighbourhood calculation is done with the values of the pre-filtered image, and the newly calculated values are stored in a new matrix.

3.2 Shape Extraction from Images

In this section, the steps in Figure 3.4 from the colour image in the top into a skeleton, conserving as much relevant information as possible, will be detailed.

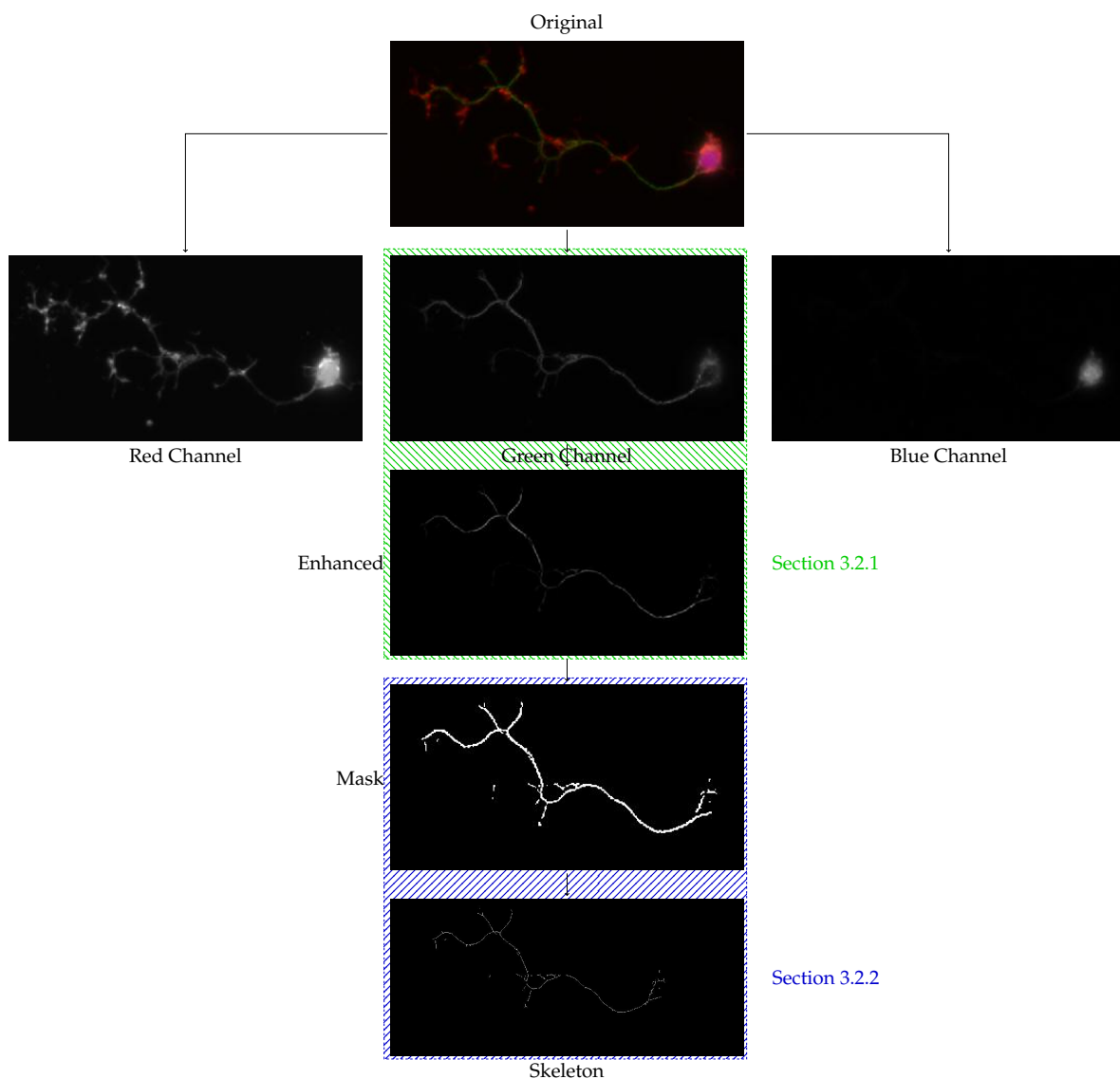


Figure 3.4: Flowchart of the shape extraction process, from the original RGB image to the skeleton. As the main focus is on microtubule networks, further processing in this case occurs only in the green channel. Green fields denote the procedures referred in Section 3.2.1, and blue fields the procedures referred to in Section 3.2.2.

As the work is based on the quantification of microtubule networks, the colour channel chosen will be in accordance. So, in the case of Figure 3.4, we are interested in the microtubule channel (green), which will be dealt with in the following two sections, describing the implementation of mask (Section 3.2.1) and skeleton generation algorithms (Section 3.2.2).

3.2.1 Identifying a suitable filter

In this project, filters have to be applied for two main reasons. First, the resolution of microscopic images is usually limited: in some cases, there are not enough pixels in the camera to describe the details, consequently merging information into a single pixel, regardless of the magnification. Furthermore, even when the number of pixels is sufficient, there are components of the cells that are even smaller than the wavelength of the fluorescent light.

Second, images may have very low levels of contrast, as well as background intensity values not completely distinguishable from the relevant components. When applying a threshold mask to the original images (as in Figure 3.5), a lot of information could either be lost, or too many background pixels would be included in the image analysis.

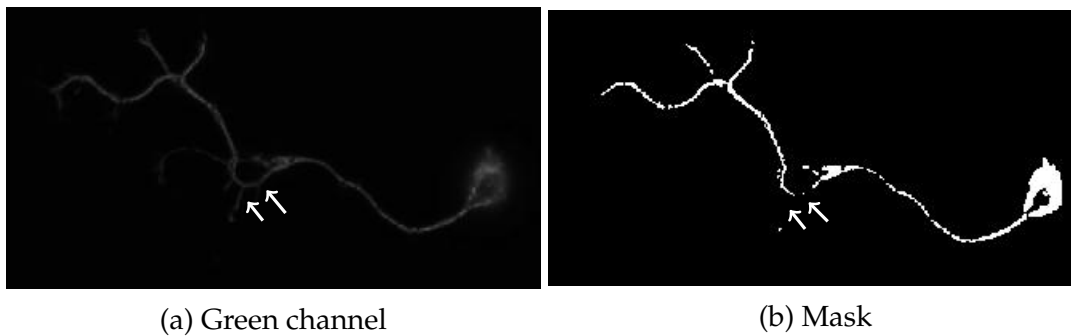


Figure 3.5: Applying a simple threshold mask to a channel. (a) Green channel of an original microscopic image of a *Drosophila* primary neuron. (b) Mask applied at a threshold of 30 to the green channel without any enhancements. Arrows denote parts of the axon where there is a gap in the mask, but visible lines exist in the original image.

In order to tackle the range of problems, a multi-filter approach needs to be considered, as contrast and resolution are different characteristics of the images. It is necessary to have a pipeline - a sequence of filters and functions applied to the initial image to provide a single outcome. In this case, the outcome is a binary mask obtained from a monoscale image. Since this processing of the images is an important step, it has been studied in-depth [70].

Microtubules in a cell can be understood as networks similar to capillaries in the eye in the context of retinopathies (Figure 3.6a), branch detection of roses for a harvesting robot (Figure 3.6b) to the study of spatial trajectory data in a city (Figure 3.6c). The first and second example are particularly striking as they obtain a binary mask and consequently a skeleton albeit with a very different starting image. All of the initial images are quite different from each other, however one type resembles the microtubule networks. Here for the first time, I use these network recognition approaches to recognise microtubules.

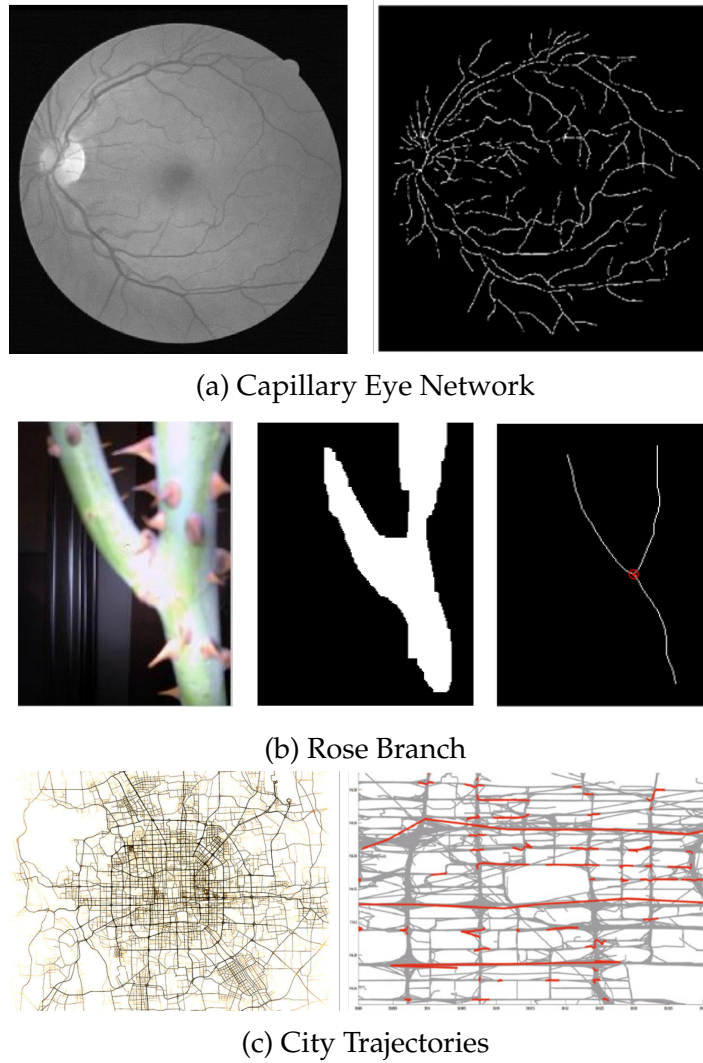


Figure 3.6: Examples of image processing into binary images, in different scales and fields. (a) Grayscale image of the inner lining of an eye, and the detected lines (adapted from [71]). (b) Rose branch detection with a binary mask, and subsequent skeletonisation (adapted from [72]). (c) City trajectories on a map of Beijing, with a processed close up (adapted from [73]).

The analysis of blood vessel patterns in the eye has been subject to extensive studies on how to improve these images for analysis [71]. As the main focus of the work in this thesis relies on the quantitative analysis of images, and consequent retrieval of parameters, the methods chosen for the processing were among the state-of-the-art. Furthermore, as the project pipeline was in MATLAB, searching for functions written in this language was pivotal for the integration into the existing pipeline. The most fitting programme found was *vesselness2D* ([74], see section 2.1.2), a 2D implementation of an improvement from the widely used Frangi filter for 3D images.

The basis of this function is calculating local tubularity, or the local probability of a certain point belonging to a tube. It calculates the eigenvalues² of a local matrix of second-order partial derivatives, Hessian matrix.

²Eigenvalues and their corresponding eigenvectors are the result of a decomposition of a matrix, showing information about their functional properties. In this case, it's the spatial properties of the points, i.e., their referential. Knowing these values enables the stretch of space in desired directions.[75]

The common procedure is to consider the convolution of an image with a Gaussian function to be taking its first derivative, i.e., $\partial(f * G) = f * \partial G$. This is done by transforming the image into the Gaussian scale space by convolving with a Gaussian function:

$$G(x, y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}} = \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \right) \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{y^2}{2\sigma^2}} \right) = G(x)G(y), \quad (3.2)$$

where σ is the standard deviation. This function is continuous but images are represented by discrete grids. As such, when sampling the function onto a grid, the resulting matrix has the maximum value in the centre pixel, with symmetrical decrease in values to the borders:

$$\frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \quad (3.3)$$

As it has symmetry in both directions, one of the most important properties of a Gaussian function is the separability of dimensions, as shown in Equation (3.2). On a discrete level and with only one dimension needed for the calculation, the neighbours considered are along each axis, reducing the number of neighbours per calculation to one vector.

The Hessian is further calculated by convolution of the image at a scale σ for the partial derivative of G for each dimension. From the resulting matrix, the eigenvalues λ_i are calculated and ordered by magnitude, where $|\lambda_i| \leq |\lambda_{i+1}|$; $i = 1, \dots, D - 1$ with D the image dimension. The sign and amplitude of these eigenvalues provides the local structure of a pixel, independently of their orientation.

Choosing the correct eigenvalues then is a matter of choosing how they relate. An analysis of structures based on their Hessian values has been performed [76], and for 3D tubular structures, the first eigenvalue would be close to zero, and the next two would have a high positive value. However, given the 2D microtubule images, there would only be one λ given that there exist two dimensions in these images. Microtubules themselves are 3D tubes and the approximation $\lambda_2 \approx \lambda_3 \wedge |\lambda_{2,3}| \gg 0 \wedge \lambda_1 \approx 0$ works for both 2D and 3D images of 3D tubular structures (as seen in [77]). The vesselness for a 2D tube can be described, therefore, with only two eigenvalues: λ_2 and λ_3

The enhanced filter function ν_P proposed by Jerman et al. is

$$\nu_P = \begin{cases} 0 & \text{if } \lambda_2 > 0 \wedge \lambda_3 > 0, \\ \lambda_2^2 (\lambda_\rho - \lambda_2) \left(\frac{3}{\lambda_2 + \lambda_\rho} \right)^3 & \text{if } \lambda_2 \leq \lambda_\rho/2, \\ 1 & \text{otherwise,} \end{cases} \quad (3.4)$$

where λ_2, λ_3 are the Hessian eigenvalues, and λ_ρ a regularisation of λ_3 by

$$\lambda_\rho = \begin{cases} \lambda_3 & \text{if } \lambda_3 < \tau \min_x \lambda_3(\mathbf{x}), \\ \tau \min_x \lambda_3(\mathbf{x}, s) & \text{otherwise,} \end{cases} \quad (3.5)$$

where τ is a cutoff threshold between zero and one, \mathbf{x} is the coordinate vector. The choice of τ will influence the difference between the magnitudes of λ_2, λ_3 for structures with low contrast. For each pixel, the value of the function is calculated, and in the end, the image will be normalised into values between 0 and 1.

While initially the value for τ is predefined as 1, the user has the freedom to choose the most appropriate for each individual image: lower quality images might need a different value to enhance the tubularity of microtubules present in the image.

Even though it seems the difference between Figure 3.7a and Figure 3.7b is not substantial, the tubular parts (microtubule network) got enhanced while others seem to have lost information (nucleus). This confirms the choice of eigenvalues based on the distinction of tubular structures in the images. Now, upon applying the same threshold as Figure 3.5, the difference in consistency of the axon is apparent (Figure 3.8).

This, however, poses the question on disorganisation, as they are generally parts of the image which, similar to the nucleus, are denser and have a higher number of lines, albeit tubular. After applying it to images presenting higher levels of disorganisation, as highlighted in Figure 3.9, it shows a comprehensive level of lines, even in higher density.

As such, due to its effectiveness in maintaining and recognising the tubularity of the microtubule bundles, vesselness is the function chosen in the pipeline to initially process the images, allowing a better grasp and understanding of the microtubule networks. However, the name itself did not make sense for the users in a context for microtubule networks. Therefore, the nomenclature used is virtualisation, as it the process is similar to the digitalising of the image into a computer-comprehensible matrix.

Not all image artefacts are addressed: by definition of the biological images, there might be parts of the networks that are not marked with the fluorescence protein. It will need, therefore, extra steps after the extraction of the skeleton. Furthermore, even with binary masks, there are some quantitative values that cannot be obtained, such as geometrical or topological characteristics as branching or curvature.

3.2.2 Obtaining binary shapes - Skeletonisation

A skeleton is the representation of the original image as the composition of one-pixel-wide lines. This is important because it allows a topological description of the image, and a comparison less dependent on resolution as it does not rely on either intensity or even thickness. The aim of this section is to describe how the binary masks obtained in the previous section can be translated into skeletons, a process referred to as **skeletonisation** of the image.

A good skeleton has to meet a number of conditions. First, it is important that the one-pixel-wide line is equidistant from the boundaries of the shape it represents, i.e. forming the centreline of the shape. This is an important step, in terms of reproducibility, as the centreline will probably be the same regardless of the thickness of the shapes.

Second, it is important to preserve the Euler number of the image. The Euler number is a topological invariant: for example, a straight line has the Euler number of 1 (one object, zero holes), while a circle has the Euler number of 0, and these values stay the same regardless of any way the image might morphologically transformed (for example, bent or stretched). If the skeleton maintains the Euler number of the mask it represents, this would mean that the skeletonisation did not produce or erase any major features.

To implement skeletonisation, it is important to decide which type of neighbourhood will be considered: either 4-connectivity (only considering the horizontal and vertical adjacent pixels, north east lines in Figure 3.10) or 8-connectivity (considering all adjacent pixels, north east lines in Figure 3.10). This will affect how the skeleton is built, as one-pixel-wide has different definitions in the two types of skeletonisation. With 4-connectivity, diagonal neighbours are not considered, which could lead to a loss of branches. On the other hand, 8-connectivity might produce artefacts, and would potentially be slower as it considers double the pixels per calculation.

For this work, two skeletonisation functions were studied: the *bwmorph* (8-connectivity, released in the version R2006a [78]) and *bwskel* functions (4-connectivity, released in version R2018a [79]) provided through MATLAB, as well as a processing function published previously [80].

The two MATLAB functions accept binary images as input, and use the same "medial axis transform" algorithm to calculate the skeleton. It is based on the principle that the medial axis of an object is the set of points on which if a circle with a centre in the point, and a radius tangent to the boundary will be a tangent in other points of the object (Figure 3.11), without crossing the boundary.

However, *bwskel* and *bwmorph* differ in other aspects as will be explained in the following:

The *bwskel* function uses 4-connectivity to calculate the medial axis. This function is often used for the deletion of branches in the skeleton, i.e., the removal of parts

that have a smaller number of pixels (the way a branch is smaller than the trunk of the tree). In larger scale images, these sometimes are artefacts of the process, rather than features of the original image. The minimum length needed for any line in the skeleton to be considered a branch and, consequently, not erased can be provided as an input to the function. Even though the skeleton calculation is done with 4-connectivity, the branch length calculation is done with 8-connectivity.

The second MATLAB function, *bwmorph*, is used for a variety of morphological transformations, one of which is the skeletonisation, calculated with 8-connectivity. In this function, the number of iterations of the algorithm is one of the inputs, which means that one can choose to iterate until the image does not change from one iteration to the next. As skeletonisation is only an option, the function can also be applied to the skeleton itself to retrieve other values and features, such as branch points (skeleton parts where a branch is connected).

There are several user-built functions in the community. For a more extensive test and comparison, one such function was chosen as a reduction step before skeletonisation, and further reduction to one pixel wide is done with *bwmorph*.

The skeletonisation function by Howe [80] is MATLAB implemented from a C++ function, based off on the method by Telea [81]. It implements a simple Augmented Fast Marching Method, before the last reduction to skeleton. After arbitrarily choosing a point in the boundary to start, each pixel on the outline of is attributed a unique, consecutive number, counter-clockwise until it completes the perimeter (Figure 3.12a). The numbered pixels are then considered background for the next iteration, i.e., the boundary "marches" forward one pixel (Figure 3.12b). The uncounted pixels will be numbered according to the closest outer number.

After iterating until all the pixels in the object have been numbered, the centre will present several pixels with a big discrepancy (i.e., difference) to their neighbours, as highlighted in Figure 3.12c. This discrepancy is important because, apart from the difference between initial and final pixels, it represents the interface of numbers originated from distant parts of the perimeter.

Reduction to the skeleton can then be performed. Even though "fast" is part of the name, this is a slow process that allows an independent comparison of the MATLAB methods with and without further pre-processing.

These processing steps will be part of a software pipeline, so performance is an important factor. The running time measurements were done by executing the MATLAB *tictoc* routine, which records the time points (toc) after the starter point tic.

To access whether the functions produce a reliable skeleton by maintaining the Euler characteristic, this was calculated in both binary images using the function *bweuler* [82].

Testing

For the first comparison of methods, the image used is the one suggested in the MATLAB documentation for *bwskel* (Figure 3.13a). In order to reproduce the results, MATLAB provides an adapted function that calculates the binary mask threshold with basis on the image itself, *imbinarize* [83], as it provides the optimal binary mask for these images. The resulting mask is shown in Figure 3.13b, on which all three methods described above were applied. Each method was timed with the MATLAB *tictoc* routine.

The default functionalities were used. For *bwskel* (Figure 3.13c), no minimum branch value was provided, so all were considered in the final skeleton. For both applications of *bwmorph* (Figure 3.13e,f), the number of iterations was declared as Infinite, i.e., function was applied to the image until the result did not change from one iteration to the next.

The Howe distances (Figure 3.13d), where the colour represents the pixels with the higher discrepancies to their neighbours, provide a non-binary mask. To be able to apply *bwmorph*, it is transformed into binary by choosing as one all the values above 35, i.e., with a discrepancy higher than 35.

The three resulting skeletons are objectively different, as evidenced by the zoomed in section. At a first glance, it is possible to see that the most artefacts are with *bwmorph*, as it creates branches at the extremities. On the other hand, *bwskel* has lost information. Without further information, the pre-processing with the Howe function appears to provide the best skeleton from the available information.

Regarding the time performance, as there were slight variances to the time each function was taking, the statistical approach was to run the code 1000 times and see the distribution of such values (Figure 3.14).

Even though all of these values are in the order of seconds, which appears to be a non noticeable difference, when taking into account that the whole analysis process is of hundreds if not thousands of images, then the difference becomes large. For example, as shown in Table 3.1, the Howe method has a total time for all the iterations of 3042 seconds, or roughly 51 minutes, having into account that it is followed by *bwmorph*.

Table 3.1: Summary of skeletonisation time performance in seconds for each of the functions presented in Figure 3.14.

	<i>bwskel</i>	<i>bwmorph</i>	Howe	<i>bwmorph</i> with Howe
Minimum (s)	0.2	0.07	2.8	0.003
Median (s)	0.3	0.1	3	0.004
Maximum (s)	1.4	0.6	6.2	0.06
Total time (s)	268.3	104.4	3042.1	4.7

The minimum time taken by the Howe pre-processing is already longer than either maximum of the other two functions, which needs to further add the skeletonisation time. The fastest method is *bwmorph*, by at least one order of magnitude, which results in less than half the time of *bwskel*.

The next testing step is the performance of the methods on microtubule network images already used in the previous section, post-vesselness (Figures 3.7c and 3.9). For this, the images are converted into binary masks (with the threshold of 30), and consequently skeletonised. The results are presented on Figure 3.15.

From the start, the function after Howe processing can be discarded as it does not maintain the Euler characteristic of the mask. On the other hand, even though the two MATLAB functions produce different skeletons, both maintain the characteristic.

In these samples, *bwskel* has the most faithful representation of the mask, while *bwmorph* creates several small artefacts (branches). However, whenever there is a gap of very few pixels on the mask, the skeleton produced by *bwmorph* bridges between lines, giving a more cohesive final result. This can be explained with the type of connectivity used: as *bwmorph* considers diagonals, it is more permissive when two objects are in close proximity.

Conclusions

In the beginning of this section, it was mentioned which characteristics were needed for an appropriate skeletonisation function: conservation of Euler number, time performance and preservation of the microtubule disorganisation. Particularly when taken with fluorescence microscopes, these biological images tend to be blurry around the disorganisation. Even with the vesselness filters, which captures most of the perceived lines, the variation in thickness of the tubes can influence the skeletonisation: thinner lines might not always be considered relevant for the contour.

Even though Howe function provides a skeleton with much less branches, it takes ten times longer to perform, and loses a lot of information when looking at thin structures, such as the axon. Furthermore, the Euler characteristic of the mask is not maintained.

On the test image, perhaps the *bwskel* would appear more suitable as it provides the cleanest skeleton when comparing to the mask. However, even without a minimum branch length determined, it accentuates the gaps in the original image.

Finally, *bwmorph* is not only the fastest, but is also the most conservative as it establishes connections between parts of the mask that are not picked up by the other function. It does create artefacts in the final skeletons, but these can be dealt with by using other of its functionalities. In the final pipeline, these artefacts can be dealt by splitting the image per branch points (places in the skeleton with more than two neighbours) and eliminating the ones with very little amount of pixels.

The disorganisation images present a number of artefacts that get propagate

through the pipeline. If part of the image has low intensity, it can be lost even when using enhancement filters such as vesselness. However, some gaps still persist, so if one of the skeletonisation functions increases the space in these gaps, it would only further replicate the artefact. As such, the function chosen for this step is *bwmorph*.

3.3 Identify paths within the shape patterns

After extracting the skeleton, the following step is its analysis. An important feature of the neurons that is affected in many mutants underlying neurodevelopmental or neurodegenerative diseases is the length of the axon. In some phenotypes, the axon is shorter. As such, retrieving and identifying the axon within the skeleton is pivotal. To do so, a path is established between the cell body and the axon terminal.

3.3.1 Existing Software and Methods

The analysis of paths has been implemented in different ways in a number of softwares for image analysis. It does fall short to what is needed for the purposes of this project, and the implementation into the pipeline would be morose.

First, in case of Cell Profiler, the software is analysing volumetric shapes in the images, i.e., cells or organelles that are well defined and within a region. As such, it does not have a good pipeline to analyse fibrous structures, such as microtubules. The software does not have implemented a path calculation algorithm.

Both NeuronJ (the enhancement of ImageJ for neurons) and Fiberapp, on the other hand, are specifically for fibrous structures.

NeuronJ does present two options for path finding: either automated or click-based. The first one is done by intensity search. From each pixel, it chooses the path according to the neighbour with the closes intensity value. However, it is not only slow, but ambiguous and performance heavy. Furthermore, it still needs the user to check whether the path is correct in each individual image.

The other option is the one also present in Fiberapp: a click-base approach that requires the user to trace the path with clicks. The path corresponds to the straight lines between each of these points, so the nuances of the axon (such as small curvature changes) are lost.

The aim of this section is to make path determination quick and integrated with the pipeline, while maintaining minimum user input.

3.3.2 Implementation

Traditional path recognition in images is slow: it consists of the searching of neighbours in 8-connectivity. Even though it can be done with a normal image, by moving on to the next pixel based on maximum intensity, it is computationally heavy with

scaling. If each pixel has to search its neighbours, even by discarding the pixels already visited, it leaves a computation of up to 7 neighbours. In a path of 1000 pixels, that would be 7000 searches.

This method could be simplified with the introduction of a skeleton, as it is not about the intensity but rather the existence of 1-numbered pixels in the image. However, how could the path discern between two neighbouring pixels in a branch? The simplest solution would go by the path search from both ends of the path, and finding a common pixel. This, however, is still computationally heavy.

Mathematically, there is a type of structure to quickly establish a connection of neighbours: a graph. By considering each pixel as a node of the graph, connected to its spacial neighbours by edges, the skeleton becomes a network of points where a path between two is easily identified. An example can be seen in Figure 3.16. What is stored is the node and its neighbours, so the connection search in the skeleton is far quicker as each node would have a maximum of 4 connections.

The transformation from binary images into graphs is already available on the MATLAB community, in the package *BinaryImageToGraph* ([84] see Section 2.1.2). This function converts each pixel valued 1 into a node, and the edges are the connections to every neighbour, considering 8-connectivity. Even though edges might have different weights in a normal graph, in this case they are homogeneous, as proximity is the same between all nodes.

For example purposes, the transformation of Figure 3.17(a) to graph is a simplified version, only considering the line segment changes as nodes. Numbering was arbitrary.

The only pivotal input of the user is at this stage, and only to determine where are the endpoints of the path. In Figure 3.17b, the path should be found between nodes 1 and 4.

As all edges have the same weight, the method used to find the shortest path in a graph was Dijkstra's algorithm, or Shortest Path First (SPF):

- Starting at the source point (first user input, node 4 in Figure 3.17b), all the other points are marked as at an infinite distance.
- The algorithm will move to all its non-visited neighbours and register the smallest distance from the source to that point.
 - As only one neighbour is available for both node 4 and 3, so it is always the same step (Figure 3.17 c and d).
- Nodes are marked as visited and do not account for any future neighbouring calculations and the algorithm moves to the next node.
- Upon reaching node 8, there are now two neighbours, and so these would potentially originate a different path. The difference relies on node 6. It calculates

two possible distances to the source:

- 3 nodes (Figure 3.17c) and 4 nodes (Figure 3.17d). At this point, it will register the shortest difference.
- The algorithm then expands until the target point (second user input, node 1) is found.

In the biological images with disorganisation patterns, the shortest path will generally go through the disorganised region. In Figure 3.18, the path finding algorithm was applied to the *bwmorph* skeletons in the section above.

Sometimes the shortest path is not the correct one. For example, if there are neurites in the image: these tend to have a much lower intensity than the axon, which makes it perceivable in the channel image. However, intensity is not a factor in the graph. As such, the first addition to the algorithm was the possibility of adding extra points that should be included in the path. This way, the graph will produce the shortest possible path including all three (or more) nodes.

Using the practical example of Figure 3.19, it is discernible in the channel image (Figure 3.19a) where the axon is, as per intensity. Once it is skeletonised in Figure 3.19b, it appears like a normal path. As such, after marking the initial and end points of an axon (in yellow), the shortest path will invariably be through the cell body and neurite (Figure 3.19c). By adding the third point (in green, Figure 3.19d), the path is corrected.

The order of the points chosen is irrelevant. The algorithm is built in a way that, starting from the first input, it will choose the shortest path to any of the other points. The algorithm moves on to the next node and measures the shortest path to all the remaining points that have not been visited. This way, no point is visited twice, the choice is always the shortest path between the nodes and the minimum input is two.

The algorithm works seamlessly for graphs that are fully connected, i.e., there is at least one possible path between any two nodes. As mentioned in the previous sections, the skeletons are not always complete, and can present artefacts such as gaps. In the existing softwares, either the gap would be clicked over or would stop the algorithm without reaching the end of the axon.

The algorithm to search for the shortest path in this project accounts for these gaps. If initially no connected path can be found between the two initial points (Figure 3.20), there are at least two groups of connected components, one to each input node (in this case, nodes 4 and 1). The program will ask for a third point, on the extremity (node with just one neighbour) of one of the groups. From that third input, the algorithm scans within a certain radius (that can be tuned by the user) for the closest point connected to the other group.

In practical terms: the user can either choose the end of the first group (node 8, Figure 3.20a) or the beginning of the next (node 7, Figure 3.20b). With the first

option, the algorithm will try to find the point connected to node 1 that is the closest to node 8. In this case, it would be node 5. On the other hand, by choosing node 7 and searching for the closest connected component of node 4, it would establish the connection with node 8.

When the algorithm finds the node at the closest distance, it creates a straight line (shortest distance of two points) between them, and establishes the path. In this particular case, the choice of the extremity would influence the final path. In Figure 3.20a, nodes 6 and 7 would not be part of the path.

The user can add as many points as necessary to establish the path. There is also the chance of directly adding a line to the path in cases where there is a part of the axon with a lot of gaps. Furthermore, the algorithm can always revert to the path before adding any points, even the initial ones.

The path is saved as the matrix coordinates of the points. It can easily be transformed into a binary matrix by defining those points as 1 and everything else as background.

Conclusion

The tested software did not provide an appropriate way of quickly and independently obtaining the axon of the neuronal images. Given that so much of the pre-processing was already done in a pipeline, it was important to find a method to find the path in a timely manner.

With graphs, the user only needs two points to start searching for a path, and it will work for the greater percentage of images. Besides, the method is built in a way that allows the user to add or remove points at any time. As there are two strategies to overcome gaps, it allows the length calculation in situations where the quality of the image is inferior: one point at a time, or by adding complete lines to overcome a part of the image. This method allows the user to obtain the path with a minimum amount of clicks.

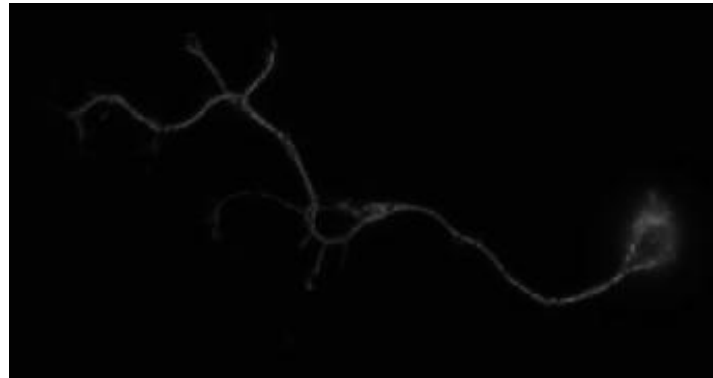
It is not completely user independent, and automation could perhaps be obtained with Machine Learning techniques. For example, training a network to classify parts of the cell as axon, cell body, disorganisation. However, this is not completely feasible with the number of images available (this will be further discussed in Chapter 7).

There could be potential ways to enhance the path finding technique. As of the current version, when creating the graph, each edge is given the same weight between neighbours. One potential way to circumvent examples as Figure 3.19 could be by weighing the edges with the difference in intensity between pixels. Pixels with similar intensity would have a stronger chance of being a path than otherwise.

Regarding the gap finding, it still needs an extra input of the user and, ideally, the program would be able to find the path independently. However, as of this project,

there would not be an appropriate way to determine what is path, and what is background. Instead of the risk of having parts of the background being considered as parts of the path, which would influence the calculations, it is better to give the control instead to the user.

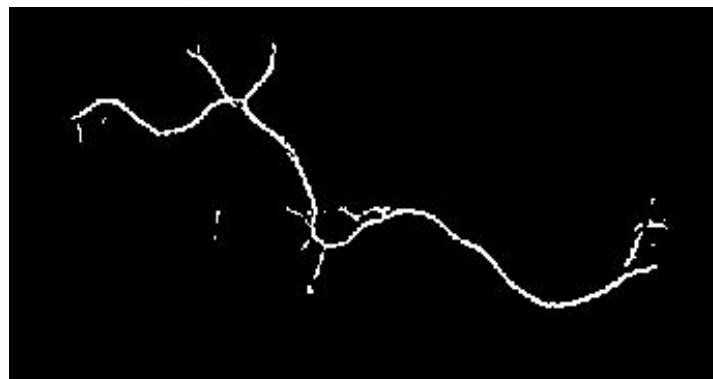
After all the pre-processing, the calculations can now be performed.



(a) Green channel



(b) Enhanced with Vesselness



(c) Mask

Figure 3.7: Applying vesselness and a subsequent threshold mask to the channel. (a) The original image (see Figure 3.5a); (b) the same image after applying the vesselness function with $\sigma = 3$ and $\tau = 1$; (c) subsequent mask with the threshold of 30.

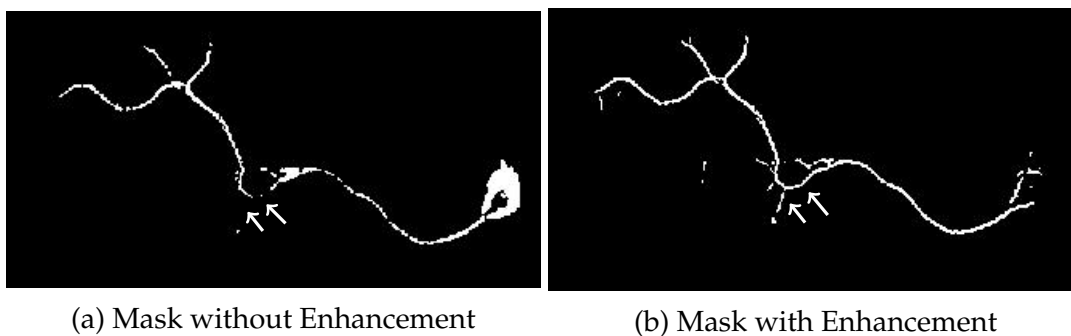


Figure 3.8: Comparison of masks with and without filtering the images by applying vesselness. Arrows denote parts of the axon where there is a gap in the mask without enhancement.

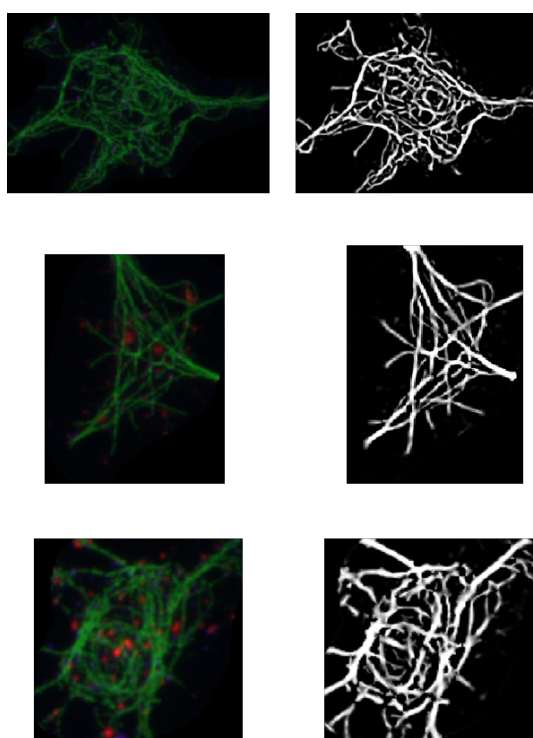


Figure 3.9: Vesselness applied to disorganisation sites of *Drosophila* primary neurons, with a mutation. RGB images on the left, the resulting monoscale image on the right. Images acquired by Dr. Yu Ting Liew. Shown areas of width: $13.2\mu m$, $9\mu m$, $10.5\mu m$ respectively.

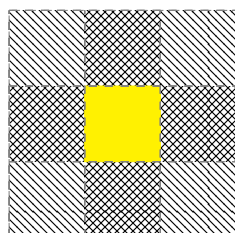


Figure 3.10: Schematics on the definition of neighbourhood. Highlighted is the focused pixel. North west lines represent 8-connectivity, north east lines represent 4-connectivity.

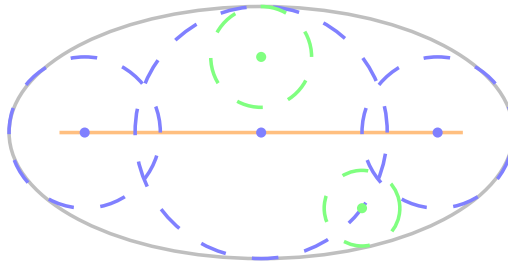


Figure 3.11: Medial Axis (orange) of an ellipse (gray). The blue dots are points in the medial axis, and the respective dashed circles are the distance to the boundary. In green are points that do not belong to the medial axis but have a circle tangent to the boundary.

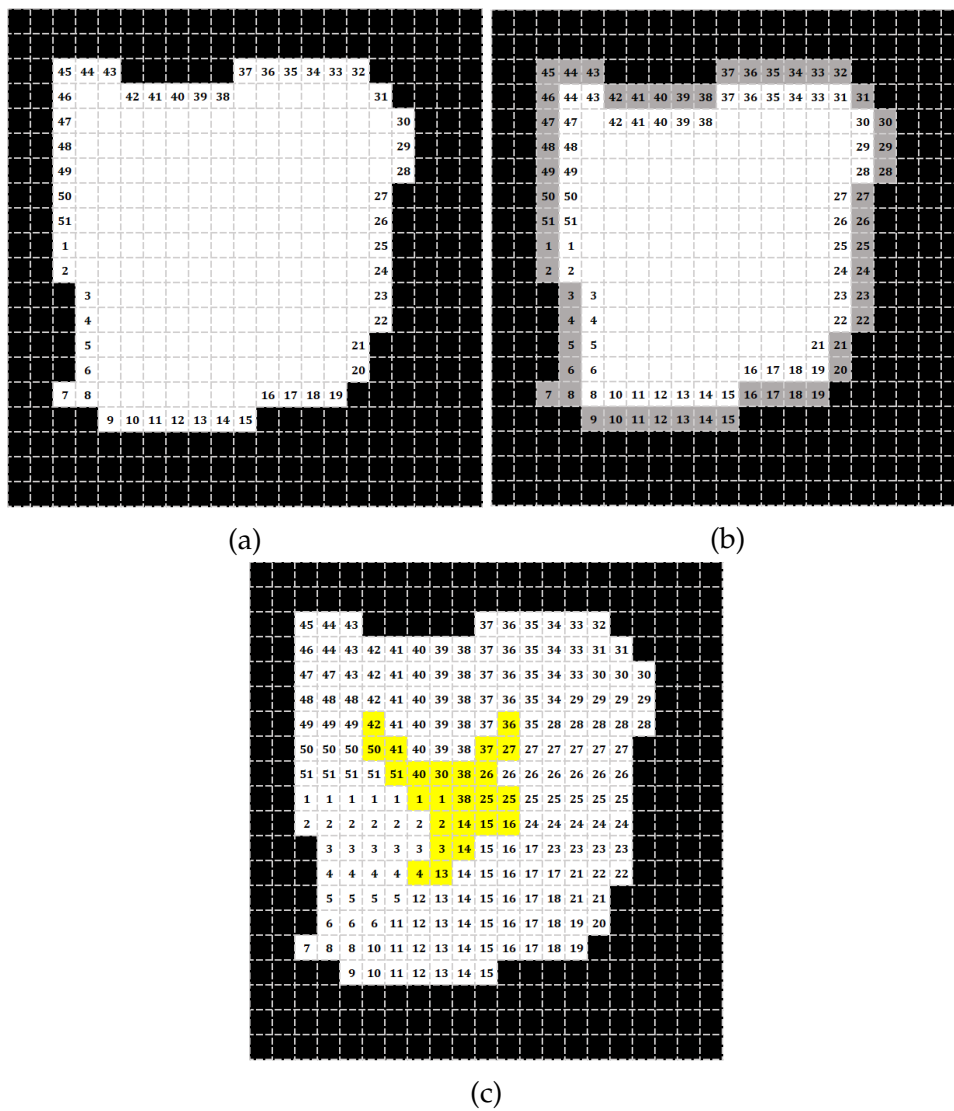


Figure 3.12: Fast Marching Method exemplified. (a) Initial boundary number assignment; (b) first iteration of the advancing boundary, where grey represents the already numbered pixels; (c) final iteration. Highlighted in yellow are the pixels that have a discrepancy to their neighbours above 8, and less than the maximum.

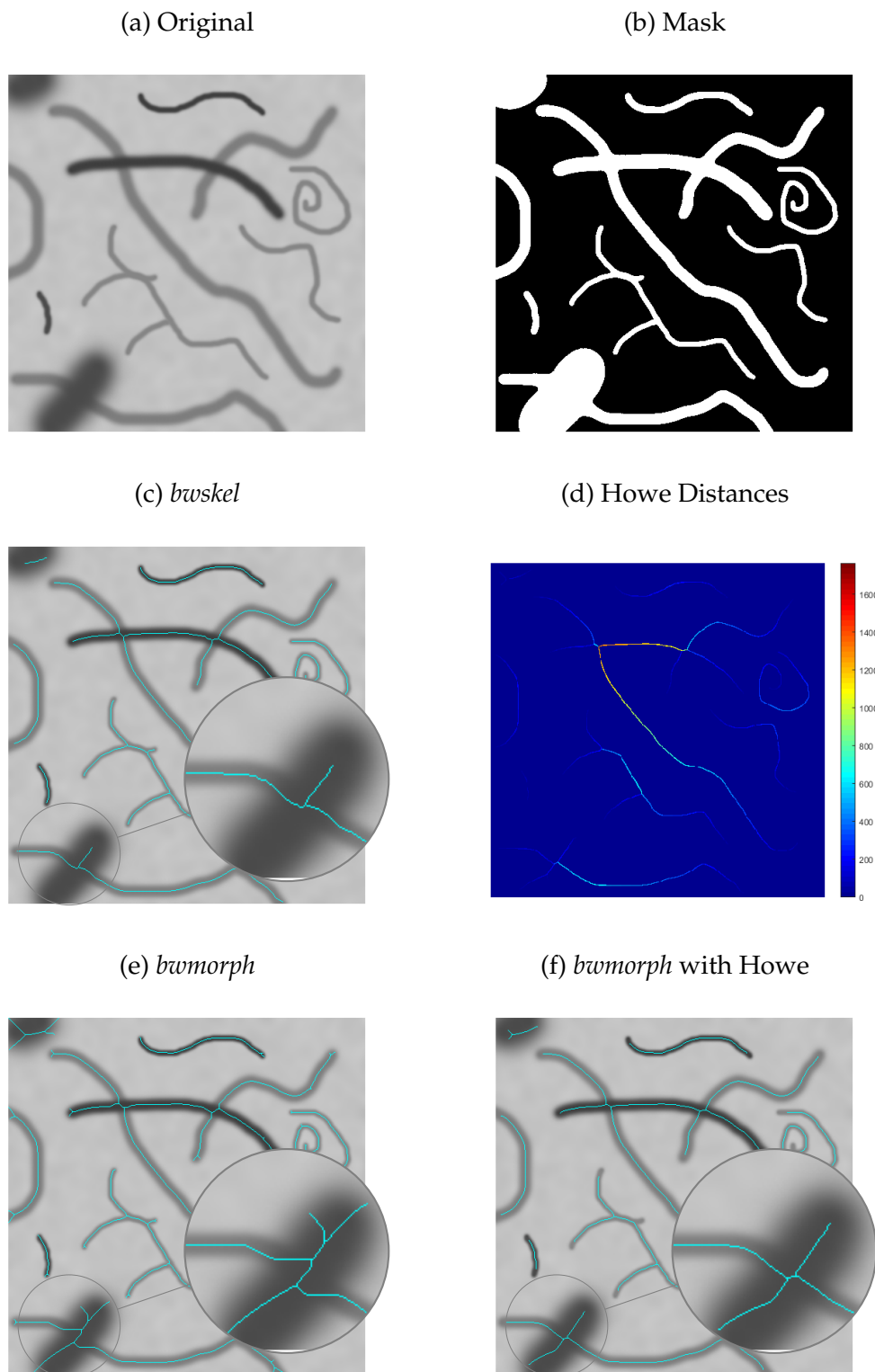


Figure 3.13: Skeletonisation of sample image "threads.png", available in MATLAB, using the different methods. (a) Original Image, (b) Mask obtained with *imbinarize* (c) Skeleton obtained with *bwskel*. (d) Distances calculated with Howe's function, colourbar represents the distances. The skeletons originated by *bwmorph* before (e) and after (f) cleaning with Howe distances. The skeletons are represented as the one pixel cyan lines, overlaid on the original image.

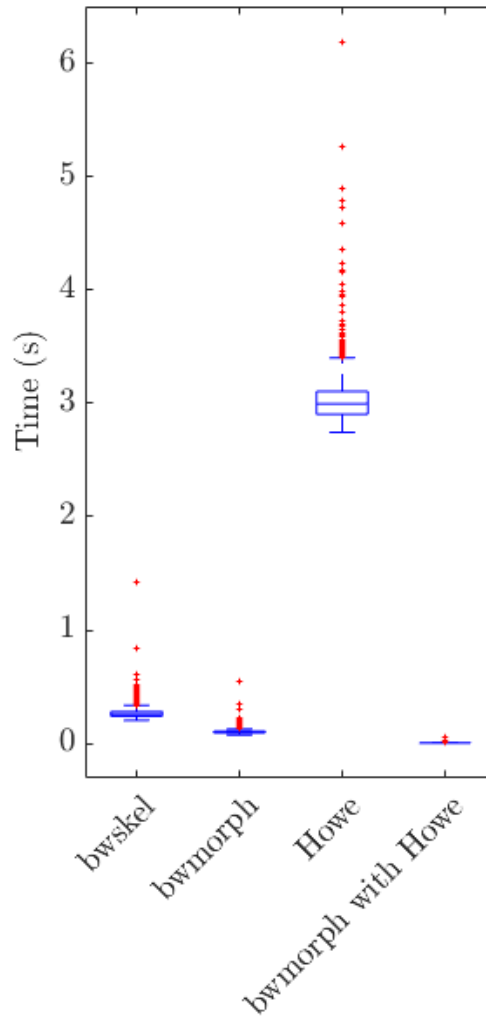


Figure 3.14: Boxplot of the time (in seconds) for each function call, for the original image in Figure 3.13. Each point represents the time each function took individually. The process involving *bwmorph* with Howe processing would take the added time of the last two columns. Blue represents the interquartile range, red the outliers. $N = 1000$.

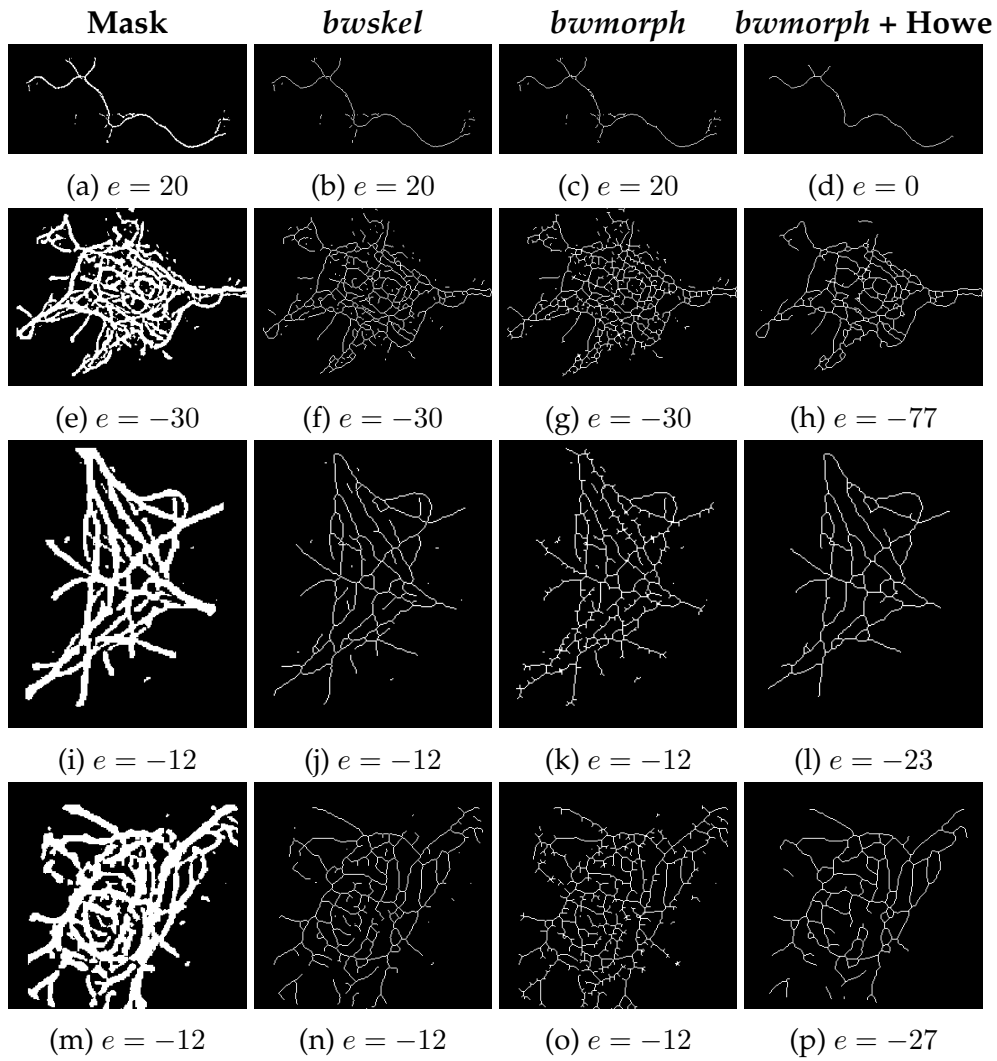


Figure 3.15: Skeletonisation of microtubule network images, presented in Figure 3.9. (a,e,i,m) masks calculated from the vesselness images in the previous sections; (b,f,j,n) skeletonisation of masks with *bwskel*, without any branch length; (c,g,k,o) skeletonisation of masks with *bwmorph*; (d,h,l,p) *bwmorph* was applied after Howe. Under is the respective Euler characteristic, calculated with *bweuler*.

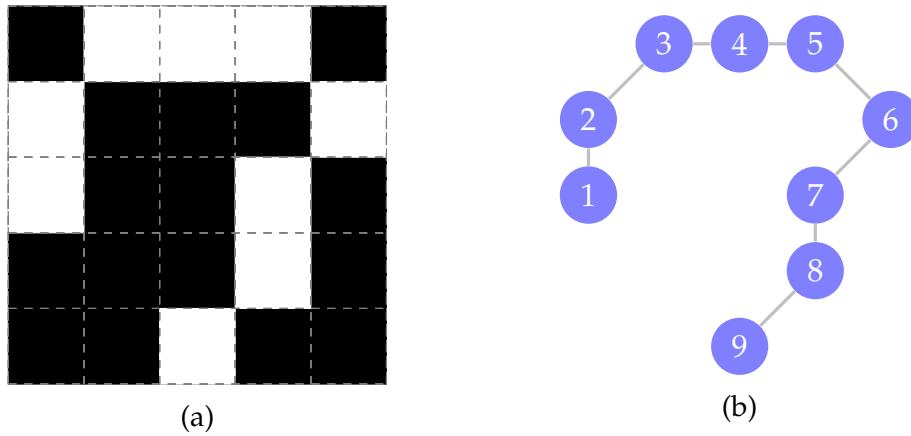


Figure 3.16: Transforming a binary image into a graph. Each white pixel on the image (a) correspond to a node in (b), and is connected by an edge to its adjacent pixels.

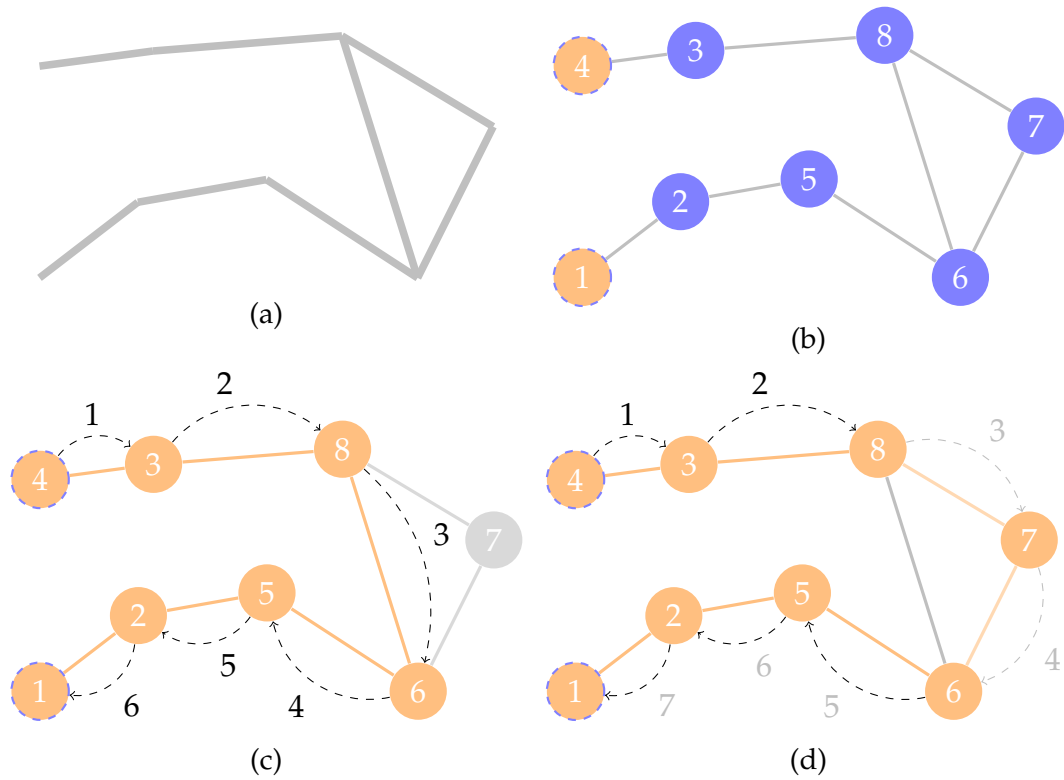
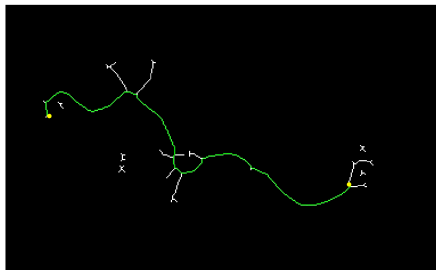
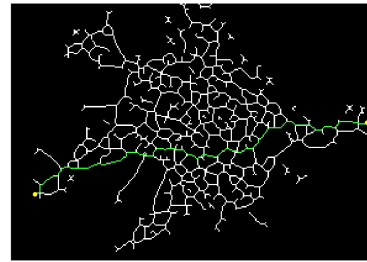


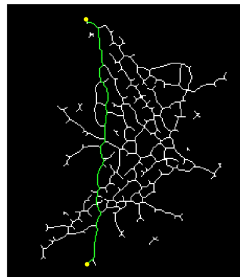
Figure 3.17: Schematics of an algorithm from skeleton to path. For simplicity, only the line changes in (a) correspond to pixels. Those points become nodes in (b), and each connection becomes an edge. The numbering is arbitrary. For the path finding, in (b) the user input is reduced to two clicks: the endpoints, nodes in orange surrounded by the dashed line (nodes 1 and 4). (c) and (d) are the two possible paths. The dashed arrows represent a step between two nodes, and the adjacent numbers represent the distance between the node at the end of the jump and the source, with black the shortest route and grey the longest.



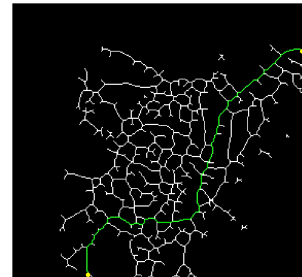
(a)



(b)



(c)



(d)

Figure 3.18: Path finding for the skeletons obtained in the previous section. Yellow dots represent end points, green represents the shortest path between them.

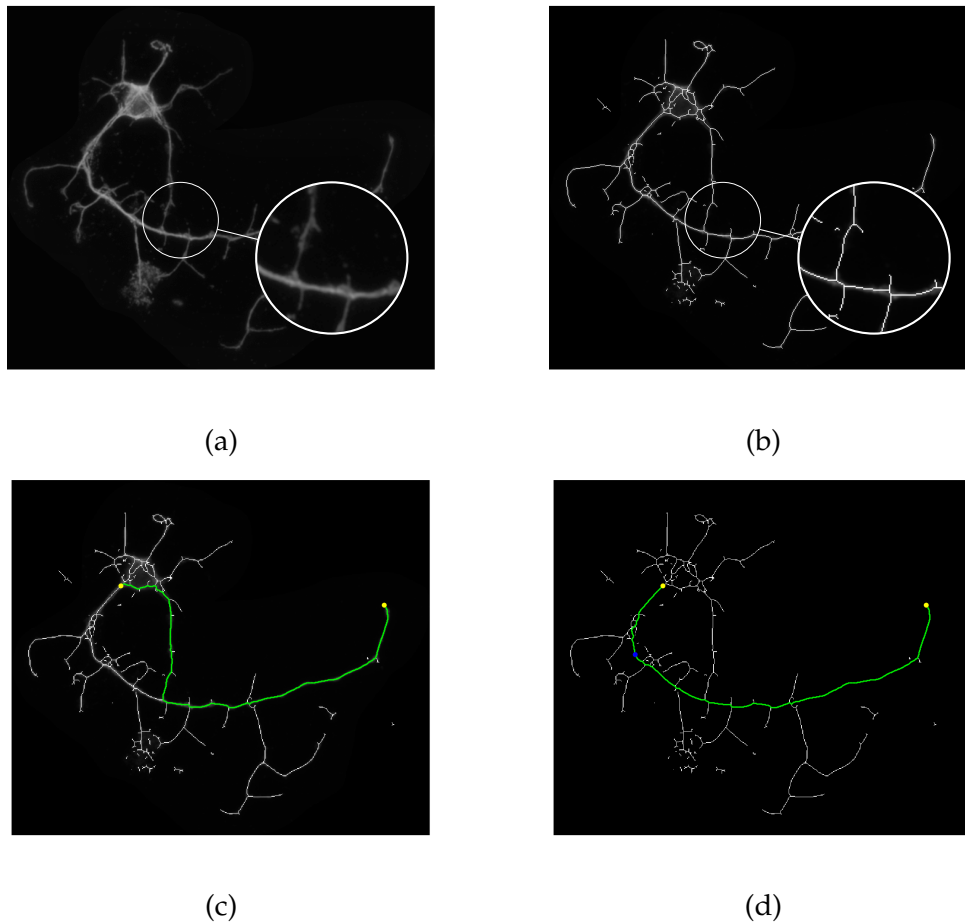


Figure 3.19: Example of the correction of the path by adding extra points. Channel image (a) with its skeleton (b), after vesselness and mask threshold of 10. Zoomed is a junction that is not connected in the intensity images, but is connected in the skeleton. (c) shows the shortest path in green, (d) path after adding the extra point on the axon. Yellow dots represent initial points, blue dot represents the correction. Shown area of width: $39.2\mu m$.

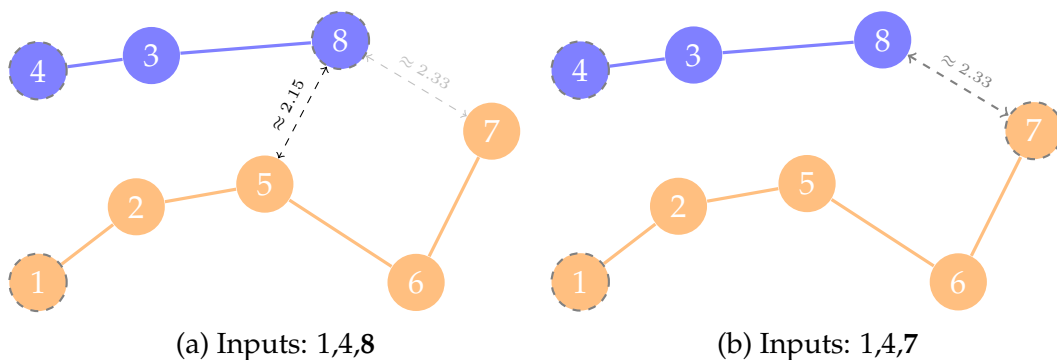


Figure 3.20: Finding paths in graphs with gaps. The initial inputs are the same in all three images: 1 and 4. Blue are the components connected to 4, orange the components connected to 1. Dashed circles denote the input nodes. Dashed edges are the distances between nodes, with dark grey the optimal, and light grey the longest. In (a), nodes 6 and 7 would not be a part of the final path.

Chapter 4

Quantitative analysis and description of the binary shape

4.1 Background and Rationale

After shape extraction, the quantitative analysis can be performed using a set of different parameters, including axon length, straightness, curvature and areas (Figure 4.1), as will be detailed below.



Figure 4.1: Regions of the skeleton with relevant shape information for extraction: axonal length (orange), straight segments (blue), microtubule disorganisation curvature and density (red).

- **Axonal Length** is defined as the distance between the cell body and the axon tip (orange in Figure 4.1). It is a function of age and health, with axons growing shorter or longer upon certain genetic or chemical manipulations. This parameter is also applied to calculate the microtubule disorganisation index where the area of microtubule disorganisation is assessed relative to axon length [28]. Furthermore, length changes of microtubules within axons may indicate potential aberrations, e.g., when microtubules become fragmented.
- **Straightness** is the ratio between the N longest straight elements (Figure 4.1, blue) of an image per axonal length. Straightness can help describe axonal morphology as a function of its original growth path (i.e., whether the extending axon grew along a direct path or performed turns) and/or the tension it is under. Straightness can also be used to describe microtubule organisation, e.g., informing whether microtubules are arranged into bundles or form areas of disorganisation.

- **Curvature** indicates how much a line deviates from being straight. Curvature provides a quantitative way of measuring and comparing looped microtubules in areas of disorganisation (Figure 4.1, red).
- **Areas** in the context of the image analysis of neurons are important to quantify axonal swellings and the amount of microtubules in them, as a means to calculate their density (area of microtubules per area of swelling).

Most parameters are straightforward characteristics of geometrical shapes, such as areas and number of segments. These are directly extracted from the skeleton, and no further calculation is needed (Section A.0.5). However, for the measurement of straightness and curvature there are no reliable methods or precise enough software solutions available, which therefore need extra consideration. My aim was to find methods to assess straightness and curvature to be used as parameters for the analysis of neuronal microtubule phenotypes.

4.2 Length and Straight Segments

4.2.1 Length

The retrieval of length from the image could be just defined as the difference between the beginning and end points of the object. While this is true for completely straight objects, axons are more intricate.

The length of the axon can be calculated from the first axonal point next to the cell body to the axon terminal. These calculations are based on the path recognition method used in each individual software. For example, in ImageJ, the user clicks along the object where different clicks are connected by straight lines, forming segments. The total length is the sum of the lengths of all segments.

In the pipeline describe in this thesis, the path recognition algorithm (Section 3.3) is applied between the endpoints of the axon. As the path is saved as a graph, the axonal length becomes the number of nodes between the points, which, given the used shortest path algorithm, provides the smallest possible length between the two ends of the axon.

4.2.2 Straight Segments

Of the various shapes that can be considered in image analysis, the simplest is a straight line. Generally, the existing software packages available for image analysis (see Section 1.4) do not measure straight segments explicitly: at most, they may provide some measurements of angles and directions between lines (angle α in Figure 4.2a). Furthermore, the concept of straightness is usually assessed as the ratio of the shortest possible path between two points (their Euclidean distance) and the actual

length of the path. However, this doesn't consider the shape of the actual path, for example, if it is composed of straight or curved segments (see Figure 4.2).



Figure 4.2: Classic definition of straightness considering Euclidean distance. Two examples of different paths (grey lines) between points A and B that have the same Euclidean distance (orange, dashed lines). The grey lines are fundamentally different: (a) is composed of straight segments while (b) is a curve.

In this thesis, straightness S is defined as the ratio between the sum of the length l_i of the N straightest segments of a selected path and the length L of the path, as in Equation (4.1).

$$S = \frac{\sum_i^N l_i}{L} \quad (4.1)$$

We found this to be a helpful measure to describe certain subtle differences in images, which can have important applications. For example, it provides us with an additional parameter to describe the shape of axons or of microtubules in areas of microtubule disorganisation regions (Figure 4.3): in the context of the axon, areas of straightness may indicate the degree of local tension or reflect the growth history of the axon. When observing microtubule disorganisation, it can indicate whether all microtubules are looped or whether there is a fraction that has kept its original straight conformation, thus quantifying the degree of disorganisation in a given area.

Hough Transform

A straight line can be described by only two parameters (m, b) as the relationship between the Cartesian coordinates (x, y) of every point on the line is $y = mx + b$. However, the calculation of the slope $m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$ becomes a problem in the case of a vertical line, where $\Delta x = 0$. This can be solved with a Hough Transform, by instead having the implicit representation of (x, y) with the Hesse normal form, thus describing a line in \mathbb{R}^2 as

$$\rho = x \cos \theta + y \sin \theta. \quad (4.2)$$

This technique represents straight lines in the parameter space of (ρ, θ) , where ρ is the normal distance from the origin to the line, and θ is the angle of the normal to

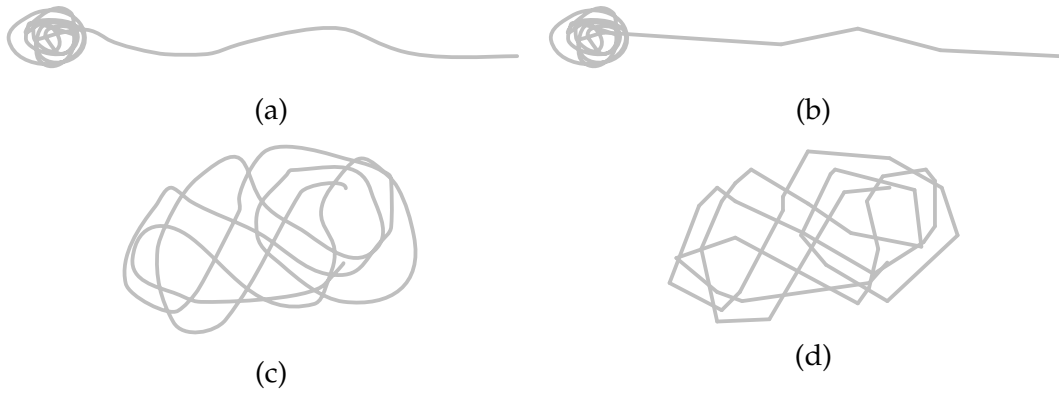


Figure 4.3: Applications of straightness values to synthetic images of interest. Examples of axons (a, b) or disorganised microtubules (c, d), showing only curvature (left) or only straight segments (right). Both images in the first row would score similarly with the Euclidean metric: 92.72% for (a) and 95.15% for (b).

the line to the x -axis (Figure 4.4).

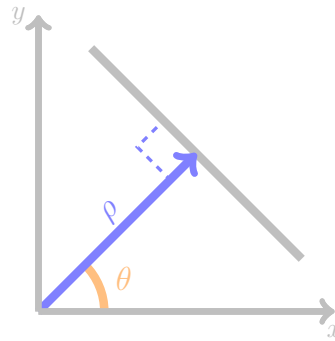


Figure 4.4: Definition of the parameters in the transform space. ρ represents the distance of the perpendicular of the line to the origin (blue), and θ the angle of the line to the x -axis (orange).

To understand how the Hough transform can be used to identify straight lines, let us have Figure 4.5a as the input image. First, select any point of the relevant pixels in the image (i.e., not the background). Here, we consider three on the vertical line (blue, orange and red dots in Figure 4.5b). Using their coordinates (x_i, y_i) , we then calculate the possible $\rho_i, i \in \{1, 2, 3\}$ from equation 4.2 by sampling a few values of θ , as illustrated for $\theta \in \{0^\circ, 45^\circ, 90^\circ\}$ (Figure 4.5, c-e). In simpler terms, we calculate the possible straight lines that go through each point i for certain values of θ .

Then, ρ_i is plotted against θ for the entire collection of straight lines considered per point (Figure 4.6a). Each point will produce one $\rho_i(\theta)$ line, for a discrete range of θ . If we collapse this onto a grid, the number of $\rho_i(\theta)$ per grid square (bin) can be counted. Points lying on the same line will intersect in the same (ρ, θ) grid square, increasing its count (as evidenced by $\rho_i(0^\circ)$), which then implies that the longest straight segments will be the grid points with the highest number of intersections. If this is done for all

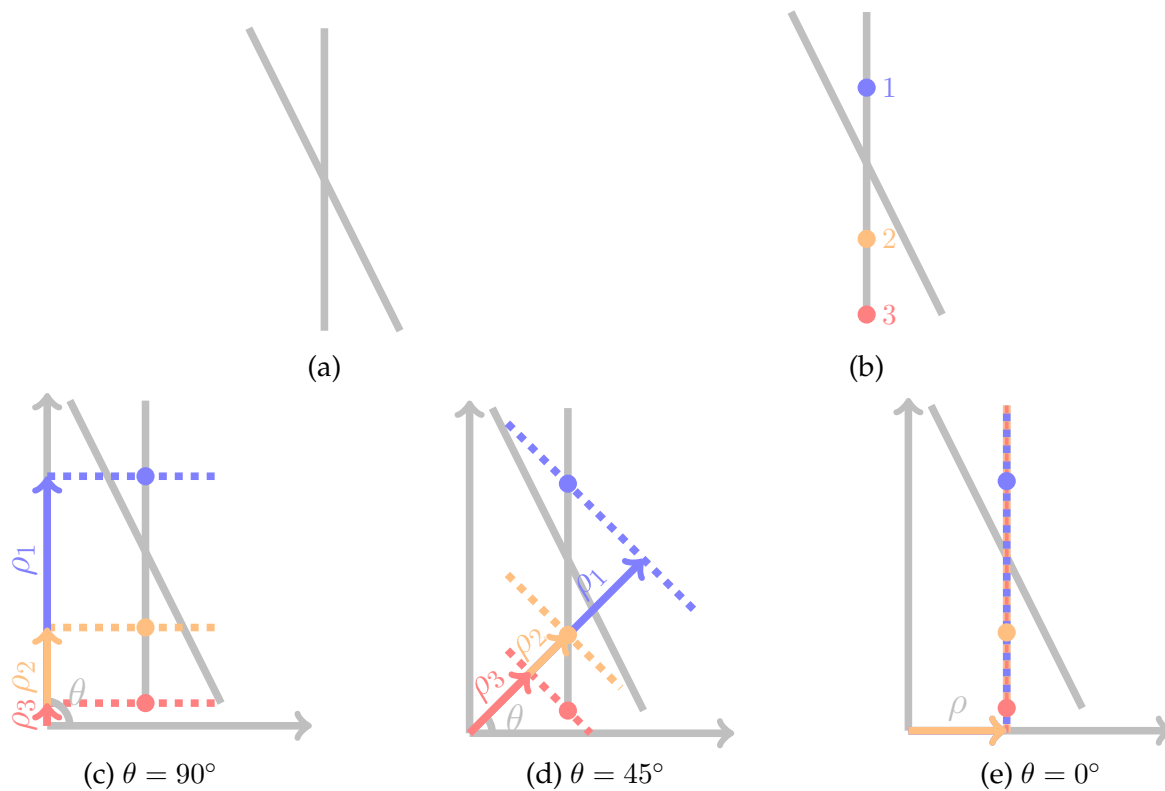


Figure 4.5: Example of how the Hough transform can be applied to identify two straight lines. (a) Input image of the two lines (grey), and we pick any point. (b) Three random points (1, 2, 3) from the same line were chosen. Plotting the possible straight lines (dashed) that include the three points selected for certain values of θ , in this case, for θ (c) 90° , (d) 45° and (e) 0° .

the points in the image that are not background while considering a large number of θ , a graph such as the one shown in Figure 4.6b will be generated.

In order to identify the lines in Cartesian coordinates, we can calculate the reverse transform and retrieving the possible (x, y) pairs that belong to the image from (ρ, θ) . Plotting them back on the image confirms that the correct straight lines were identified (Figure 4.6c).

In conclusion, the Hough Transform provides a fast means to identify straight lines in an image; to obtain the N straightest lines in any given image, one just needs to retrieve the (ρ, θ) pairs for the N bins with the highest count.

Limitations

The efficiency of the method is expected to be a function of the image quality, as a low quality (for example, due to low resolution of the images) will cause considerable noise around the bins with the highest count, given that the edges of the shape would not be sharp. However, applying the method to a binary skeleton reduces the background noise influence on the results. Regarding pixel robustness, if there's a significant alteration in the angle between several points, then it won't count as a long straight segment but rather a collection of smaller ones. The rotation of the

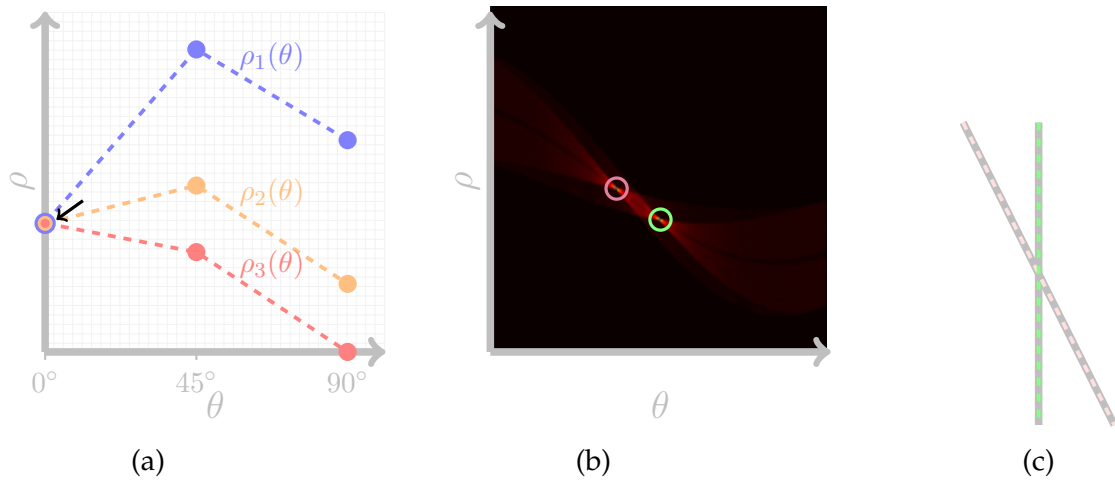


Figure 4.6: Plotting ρ and θ in a cumulative graph. (a) Plot for the example of Figure 4.5, each colour corresponding to the equivalent point; arrow denotes the overlap for $\theta = 0^\circ$, (b) Full calculated plot for the example image in Figure 4.5 (a), each circle denoting a high count bin. The θ -axis has been shifted to the right (so that the origin is not for $\theta = 0^\circ$) for clarity, as one of the high count points would have been on the ρ -axis, as evidenced in (a). (c) Plot of the detected straight lines (dashed) after converting back to Cartesian coordinates. Pink and green refer to the corresponding coloured circle in (b).

digital image can also cause different results: a diagonal will have a ladder of small straight segments rather than a long one. Nevertheless, small pixel fluctuations will not influence the overall results, as both the function used and the pipeline have leniency.

The transform described here only comprises two parameters. This can be amplified to any number of parameters in order to recognise other shapes, but this would increase the complexity of the transform. For example, with one more parameter, we could potentially use the Transform to find circles in the image, and therefore obtain the curvatures. As we do not know the target radius that we would be searching, this would be computationally expensive as it would have to search for all the possible circles, or an arbitrary number of those. Any other shape could be found by increasing the number of parameters but this, in turn, becomes increasingly prohibitive, as the more parameters present, the harder it would be to obtain a bin with a high count in the (ρ, θ) referential.

MATLAB Implementation and Software Incorporation

Having decided on the Hough Transform strategy, I chose to implement it using MATLAB, as it already provided some of the tools needed to quickly calculate ρ and θ (`hough` [85]), as well as their peaks in the transform space (the bins with the highest counts, obtained with `houghpeaks` [86]) and the consequent lines (which will represent the longest straight segments detected, obtained with `houghlines` [87]). This implementation was done as a function `houghTest`, represented in Figure 4.7).

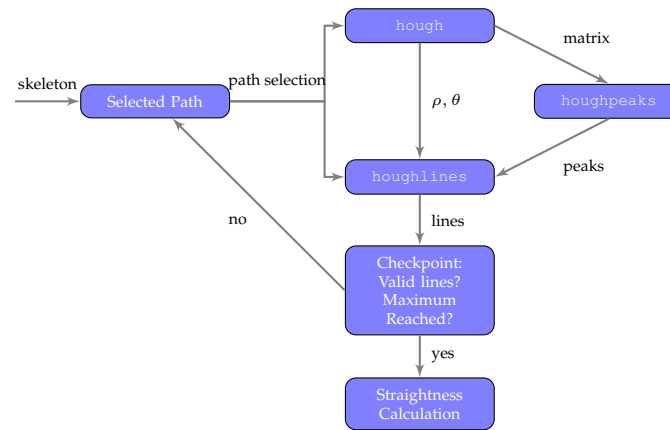


Figure 4.7: Implementation of the Hough Transform, as per the file `houghTest.m`. After the user has selected the path, the one-pixel wide coordinates are the input for the `hough` function, which will output the values of ρ , θ and the parameter matrix where these are represented in rows and columns, i.e., the bin count matrix. This matrix is then the input for `houghpeaks`, that will retrieve the N highest count bins. Finally, with the skeleton, the values of ρ and θ and the N peaks, `houghlines` will select the merged lines in the path that correspond to the values of ρ and θ for the peaks. There is a checkpoint for saturation: validity of the lines, number of peaks reached, entirety of the lines covered. In each iteration, the chosen lines are removed from the selected path. If any of these are met, the program will then calculate the straightness.

Furthermore, the implementation has an end user in mind, with the main goal being the seamless calculation of the transform, while allowing for adjustments to be considered.

It is integrated on the pipeline (further explained in Chapter 5) after the selection of the path made by the user, which is passed into the function as the skeleton of the desired region (and no other background). This means that all of the non-zero values that are selected in the region parsed to calculate the straightness are relevant. An important remark is that while the calculation is done in the (x, y) space, the graph of the image is also considered for the validity of the nodes.

Even though MATLAB provides the tools for the retrieval of the lines, it doesn't check their validity. As such, a quality control has to be implemented.

Given the list of the lines chosen, the first consideration relies on checking whether there is an overlap between any of the lines. As mentioned, if the quality of the image is sub par, the surrounding bins of the true highest count would also be selected as peaks, which would result in the overlap of lines (Figure 4.8). To overcome this, using the graph nodes, the program checks whether there is an overlap of over 10% of the shortest of the two lines.

After the lines have been confirmed, there are two extra steps to check whether the program can stop the calculation of the longest straight lines:

Has the intended number of peaks N been reached? As the program doesn't

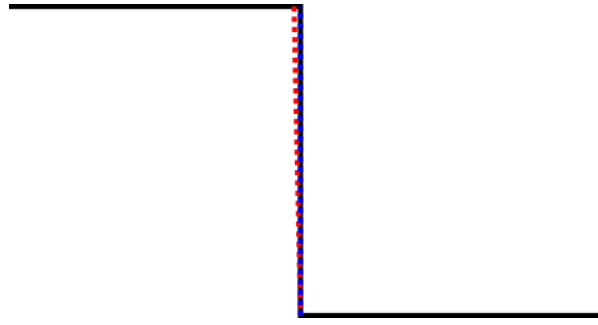


Figure 4.8: Example of how an overlap could occur while performing a Hough transform. Without the quality control, the Hough transform function would recognise the end points of both dashed lines (in red and blue) as they have different start points and a slightly skewed angle, which would make the function consider as a different (ρ, θ) pairing.

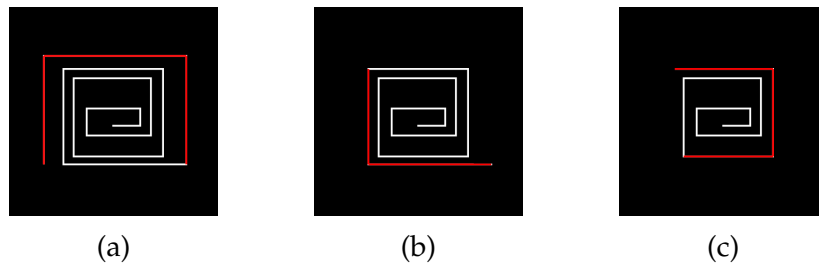


Figure 4.9: Iterative sequence of the file `houghTest.m` applied to a skeleton.. (a) is the first step while (c) is the last. In each iteration, the longest straight segments (red) are changed from one to zero so that the next set of longest straight segments can be chosen.

automatically search for all the possible straight lines, the user has to determine a maximum value. Even though there is a limitation because that means we are capping the number of peaks to be found, it's balancing the computational power that it would potentially require if it were to exhaust the number of lines available in a given image, for example high density of independent curved lines.

Have we covered the entirety of the desired path? Sometimes, the images comprise of only a handful of straight segments. As such, the length saturation, i.e., a straightness of $S = 100\%$ is achieved. This would mean that no other lines can be found and, therefore, the program can stop running.

If the answer to any of the questions is yes, then the program will exit and proceed to the calculation of the straightness. However, if both are negative, then a new temporary skeleton is created where the already selected longest straight lines are changed from one to zero, and is used in a recursive call of the `houghTest` function (Figure 4.9). The application will then either select everything and the temporary skeleton is a matrix of zeros, or will saturate the number of longest straight lines - this guarantees that the largest amount of straight segments (i.e. N) can be retrieved from the path.

The application is written in a way that future users will not see the reiterations,

and that any changes made on the selection will result in an automatic update of the calculations.

Validation

Before implementing the method and incorporating it into the general pipeline, it is pivotal that critical points are addressed and tested, as well as guaranteeing a proper quality control. Given that resolution is one of the key points in the analysis performed, one has to understand what the limits of the application are:

For example, what is the minimum distance (in pixels) that the algorithm recognises as two different lines? What is the minimum angle between two lines for them to be recognised as distinct? How much does the positioning of a line (i.e. vertical vs diagonal) influence the end result of the transform?

To answer these, some synthetic images were created to explore the edge cases of the implementation (Figure 4.10). An important note is that these images were created as vector graphics (image file that uses mathematical functions to describe the shapes rather than pixels), but compressed into `.png` files to emulate a real pixelated image.

With regards to the resolution limitations, as shown in Figure 4.10, lines are distinguishable until separated by only one pixel or with minimum angle of 1° between them. However, the method does not perform as well with diagonal lines as it detects far more segments than present.

Curiously, the algorithm recognises the same number of segments for the diagonals at 5° and 1° . The nature of these recognised segments is different, however, as in the second case it appears to be consecutive segments in the same diagonal line, whereas in the first case it recognises the distance between the two lines.

Table 4.1 shows the results of straightness calculation in different test images. The tests were performed without a cap on minimum or maximum number of segments detected, which caused a variation between different images. Besides the total straightness calculation (where the sum of the lengths of segments detected is divided by the total length), two other measurements are present: average segment straightness (total straightness divided by the number of segments detected) and the ratio between the Euclidean distance of the end points per the actual length (only applied to the linear images).

In each of the different types of test images (Neuron, Disorganisation and Lines), there is a version comprised of curved lines (above) and straight lines (below). The skeleton is presented for each case, with the extracted segments highlighted in green (yellow is the start, red is the end).

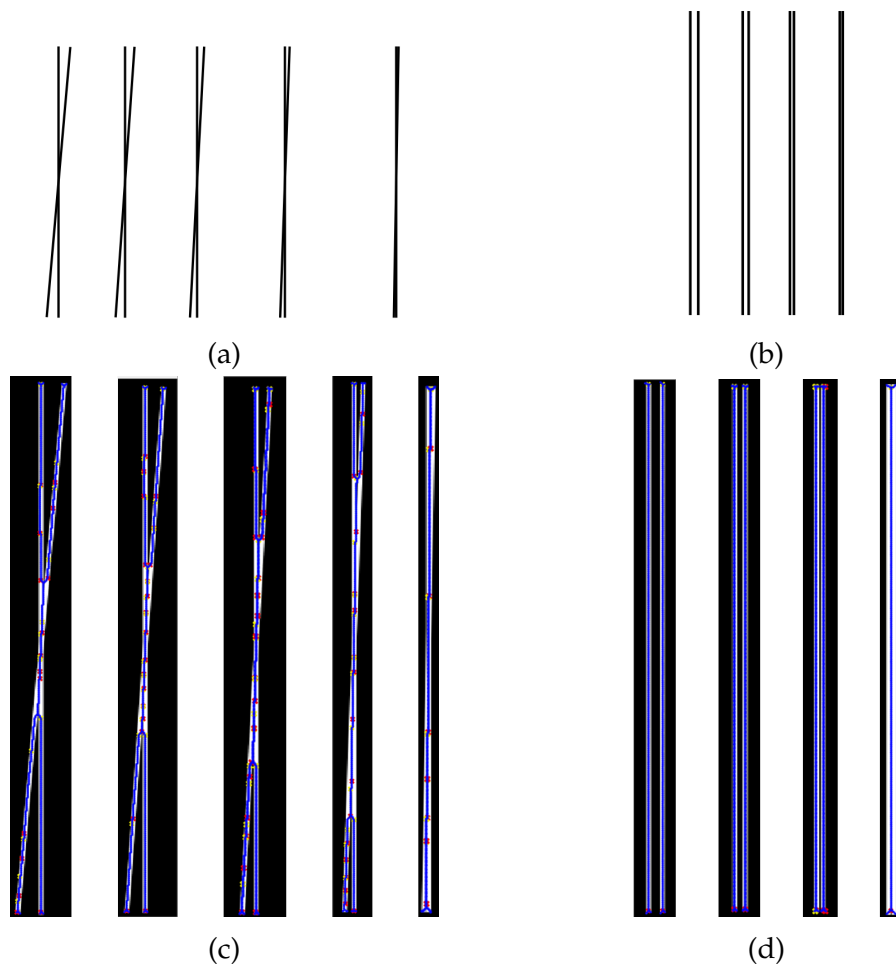


Figure 4.10: Synthetic images to test some limitations of the Hough Transform. (a) has two lines with an angle from 5° to 1° . (b) has two parallel lines increasingly closer. In (c) and (d) is segment recognition (blue) using the `houghTest` function with edge cases. In (c), the function recognises many straight segments in the rotated line (from left to right, the function recognises 7, 20, 19, 17, 7 segments on each group), and there is a loss of information in the overlap area. (d) The function manages to recognise both segments in all cases except when the lines are separated only by a pixel (which is lost when compressed into an image file).

The average segment straightness can also be an element of comparison between shapes, as evidenced by the "Neuron" section in Table 4.1. In this case, the Euclidean metric appears to provide a better result but it is not adaptable to any other type of measured length (i.e. if it has higher angles, as in the "Line" section, it loses a lot of information).

When comparing disorganisation images, similar number of straight segments provides significantly different ratios.

There can be a performance bottleneck: if the segments are a small part of the image, the ratio will take a long time to be calculated, and the algorithm might not find all the straight segments in an image.

4.3 Curvature

1

To the human eye, one of the most distinguishing characteristics of a disorganised phenotype are the “loops” in the microtubules, generally organised into tight bundles. It is important to quantitatively understand how much microtubules deviate from their usual straight shape. Mathematically, it is the definition of curvature. There was a pivotal effort into retrieving curvature as a parameter from the disorganisation images.

Mathematical Description of Curvature

Even though curvature can be any number that is roughly related with the shape of an object, in this project we are interested in the geometric curvature κ , which describes the rate of change of the slope of the curve [70].

$$\kappa = \frac{1}{R} \quad (4.3)$$

It is the reciprocal of the radius R (orange arrow, Figure 4.11) of a circle (blue, Figure 4.11) that is tangent to the given curve at a specific point. The curvature will be zero for straight lines, and increase for progressively smaller circles.

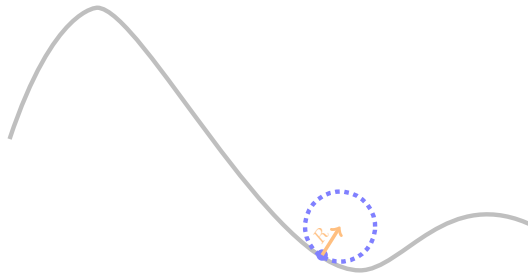


Figure 4.11: Mathematical definition of radius of curvature (orange), as per Equation (4.3). Blue dot represents the point where the radius is being obtained, and the dashed line represents the circle with radius R that is tangent to the point.

By working with the skeleton, each point in the disorganisation can be described by their (x, y) coordinates. Suppose the dot in Figure 4.11 moves along the curve at a constant pace. Mathematically, its position can be described in terms of the position vector $\vec{p}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$, where t is the instant of measurement.

The velocity of the dot will therefore be $\vec{p}'(t) = \frac{d\vec{p}}{dt} = \begin{bmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{bmatrix} = \begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix}$. As the dot moves at constant pace, the velocity is only influenced by the slope of the object. The

¹Acknowledgement: the mathematical theory behind this section was developed in collaboration with Nuno Nobre.

rate of change for this velocity is the tangent vector of the curve and represents the curvature.

$$\kappa = \frac{d^2\vec{p}}{dt^2} = \frac{|x'y'' - y'x''|}{(x'^2 + y'^2)^{3/2}} \quad (4.4)$$

Setbacks and Theoretical Approaches

Curvature measurements have been historically complex [70, 88]. The greatest problem with computing the curvature is that we are collapsing the real continuous image onto a discrete grid (pixels). Unless the capturing of the images was done into vector graphics (where the image is saved as the description of what is portraying, rather than the RGB values per pixel), which is not possible for now, continuous lines (e.g., microtubules) will be forced onto a discrete shape.

Several approaches have been developed, not without shortcomings. In this section, I will briefly explain them and which was chosen for the software.

Pixel Angles and Distances

The simplest approach is by calculation of the curvature using the angles and distances between the pixels on an image (Figure 4.12).

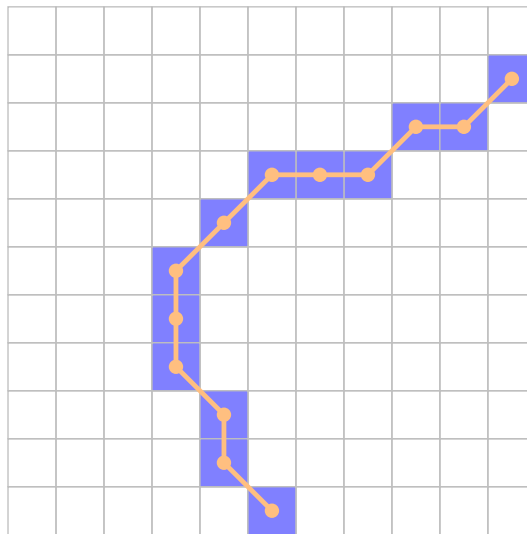


Figure 4.12: Pixel Angles example: The curvature is calculated by the relation to the angles between skeleton pixels (in blue), or vectors between the centre (in orange).

Some estimations can be better than others and although it has improved, the major issue relies on resolution dependence: the same real line will have very different pixel description depending on how many pixels are available. Furthermore, given the discrete nature of the grid, it adds an artificial straightness to the curvature.

This method of calculating curvature was applied in FiberApp [51], where a distance of three pixels is used.

Pattern Fitting

This method consists in the fitting of an existing pattern (in this case, known and specified circles) to the image in order to find the curvature values (Figure 4.13). An example of this would be applying the Hough transform described in Section 4.2 but with circle detection.

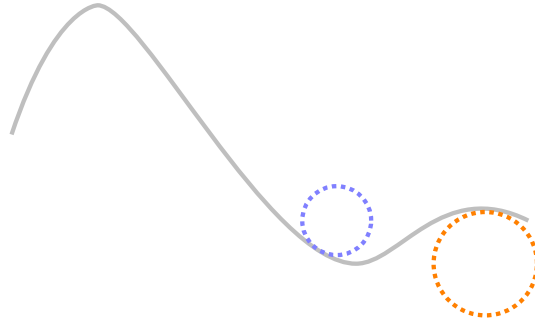


Figure 4.13: Pattern Fitting example: The algorithm searches for all of the lines that will fit the arc for different circle sizes (illustrated in blue and orange, dashed).

It is limited to approximate curvature with the most similar values of the given patterns. However, it is a brute force approach to find all the curvatures present in the image, which can have very limiting performance values.

Different implementations have been done by [89–92].

Polynomial Fitting

Like regression in statistics, where a straight line is calculated from the data, a different approach to curvature calculation can be fitting a curved function to the lines of the image (Figure 4.14, blue dashed line on the skeleton).

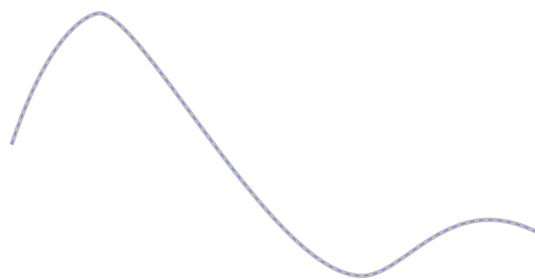


Figure 4.14: Fitting a function to the lines: the algorithm will try to fit a single polynomial (blue, dashed) to the entire line (grey), with Equation (4.5).

Curved lines can be described by polynomials, of which the exponent of the variable provides the degree.

$$f(x) = \sum_{n=0}^d C_n x^n \quad (4.5)$$

For example, a straight line is a polynomial of first degree ($n = 1$). By having a function for both axis, the curvature calculation from these polynomials is trivial as the first derivatives can be easily calculated by:

$$f'(x) = \sum_{n=0}^d nC_n x^{n-1} \quad (4.6)$$

and the second:

$$f''(x) = \sum_{n=0}^d n(n-1)C_n x^{n-2} \quad (4.7)$$

While it can be a powerful application, there are several setbacks. If the function has a defined limit for the degree (for example, using quadratic functions), then it will not work for all types of images. On the other hand, if the degree can change with each image, it would be very time and computationally consuming. Furthermore, if the image is intricate and a high degree polynomial is needed, then it might occur artificial oscillation at the edges of the function, known as Runge's phenomenon [93] - which states that increasing the degree higher than a certain level does not increase accuracy.

This method has been applied on a variety of fields, for example land surface curvature [94] or agriculture [95].

Fourier Transform

On the same type of approximation, instead of using a polynomial, it would be possible to use functions that already has a curved shape: sine and cosine. By using a Fourier transformation, the lines are described as:

$$f(x) = a_0 + a_1 \cos(\omega x) + b_1 \sin(\omega x) \quad (4.8)$$

The fitted coefficients a_0, a_1, b_1 and the frequency ω are then used on the derivatives and consequently on the curvature calculation.

It is a general approximation and provides one result for the entire image: when looking at structures such as microtubule disorganisation this type of curvature calculation will not provide sufficient information. It is used, for example, for damage detection in 2D images [96].

Gaussian Windows

A different approach is by using Gaussian windows [97]. A window filter (similar to those mentioned in 3.2.1) of a certain size is applied to each point of the skeleton.

As mentioned in 3.2.1, the convolution of an image with a Gaussian function is retrieving the first derivative. By using a point to point multiplication with the filter,

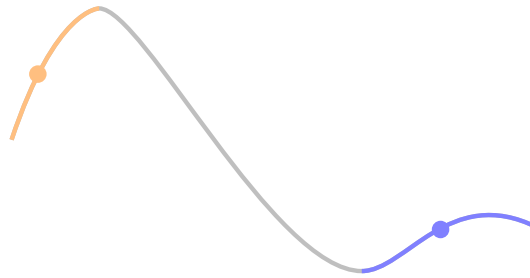


Figure 4.15: Gaussian Windows example: A window of a certain number of neighbours around a certain point (orange in one point, blue in another) will run through the lines, where the line is multiplied point to point with the filter, and the final sum reduction produces the desired derivatives.

a final sum reduction produces the desired derivatives. Each point provides a curvature: with an adjusted window size, it can work out good approximations for the local curvatures along the image.

However, having an adaptive window size can be computationally consuming, and it would have to adjust for each image. Therefore, differences in curvature within the same image could potentially be missed by this method.

Smoothing Spline Fitting

The method used in this thesis is an improvement from fitting polynomials, where the principles are the same but instead of applying the function to the entire image, there are knots (orange in Figure 4.16) that split the image into different parts.

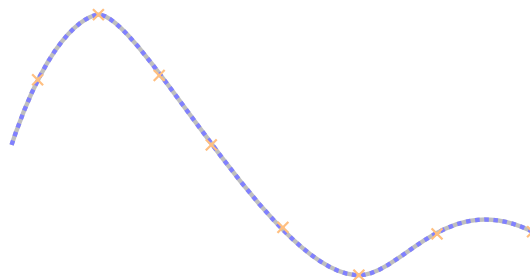


Figure 4.16: Smoothing Spline Fitting example: The line is split using knots (orange, crosses). To each segment is then fit a polynomial (blue, dashed) that will be, at most, of degree 3 (See Equation (4.9)).

A polynomial of degree three (Equation 4.9) is then fitted to the segments.

$$f(x) = \sum_{n=0}^3 C_n x^n \quad (4.9)$$

Using this piecewise polynomial, local curvatures can be calculated for any desired point.

4.3.1 Testing

Curvature calculations are generally presented as distributions of values. However, it does not provide locational information, which is necessary for an accuracy assessment and comparison. This can be circumvented if the curvature (or the inverse, radius) is described as a function of its length (i.e., the cumulative arc-length). Fundamentally, the radius is plotted for each point on the shape in order of the sum of arc-lengths from the origin until said point.

As curvatures can be analytically calculated, the most accurate way to assess our method is by comparing directly with analytical functions of well-known shapes. The simplest shape for these comparisons, logarithmic spiral (Figure 4.17a), has the radius growing linearly with the arc-length. Further, another shape of interest is an ellipse (Figure 4.17b), with a varying radius along the length (sinusoidal behaviour with higher values along the smaller axis).

The initial point (arc-length of zero) for the logarithmic spiral is the centre and it moves outwards. The radius of an ellipse has a periodic behaviour, so the start point is not relevant.

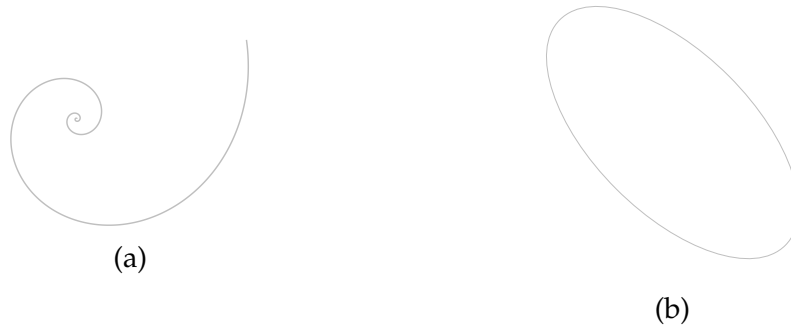


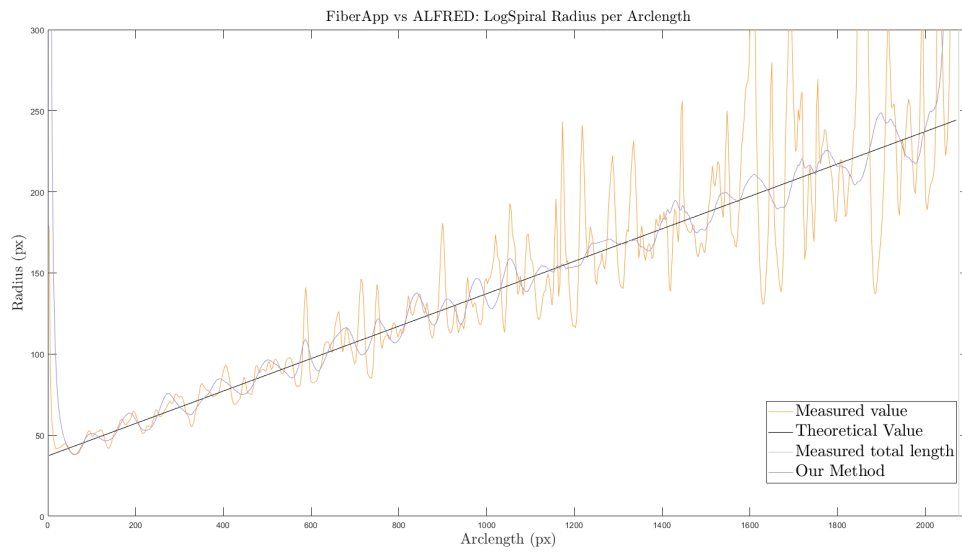
Figure 4.17: (a) Logarithmic spiral as per equation present in Table 4.2. In this case, $a = 1$ and $b = 2$, $\theta \in [8\pi, 14\pi]$. (b) Ellipse as per equation present in Table 4.2, rotated 45° , with $a=1$, $b=2$ the small and big axis, respectively.

The analytical functions for these shapes are in Table 4.2, including their curvature, arc-length and the range of angles θ in which the measurements are done (limited range for the logarithmic spiral as it grows infinitely and one full circle for the ellipse).

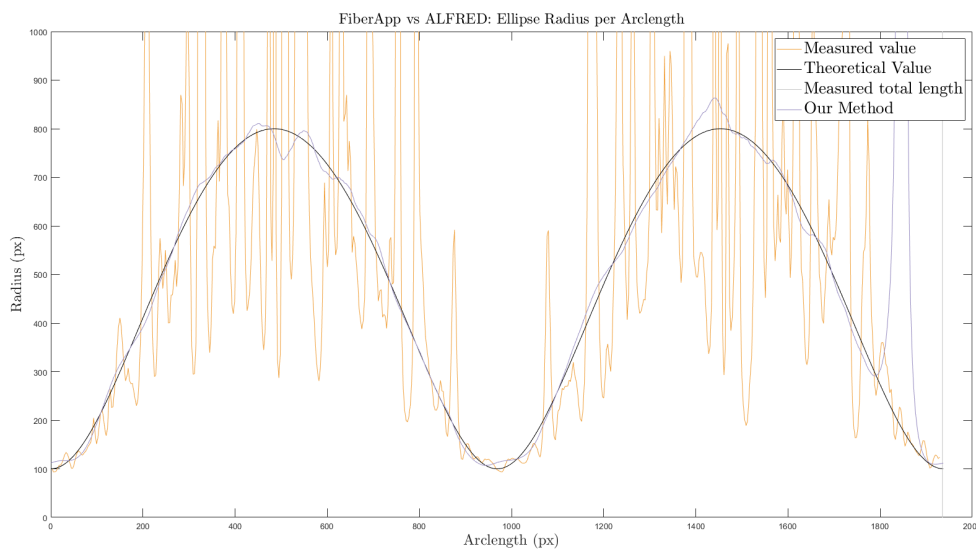
The length is calculated as a cumulative arc-length, i.e., the distance from any point to the beginning of the measurements is the sum of all the small arc-lengths calculated for each section of the line.

Figure 4.18 shows the plotted radius as a function of the cumulative arc-length for the analytical function in black and our results in purple. In order to understand whether our radius/curvature calculations are more accurate than existing methods,

the values obtained with FiberApp are plotted in orange. The grey vertical bars correspond to the total length of the shapes.



(a)



(b)

Figure 4.18: Plotting of radius values per cumulative arc-length for logarithmic spiral (a) and ellipse (b). Three different radius values are plotted: black for the analytical function, orange for the values measured with FiberApp and purple for our method. The grey vertical lines depict the total length measured.

To analyse the performance of both methods, the MATLAB function `immse` was used to calculate the Root Mean Squared error (RMSE) between the calculated values and the respective expected (or theoretical) counterpart. The results are presented in Table 4.3.

When analysing the values for the logarithmic spiral (Figure 4.18a), both calculations are erratic around the theoretical values. However, while for smaller radius it seems that the frequencies are similar, the difference becomes evident for higher radius, where the amplitudes are significantly divergent. This results in a slight decrease of the RMSE.

For the ellipse, however, the difference between the two methods is substantial. While there are some variances for higher radius (and, therefore, for smaller curvatures) with our method, it shows a much smoother approximation to the expected values than the one provided by FiberApp, which is visible in the RMSE values. An important note is that the RMSE were calculated ignoring the artefacts of the curvature calculations.

The small variance of our values compared to the theoretical ones can be explained by the discrete description of the shapes on the pixel grid, which provide artefacts that are not present in the continuous analytical functions. As such, the curvature distributions provided by our method provide an approximation to the local curvatures in the real images.

Real images contain, however, intersection between different lines and these create an artificial curvature (Figure 4.19a). To avoid skewing the results with these, an intersection removal step is performed before any curvature calculation can be applied (arrow, Figure 4.19b).



Figure 4.19: Skeleton treatment for curvature calculation. (a) For any lines (grey), the first step to obtain curvature relies in the retrieval of the skeleton (in blue). (b) Given the skeleton, we remove the junctions (arrow) as these are points of fake curvature, i.e., abrupt line change would give a high curvature.

As all quantitative results can now be calculated and the pipeline is defined, it is necessary to make it user-friendly in a continuous environment.

Table 4.1: Results of calculating the straightness after applying the Hough Transform to different test images. **First column:** original drawings (above) and skeleton of the images (below - straightest segments in green, start in yellow and end in red). **Second column (Number of Segments):** detected number of straight segments. **Third column (Total Straightness):** ratio between the sum of the length of those segments per sum of all the pixels in the chosen part of the skeleton. **Fourth column (Average Segment Straightness):** ratio between total straightness and the number of segments. **Fifth column (Euclidean Metric):** ratio between the Euclidean distance of the end points and the length of the path, in the linear cases.

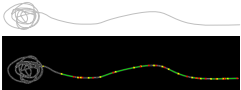
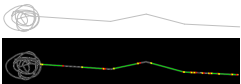
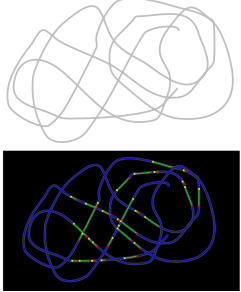
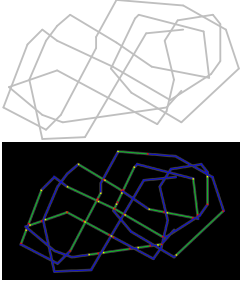
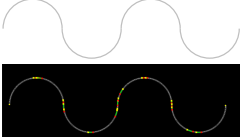
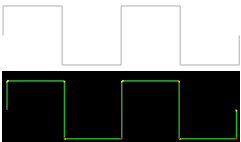
	Number of Segments	Total Straightness (%)	Average Segment Straightness (%)	Euclidean Metric (%)
Neuron				
	17	66.3	3.9	92.7
	8	68.4	8.6	95.1
Disorganisation				
	31	22.8	0.7	—
	28	43.6	1.6	—
Line				
	8	15.2	1.9	60.5
	9	98.4	10.9	50.2

Table 4.2: Analytical functions describing a logarithmic spiral (Figure 4.17a) and an ellipse (Figure 4.17b). The range of angles for the logarithmic spiral is limited to provide a resolution that would be visible by the methods (smaller values would correspond to pixel-sized curves).

	Logarithmic Spiral	Ellipse
Equation	$r = ae^{b\theta}$	$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$
Curvature	$\kappa(\theta) = \frac{1}{a\sqrt{1+b^2}e^{b\theta}}$	$\kappa(t) = \frac{ab}{(b^2 \cos^2(t) + a^2 \sin^2(t))^{3/2}}$
Arc-length	$s(\theta) = \frac{a\sqrt{1+b^2}e^{b\theta}}{b}$	$s(\theta) = a \int_0^\phi \sqrt{1 - \sqrt{1 - \frac{b^2}{a^2}} \sin^2(t)} dt$
	$\theta \in [8\pi, 14\pi]$	$\theta = \tan^{-1}\left(\frac{b}{a} \tan(t)\right) \in [0, 2\pi]$

Table 4.3: Root mean square error calculated between each method and the theoretical values of radius.

	Logarithmic Spiral	Ellipse
FiberApp	23413	1.57×10^9
Our Method	22112	2373.3

Chapter 5

ALFRED: the pipeline and the software

5.1 Rationale

The main goal of this project is to create a software package with an image analysis pipeline to facilitate the analysis of biological images most commonly obtained in the group. One aim is to make the analysis more independent of variations in user input. Another is to enhance the image analysis capabilities by two important parameters: straightness and curvature. The methods mentioned in Chapter 3 and 4 build the basis for the functionality of the software package.

One potential option could have been the implementation as ImageJ plug-ins. However, ImageJ has its own scripting language, which I was not familiar with. Instead, I was already proficient in MATLAB. The language offers a wider range of tools for image processing, allows the statistical analysis at the end of the parameter acquisition, as well as data visualisation, and easy interface implementation with its GUIDE functionality. These allow not only the easy incorporation of the scripts already written but also an overview of the interface and can automatically generate the necessary code for the components of the interface.

The end users of these methods are preferentially biologists, often with limited IT knowledge. As such, to achieve a wider use and data collection, a user-friendly implementation becomes imperative. I chose to add a Graphical User Interface (GUI) to allow end users an easy interaction with the underlying algorithms and control a number of input parameters. With the implementation of a GUI, new challenges appear: the software needs to be robust to any user interaction. Even if the order of buttons to use is clear, users can forget steps and that could cause the software to break (for example, trying to get a skeleton from an RGB image).

My aim was to develop an interface that is user-friendly and robust, which allows people with non-programming backgrounds to analyse their images with the algorithms that were chosen and developed, as a strategy to gain more and better data about microtubule organisation in the future, avoiding the time-consuming necessity to pre- and post-process the images.

I therefore decided to develop my own software in MATLAB, named ALFRED, the acronym for Advanced Labelling, Fitting, Recognition and Enhancement of Data [98].

In this chapter, I will explain the software implementation of the methods mentioned in Chapters 3 and 4 as the user-friendly ALFRED program. I will explain the strategic reasoning behind the general structure of the interface, and all features gradually added to fit the expected needs of the target users.

5.2 Strategic decisions for software design

An important outcome of having analysis performed on a software with a higher level of automation is the reduction of human error, accuracy improvement, potential inclusion of new readout and, in consequence, the generation of more data that can help to decipher the scientific problem in question.

As a starting point and concrete example of application, ALFRED was developed as an interface that facilitates the work and image analysis within our group, i.e., the analysis of microtubule networks in the axons of primary neurons.

As such, ALFRED pipeline was closely designed to steps of the current workflow for neuronal image analysis in the group. The focus on a concrete goal made it possible that each version, ALFRED was extensively tested by my colleague André Voelzmann in order to assess the performance of the software and its general adaptability to the workflow from the users point of view.

In our group, images are obtained from *Drosophila* neuronal cultures following standardised procedures (Figure 5.1):

- at least **two** genotypes are needed in an experiment: e.g., wild-type and at least one mutant condition;
- each genotype will be cultured on at least **three** different slides providing three repeats¹;
- each experiment is performed at least twice, providing replicates of the same conditions;
- from each slide at least **30 images** are taken;
- each image contains at least **one neuron**;
- each neuron displays an **axon** which can have none to various **microtubule disorganisation regions**.

¹Repeat measurements are taken during the same experimental run, or consecutive runs. Replicate measurements are taken during identical but different experimental runs.

Region of Interests (ROIs) are sections of the image used for specific individual analysis. As both axons and disorganisation regions are analysed, each one is a ROI, and consequently this number is highly variable.

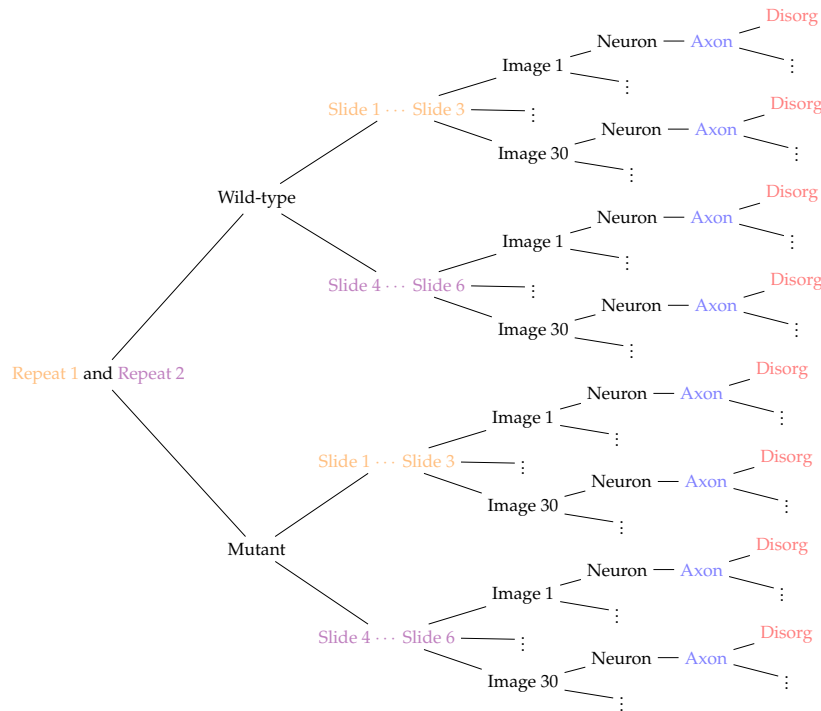


Figure 5.1: Diagram for the final number of ROIs in an experiment with only one repeat. Axons in blue represent ROIs for length and curvature analysis, disorganisation in red represents ROIs for microtubule curling analysis.

Before delving into the details of how to automate the analysis of these specimens through the implementation of ALFRED, I will first present an overview of the final outcome provided as a flowchart illustrating how ALFRED is organised (5.2).

The user interface has three windows: the loading window dedicated to the handling of files, the main window dedicated to image processing and the ROI window dedicated to path selection. Each will be described in more detail in the respective sections indicated in the brackets above.

Keeping the windows separate is a deliberate decision: it is easier for the user to follow the scope of the image: the loading window comprises the group of images, the main window focus on one image and the ROI focuses on one selected part of the image at the time. Furthermore, it has the added advantage that a program malfunction in one of the windows will not retroactively influence the work in the others.

The algorithms mentioned in the previous chapters are called also at different stages of the analysis, particularly given the level of the image (group of images, individual image, ROI within the image) and the computational power that it would require. During the pre-processing, both the filter algorithms and the application of a mask can be quickly applied to the entire image, and allow the user to have a better

perception of what the program can recognise. On the other hand, the skeletonisation (which is performance heavy) happens at a regional stage with the opening of each ROI window, and subsequently the path extraction happens only after the user has selected the end points. This also means that the length and straight segment acquisition also happen on the ROI window. Furthermore, the final calculations, including the straightness ratios and curvature, happen after the entire group of images has been processed, in the main window.

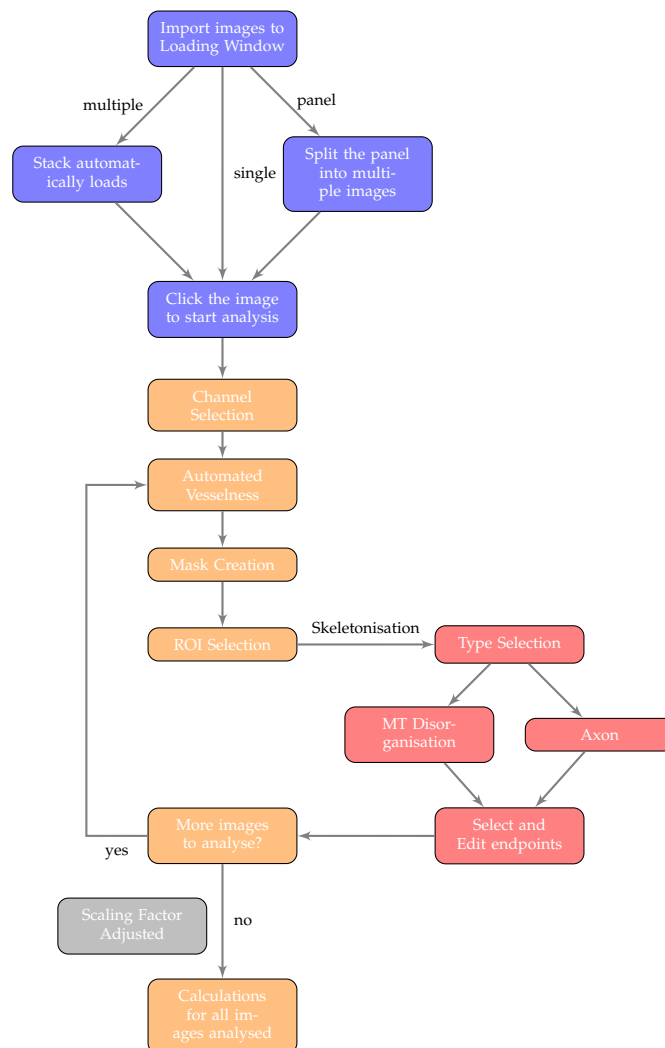


Figure 5.2: Flowchart of the user pipeline of ALFRED. The different colours represent the functions executed on different windows: loading window (blue) takes care of the loading of the images into the software, the main window (orange) takes care of navigation and the main pre-processing needed, as well as the calculations at the end, ROI window (red) has specific functions for either selection of axon or microtubule disorganisation, scaling factor adjusted on a pop-up (grey).

The implementation and user manual of the software can be seen in Appendix A.

A specific value: Microtubule Disorganisation Index (MDI)

Microtubule disorganisation is not a regular phenotype across different mutations. Several values can be compared, such as number of disorganisations or area. However, these alone are not sufficient. The impact of a disorganised region of a certain area will have a very different effect on a smaller axon. The Microtubule Disorganisation Index represents the ratio between area of microtubule disorganisation and axon length. The method so far was performed with manual calculations.

In ALFRED, both the area and the axon length have been obtained, so the ratio could be easily computed. The simplest solution would be the calculation of these values when the user selects the disorganisation, but does not account for the variable number of disorganisations in each axon.

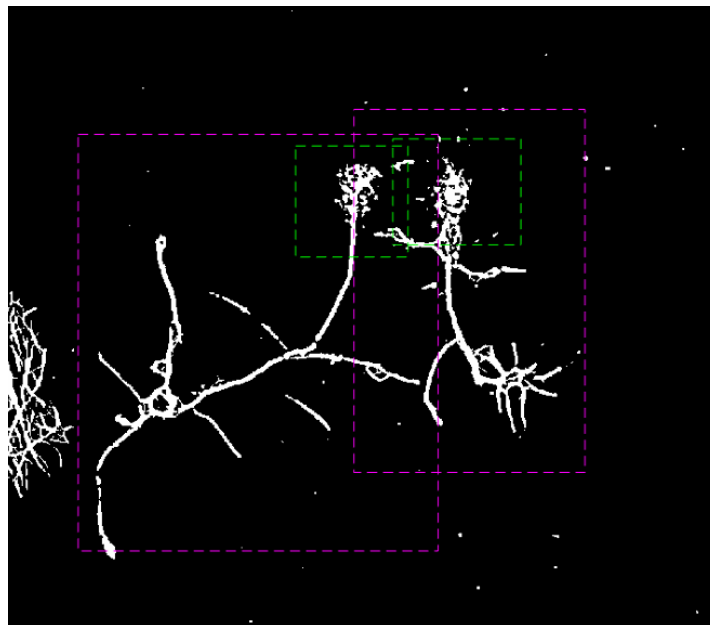


Figure 5.3: Example of overlapping bounding boxes of two neurons close by. In this case, the criteria to calculate MDI would be the percentage of overlap between the disorganisation bounding box and the axons. Magenta bounding box corresponds to axons, green to microtubule disorganisation. Shown area of width $29.7\mu m$.

After each ROI selection, the bounding boxes are saved and the designated processed areas are marked in the image (Figure 5.3, magenta and green).

Since the disorganisation regions are invariably a part of the axon, their bounding boxes will overlap on some level. The MDI can be calculated using the disorganisation bounding boxes that overlap significantly (if not completely) with the corresponding axon.

If one bounding box overlaps with more than one axon (Figure 5.3), the disorganisation will be considered for the axon with which it has a higher overlap.

In the unlikely case that the overlap is equal, a tie-breaker method is applied by

checking the overlap of axon skeletons, as these will invariably be part of the disorganisation. This method is much more time consuming and, therefore, is only used as last resource.

The values for MDI are attached to the already saved values of each axon.

Chapter 6

Biological Image Analysis

6.1 Description of the Images and Aim

After verifying the validity of each ALFRED component individually, it is necessary to evaluate its performance with the target images. As explained in Section 2.3, the biological data obtained in the group are fluorescence microscopy images of cultured *Drosophila* neurons.

6.2 ALFRED Analysis

Before any novel analysis can be performed with the ALFRED pipeline, it is pivotal to guarantee that the data acquired with the software is statistically non-significantly different from the manual protocol data.

The manual analysis was performed first, and ALFRED was tested by choosing the same ROIs as the manual user.

Axonal Length

Axonal length is one of the most important measurements for phenotype studies in our group. As described in Section 2.3.1, the manual analysis is usually performed using ImageJ via a click-based approach where the user clicks along the axon from the cell body to the axon tip (Figure 6.1(a)). For a relatively straight axon, this could be done in a small number of clicks but, as can be seen with the 20 white squares in Figure 6.1a, a lot more are usually necessary for curved axons. In contrast, with ALFRED the user needs to click only twice, on average (Figure 6.1b, green dots indicated by arrows). Extra clicks can be added to improve the path chosen by the algorithm.

Furthermore, ALFRED follows the pixels as they are lined up in the image, thus tracing curves accurately, whereas the manual approach is composed of straight lines that can only approximate bent sections (see magnifications in Figure 6.1). Both approaches allow scaling from pixels to micrometres.

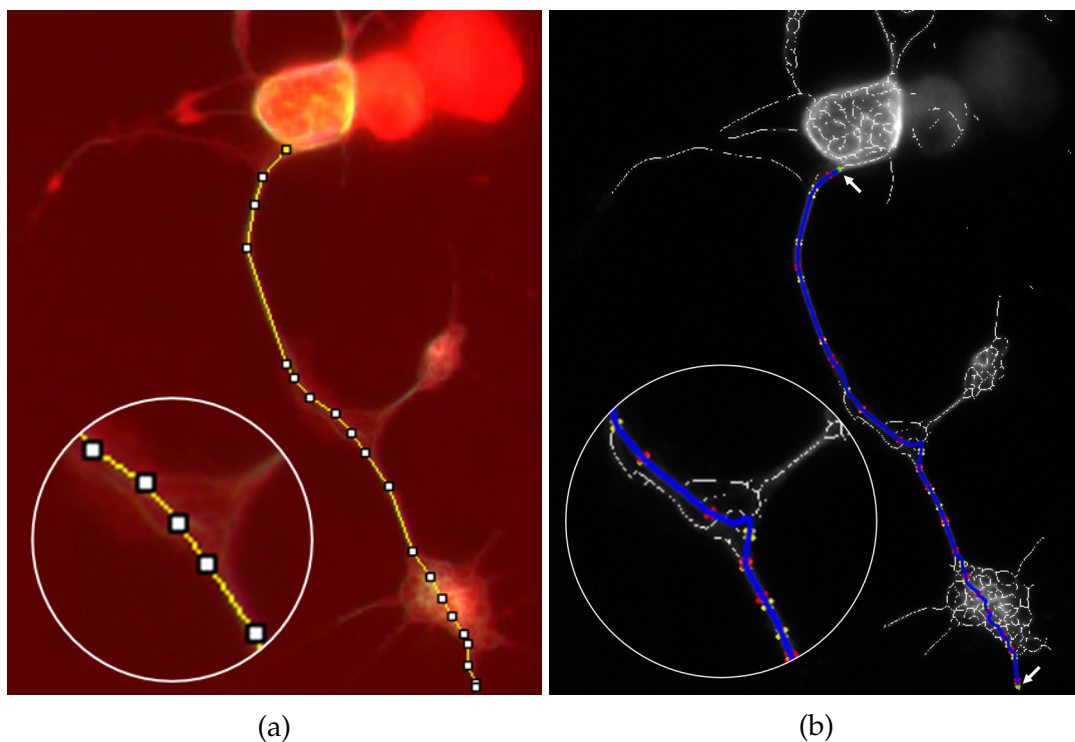


Figure 6.1: Manual measurement of axonal length with (a) ImageJ protocol and (b) ALFRED. The clicks performed by the user are represented by: 20 squares in (a), and 2 green dots evidenced by arrows in (b). Magnifications shows a region with a different path in the manual and ALFRED analysis. Shown area of width $19.1\mu m$.

To test the performance of ALFRED, four data sets representing different genotypes and each containing 30+ images, were analysed manually and via ALFRED. Data were sorted and plotted from the shortest to the longest axon (Figure 6.2). As can be seen, both analyses match closely, although ALFRED values (blue) tend to be slightly higher, especially in longer axons. This deviation is expected, since ALFRED precisely follows the path, including curves that were only approximated with straight lines in the manual analysis.

As already suggested by the graphs in Figure 6.2, also statistical comparisons revealed similarity of data. As shown in Figure 6.3, ImageJ and ALFRED analyses showed the same inter-genotype relationships that were statistically significant (blue connectors for ALFRED, orange for ImageJ). The distribution of the length values for the different genotypes was compared to the respective wild-type values and tested for significance in the differences using non-parametric multiple comparison tests. The p-values indicate with which probability the difference between the two datasets is random.

Furthermore, when comparing ImageJ versus ALFRED data for each genotype (normalised to the median of the wild-type), there were no significant differences (black connectors, all p-values are above 0.05).

These data were very encouraging and demonstrate that ALFRED is a reliable tool

to measure axon lengths in this case, suggesting that it will be applicable to length measurements in general.

Disorganised Area

In the next trial, I measured axonal areas containing disorganised microtubules scaled to square micrometres. The protocol with FIJI allows to draw areas freehand (Figure 6.4a). For the ALFRED analysis, however, no selection tool is necessary other than choosing two axonal points surrounding the disorganisation (i.e., the “beginning” and “end” of the disorganisation), with added clicks to remove any unwanted branches (Figure 6.4b) - the disorganisation area is retrieved as the area enclosed in the outer lines of the path. It is clear that the area calculated by ALFRED includes several creases that are overseen in the manual selection.

Figure 6.5 shows the sorted data for the converted measurements of area of disorganisation, for each genotype, with the two different analysis modes.

Figure 6.5 shows the area measurements arranged by size, obtained from the same neurons and the same genotypes used already for axon length measurements (Figure 6.2). While the graphs have a similar shape, there is a clear trend for ALFRED data to be gradually higher than ImageJ data, for larger areas. However, given the precision in pixel recognition present in ALFRED, these are expected to be higher than the manual analysis.

While the discrepancy seems higher in this case than the previous length, the explanation relies on the conversion itself: each pixel in an area is squared. This implies that the seemingly insignificant differences in linear results become enhanced when squared.

Accordingly, the medians of data normalised to the respective wild-type medians are always higher in ALFRED analyses. Upon statistical comparisons using non-parametric tests, the p-values comparing mutant genotypes to the respective wild-types are within the same order of magnitude for both ImageJ- and ALFRED-derived data, and comparisons between ImageJ- and ALFRED-derived data for each genotype are non-significant ($p\text{-value} > 0.05$, black connectors in Figure 6.6).

As both measurements provided non-significant differences with the previous manual analysis, there is an *a priori* confidence in the calculation of the microtubule disorganisation index. However, it is still necessary to analyse those results.

Microtubule Disorganisation Index

Even though the *de facto* measurements obtained from the images are the axonal length, and the disorganisation areas, the value used in publications and in the definition of the genotypes is the microtubule disorganisation index. This means that two independent values are being combined which might lead to a stronger deviation between ImageJ- and ALFRED-derived data than when comparing each value

in isolation. The results of the statistical analysis of these calculations are shown in Figure 6.7.

The genotype comparisons intra-analysis presents the same order of magnitude of difference between the genotypes and their respective wild-types when comparing ImageJ data sets (orange connectors) to ALFRED data sets (blue connectors). There is a tendency for MDI values derived from analyses with ALFRED to be higher than with manual analyses but the comparison within genotypes of the two analysis methods did not reveal any significance to these differences (black connectors).

Curvature

The first new value obtained with ALFRED is the curvature. In this case, there are no manually acquired values to compare with and, as discussed before, there is no other software available that performs these analyses. FibreApp can only measure single filamentous structures that need to be traced by hand.

The closest way to measure the real curvature of the disorganisation would be by manually calculating the radius of the loops. In Figure 6.8 is an example of manually obtained radii.

Given the literature available [23, 99], the true range considered will be of radius in the order of magnitude of hundred nanometres (10^2 nm).

With ALFRED, curvature calculations are applied to the path recognised in the "MT disorganisation" option, that retrieves the skeleton of the disorganisation (all the blue lines in Figure 6.4b).

In all graphs of Figure 6.9, the curvature is shown sorted by size. Each point corresponds to the curvature measured in each individual spline (i.e. each segment in which the algorithm splits the skeleton) as obtained per Section 4.3.

As the presence of disorganisation was significantly different within genotypes, this translates into a different number of measured points. Figure 6.9a evidences this: wild-type genotype has half the points of $khc^{27/27}$, and a third of the points of the other two genotypes ($shot^3$ and $efa6^{ko}$).

However, comparing just the number of points does not provide any information in the similarity within the disorganised regions, when they are present. To this end, Figure 6.9b shows the sorted values normalised for the number of points.

The distributions appear to have an exponential behaviour. In order to further analyse the values in the first half of the graph, a log transformation was applied (as per Figure 6.9c), where the initial behaviours become more evident.

The most evident difference is between wild-type and the other three genotypes. However, it seems that $efa6^{ko}$ is more similar to $khc^{27/27}$, with $shot^3$ slightly isolated. In other words, it appears that when disorganisation is present in wild-type, the log transformation curvature tends to be larger (i.e., smaller radius for the loops).

When measuring computationally, all the lines are considered, even the straight

ones. Furthermore, given the spline method used, there are millions of values for each genotype. In Figure 6.9, most of the curvature measurements are under $2.5\mu m^{-1}$ and above $0.05\mu m^{-1}$, which correspond to radii superior to $0.4\mu m$ and inferior to $20\mu m$.

It is also important to consider the average curvature of the region, as there are a lot of straight segments and pixel-length radius. In Table 6.1 is presented the average value for each genotype, after eliminating some outliers (i.e. the radius larger than 10^{11} nm, which corresponds to 99.99% of the data).

Table 6.1: Average radius measured with ALFRED for each genotype, after removing the values above 10^{11} nm. N represents the number of valid points per set, σ is the standard deviation.

Genotype	Average (nm)	N	$\frac{\sigma}{\sqrt{N}}$
wild-type	420	914959	29.2
khc ^{27/27}	154	2397918	6.74
efa6 ^{ko}	203	1767941	51.2
shot ³	1592.2	2210909	158.2

While the values for shot³ are slightly higher, all genotype curvatures appear to be within the expected range. Furthermore, the average values seem to concur with the above statement where the curvatures present in the khc^{27/27} and efa6^{ko} genotypes are closer than to the others. The higher values for shot³ do not necessarily mean larger circles, but can correspond to the presence of a higher number of straight lines within the measured splines.

Axon Straightness

The final value to be a novel calculation by ALFRED is the axonal straightness ratio, the total sum of the straightest segments measured divided by the total length. The path considered for the segment retrieval is the one obtained for the length measurements, as per Figure 6.1b. Figure 6.10 shows the values obtained for each genotype, for each of the axons recognised in the previous sections.

In Figure 6.10a, the ratio values are sorted by size, with the number of axons considered the same as the one analysed for Axonal Length and MDI. The general distribution for the mutations seems to be similar, even though shot appears to be slightly different from the other two. However, when the values are compared relative to the wild-type (Figure 6.10b), although the p-value obtained for shot³ is one order of magnitude lower than the other two there are no significant statistical differences between them.

Furthermore, the next step relies on trying automatic methods (with machine learning) to try and distinguish between the genotypes.

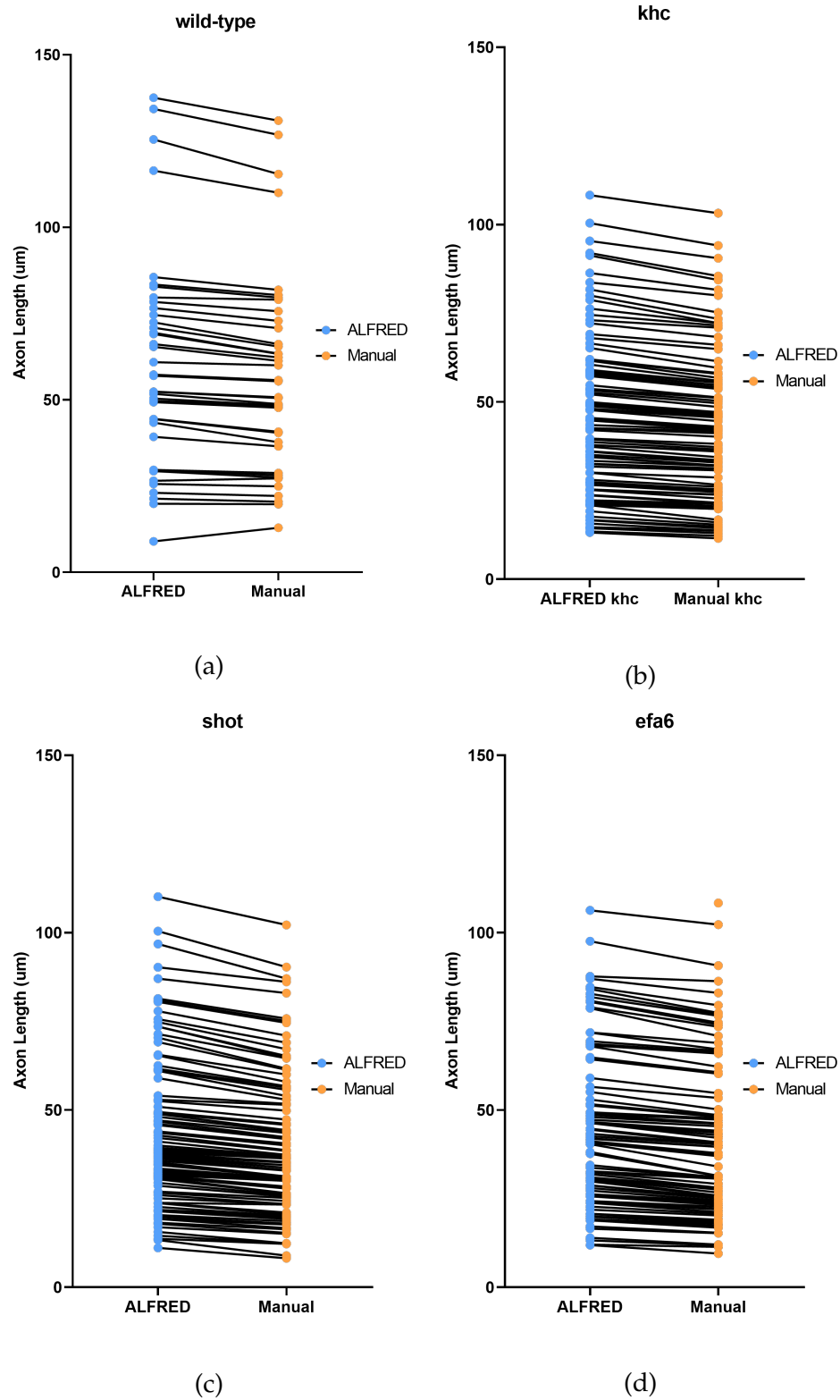


Figure 6.2: Ladder plot showing axonal length measurements obtained manually via ImageJ versus semi-automated ALFRED analysis, where the latter are usually longer. The cultured primary *Drosophila* neurons (a, wild-type, N=39; b, *khc*²⁷, N=92; c, *shot*³, N=91; d, *efa6*^{ko}, N=86 and N=87) were assessed for axon length using semi-automated ALFRED (blue) or manual ImageJ (orange) approaches.

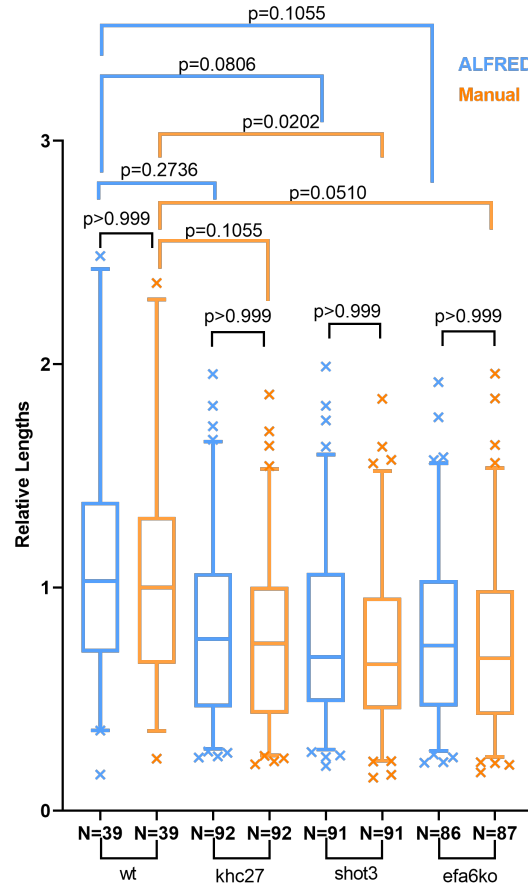


Figure 6.3: Statistical analysis of axon length data from Figure 6.2. Axonal length data for wild-type (N=39), *khc*²⁷ (N=92), *shot*³ (N=91) and *Efa6*^{ko} (N=86 and N=87) neurons acquired manually via FIJI (orange) or via semi-automated analysis in ALFRED (blue) were normalised to the respective median of wild-type. P-values were calculated using Dunn's multiple comparison tests, either comparing data of mutant neurons to the the respective wild-type, or comparing ImageJ versus ALFRED data for each genotype. Boxes represent the values within 1st and 3rd quartile, whiskers show 5-95% of the data, with the outliers presented as x.

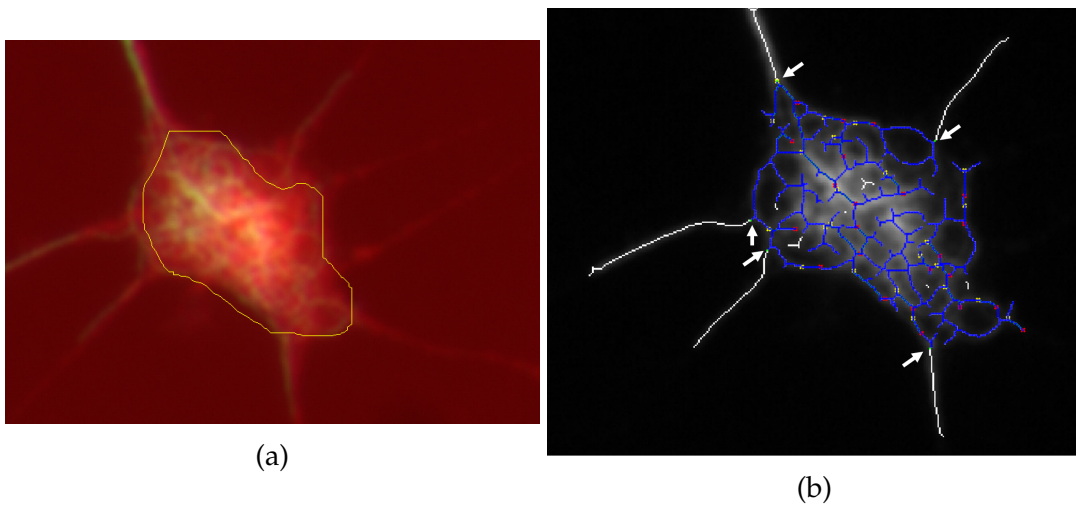


Figure 6.4: Manual measurement of disorganisation with (a) ImageJ protocol with the freehand tool and (b) ALFRED. The clicks performed by the user in ALFRED are shown with arrows. Scale: $22.16px/\mu m$.

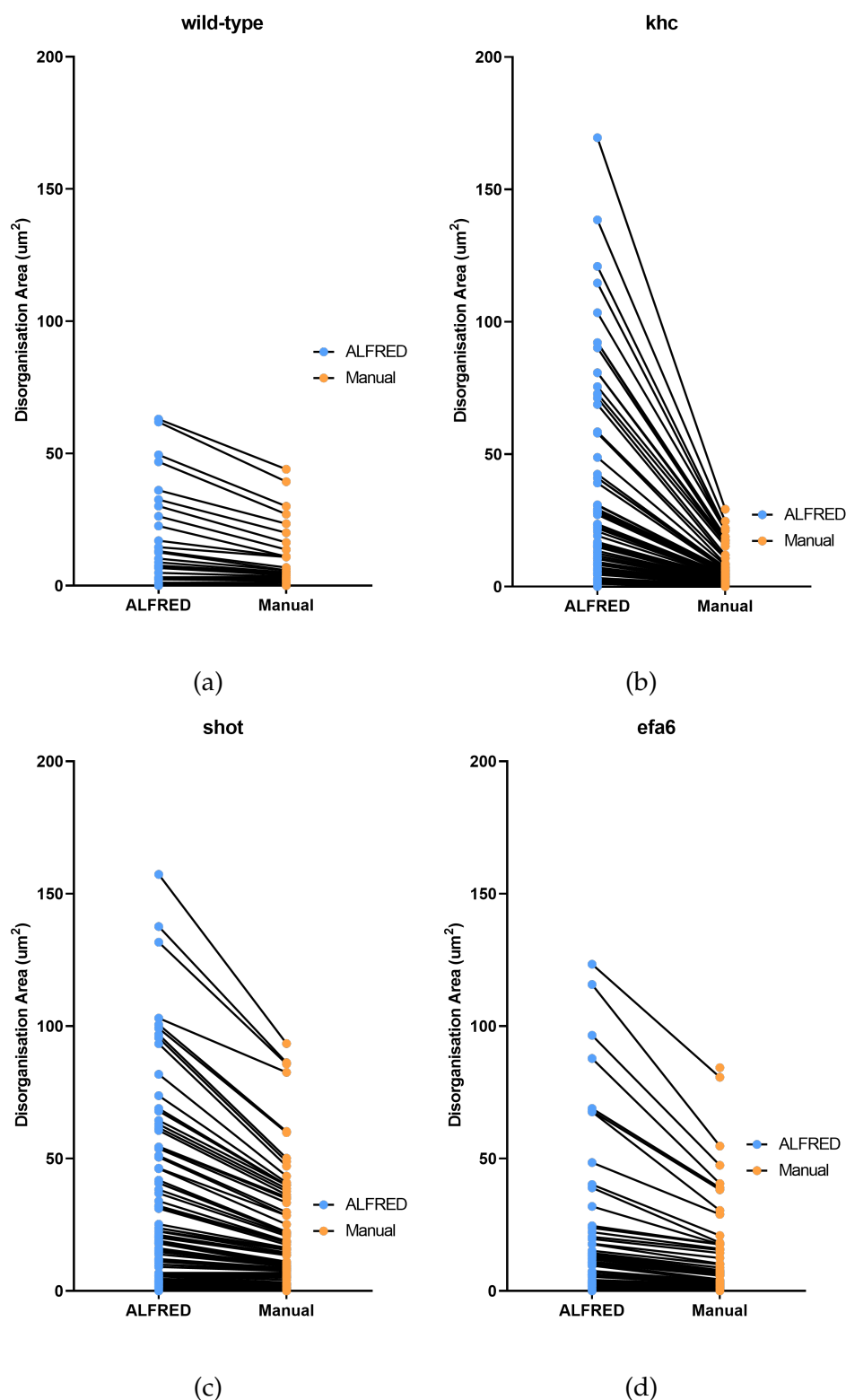


Figure 6.5: Ladder plots comparing disorganisation area measurements obtained manually via ImageJ versus semi-automated ALFRED analysis, where the latter are larger. The same cultured primary *Drosophila* neurons used for axon length analysis in Figure 6.2 (a, wild-type, N=39; b, *khc*²⁷, N=92; c, *shot*³, N=91; d, *efa6*^{ko}, N=86 and N=87) were used to measure axonal areas with microtubule disorganisation, either via semi-automated approaches with ALFRED (blue) or manual analyses with ImageJ (orange). Data were sorted by size.

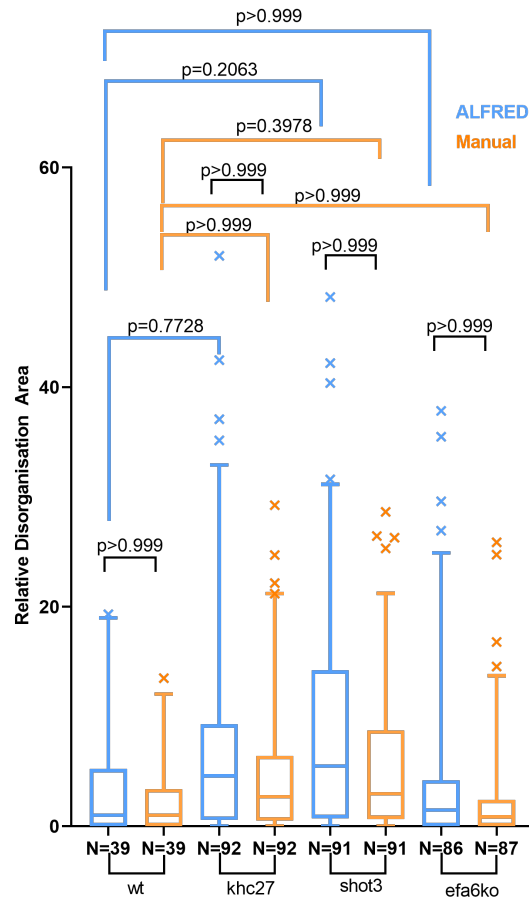


Figure 6.6: Statistical analysis of axon area data from Figure 6.5. Axonal area data for wild-type, $khc^{27/27}$, $shot^3$ and $efa6^{ko}$ neurons acquired manually via ImageJ (orange) or via ALFRED (blue) were normalised to the respective median of wild-type. P-values were calculated using Dunn's multiple comparison tests, either comparing data of mutant neurons to the the respective wild-type, or comparing ImageJ- versus ALFRED-derived data for each genotype. Boxes represent the values within 1st and 3rd quartile, whiskers show 5-95% of the data, with the outliers presented as x.

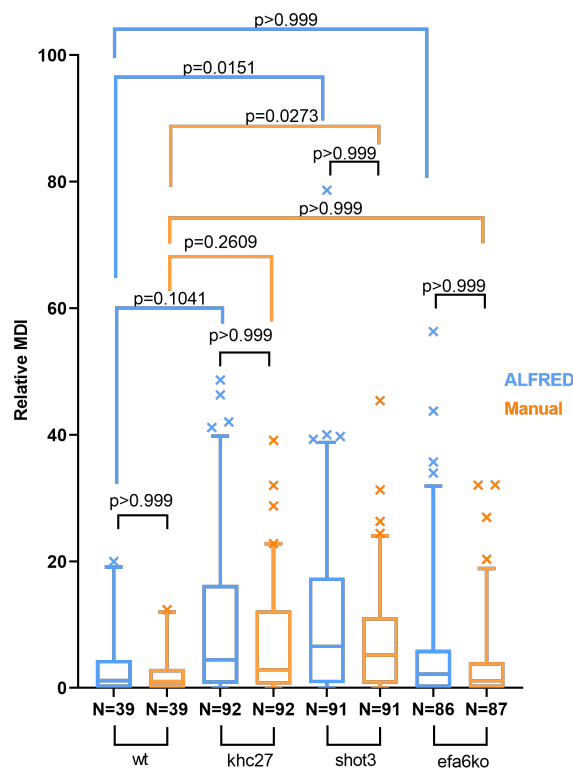


Figure 6.7: Statistical analysis of relative Microtubule Disorganisation Index (MDI) data calculated from ImageJ- and ALFRED-derived data. MDI data for wild-type, *khc*²⁷, *shot*³ and *efa6*^{ko} neurons calculated from manual ImageJ data (orange) or semi-automated ALFRED data (blue) were normalised to the respective median of wild-type. P-values were calculated using Dunn's multiple comparison tests, either comparing data of mutant neurons to the the respective wild-type (blue and orange connectors) or comparing ImageJ- versus ALFRED-derived data for each genotype (black connectors). Boxes represent the values within 1st and 3rd quartile, whiskers show 5-95% of the data, with the outliers presented as x.

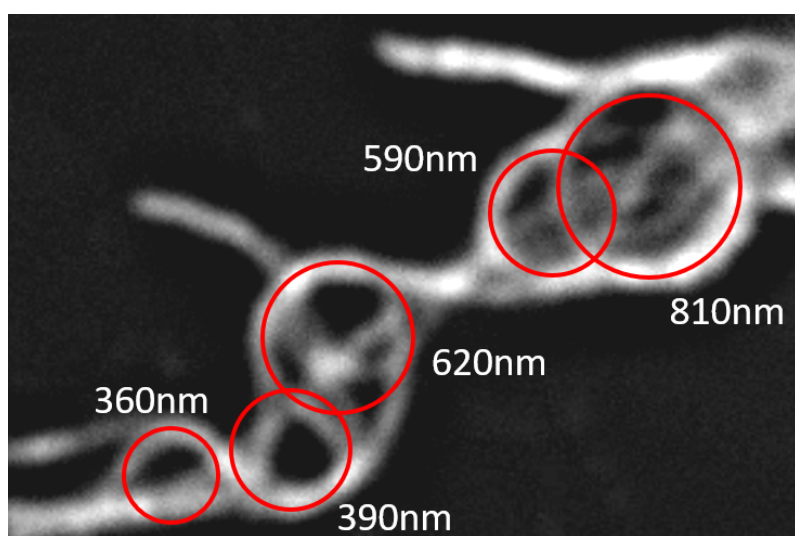


Figure 6.8: Manual approximate radius measurements. Red circles denote the apparent loops, with the respective radius in nm in white. Adapted from Dr. Simon Pearce's work [99].

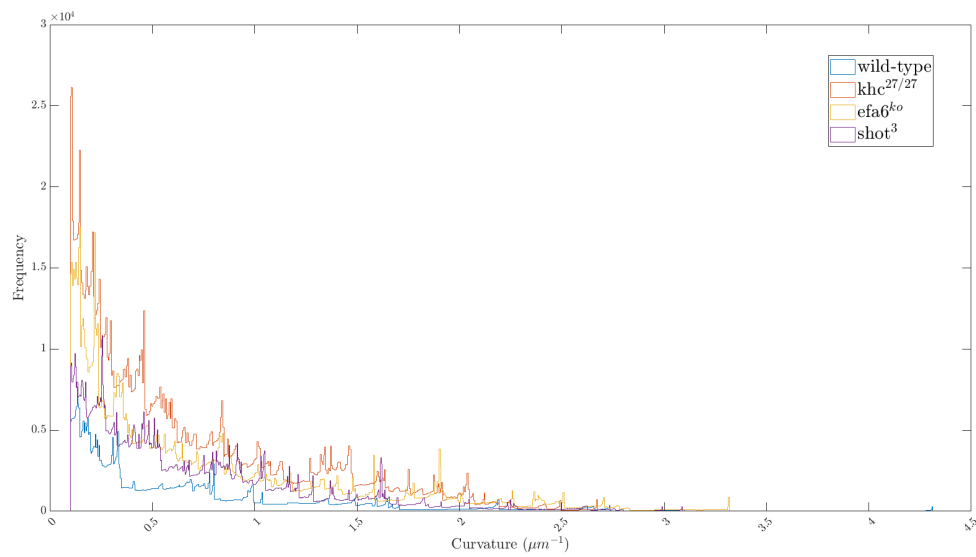
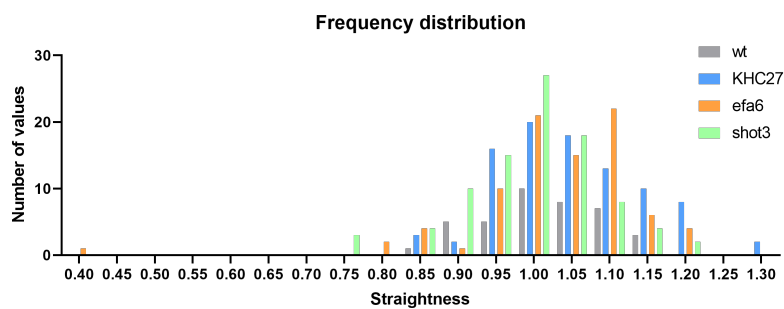
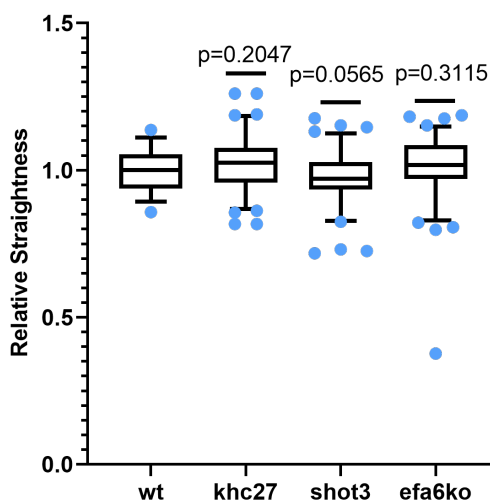


Figure 6.9: Curvature histogram profile of neurons with different genotypes. Curvature data obtained via ALFRED for areas of microtubule disorganisation in primary *Drosophila* neurons of different genotypes (colour-coded as indicated: wild-type, $\text{khc}^{27/27}$, shot^3 and efa6^{ko}). Each point represents the value of the curvature for each individual spline obtained from the image.



(a)



(b)

Figure 6.10: Straightness data obtained via ALFRED from axons of primary *Drosophila* neurons. Neurons were the same as in previous figures from the same four genotypes (colour-coded as indicated: wild-type, $khc^{27/27}$, $shot^3$ and $efa6^{ko}$). (a) Frequency histogram of straightness values. Each bar represents the number of axons with a straightness within the limits of the bin. (b) Straightness relative to the wild-type median. Boxes represent the values within 1st and 3rd quartile, whiskers show 5-95% of the data, with the outliers presented as dots.

Chapter 7

Using machine learning methods to further analyse the biological images

7.1 Background and Rationale

One of the key goals of the ALFRED software is to have as little user input as possible. Even though the calculation section does not require such input, the initial stages of processing and shape extraction do. Taking into account that different users might have different impressions of each image, as well as different line interpretations for the paths, automation would provide an extra level of user abstractness. Furthermore, it could make the pipeline not only faster from the user perspective but also allow the potential development of the software for uses in high-throughput experiments.

In all of the mentioned processes of this thesis so far, it has been under the assumption that the chosen parameters (curvature, straightness) are, in fact, the most relevant to characterise different phenotypes. However, this might not be the case. Although ALFRED provides a pipeline to analyse the experimental images of the group, there could be further information to be retrieved.

The data boom of the last decade has propelled the development of **machine learning** methods that are not completely dependent on the programmer and can, in fact, evolve with each iteration by learning and predicting from the data itself [100]. These algorithms allow to pose different kinds of questions that do not need complete knowledge of the system or parameters to describe them. If there is no information about the data, the methods applied are unsupervised and try to find patterns and commonalities among the data. On the other hand, if there is *a priori* knowledge of the data (i.e., a ground truth), **supervised** methods can be applied.

Supervised methods can be used to understand whether a property of interest (in our case, the genotype) can be predicted via other properties (certain aspects in their images, such as microtubule disorganisation) [101]. The prediction rule (i.e. what defines the connection between properties and target) is deduced by teaching the

algorithms with a labelled training data set. After this deduction, new blind inputs can be provided to the algorithms for classification.

Furthermore, to train a classification model, it is necessary to have thousands of input images. Even though each experiment results in hundreds of neurons to be analysed, the amount of data necessary for classification is far higher. As such, it is important increase the number of input images without compromising the results. This can be accomplished with data augmentation methods.

To implement machine learning strategies during my project, I was accepted to participate in The Alan Turing Institute Enrichment Scheme of 2019, where I collaborated with Turing Fellow Dr. Iain Styles, whose expertise lies in applying computational methods to understanding biological data. The placement lasted from September 2019 to March 2020.

7.2 Methods

The Turing project was divided into two consecutive sections: Data Clean-up and Augmentation, and Classification, where the first had to be achieved to provide enough data for the second. Figure 7.1 shows the flowchart of the data processing precedent to the classification. For simplicity of methods due to time constraint, the images used were comprised only of the green (microtubule) channel.

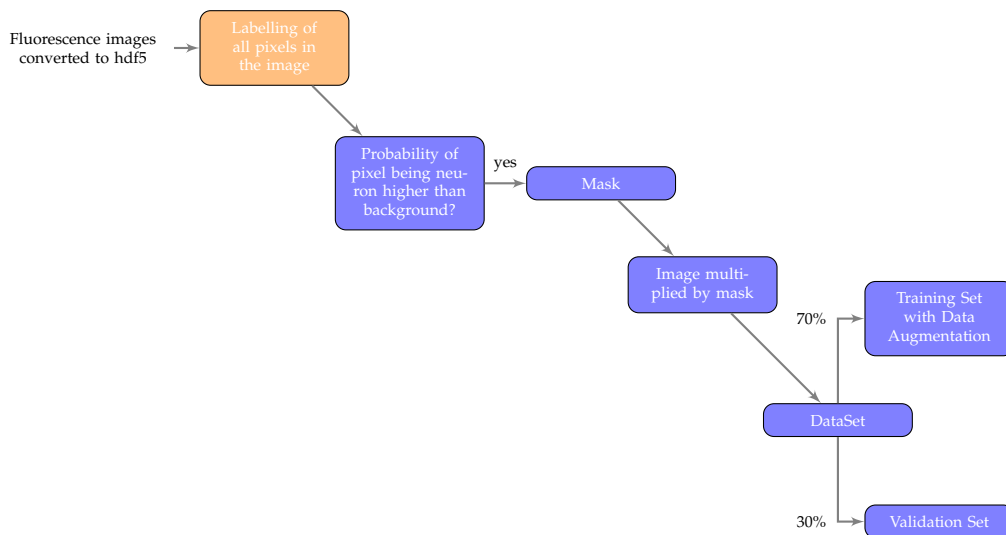


Figure 7.1: Flowchart of the pipeline created for the image processing before classification, using both Ilastik (orange) and MATLAB (blue).

7.2.1 Data Clean-Up and Augmentation

The way in which the primary neuron cultures are generated and stained, neurons are intermingled with other non-neuronal cell types and cellular debris [69] (Figure

7.2). In order to achieve an automated pipeline, it is pivotal to reliably identify the actual neurons in the images obtained.

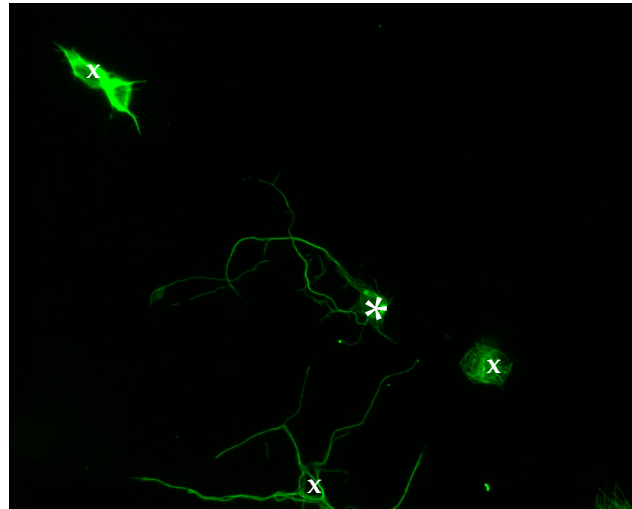


Figure 7.2: A cell culture derived from wild-type *Drosophila* embryos. Cells are stained with anti-tubulin (green) to label microtubules. The asterisk denotes the cell body of a neuron of interest that fulfils key criteria: it displays axons longer than the diameter of the cell body and fully visible (in contrast to the neuron at the bottom). x denotes all other irrelevant cells. Shown area of width $41.3\mu m$.

To develop neuron identification strategies, I used Ilastik [61], a program that is user-friendly and allows a quick deployment of a work-flow adapted to the images. The workflow most suited for my purposes was Pixel Classification, which produces a semantic segmentation of the images, i.e. classifies each pixel as either relevant or background.

The first step was to define the classes the image should be divided into, which were "neuron" (fulfilling all criteria, Figure 7.2) and "not-neuron" (unstained areas/debris that is not fulfilling key criteria). The ground truth annotation was manual and could be adapted in real time, as brushes allow the user to define which part of the image was relevant. Furthermore, each step could be backtracked if with the new learning stages, the classification was less-optimal.

The output of this program is, for each image, the probabilities of each pixel being part of a neuron or not which can be used to create a binary mask (on MATLAB), where all the pixels with a probability of being "neuron" superior to 50% are assigned a value of 1, and the rest are zeros (Figure 7.3b). When multiplying the original image (Figure 7.3a) by the mask, the result should be a clean image with only the neuron present (Figure 7.3c).

However, while it worked perfectly with the images annotated manually, the similarity between neurons and the surrounding cells was often cause for misclassification.

A further attempt was done with all three channels present in the classification,

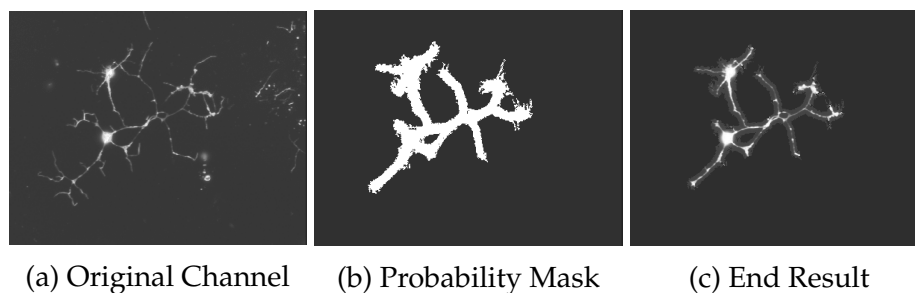


Figure 7.3: Images of a neuron in culture before (a) and after (c) application of the probability mask (b) obtained with Ilastik, to reduce the background noise.

but not only did it heavily increase the performance (as now there were three matrices per image, instead of one), it produced the same results with the same problems. As such, only the microtubule channel was used for classification. Further, even though the cleaning of the images was not ideal, it removed parts of the background that would be present otherwise. As such, the images chosen for the classification were processed through Ilastik.

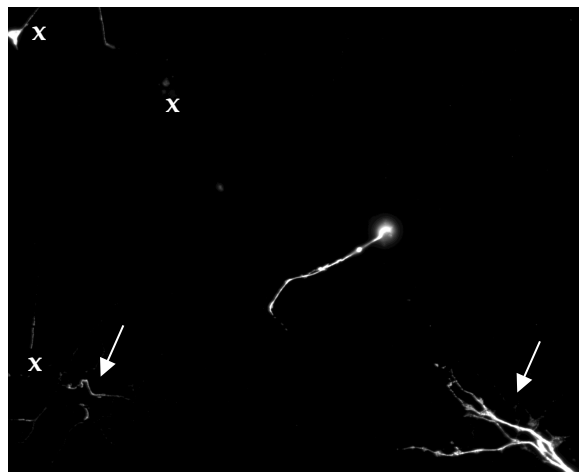
Before proceeding to the classification, the available dataset is still not sufficiently large. Further, in order to also prevent overfitting (the algorithms become extremely accurate for the training data, but fails when presented with new inputs), augmentation is necessary.

Augmentation is done by adding to the dataset slightly modified copies of the already available data. However, these modifications have to make sense given the type of data. For example, if one is considering images of cars, flipping them vertically would produce an upside-down car, which is not a viable solution. Rotations of all angles are possible with cell cultures, as the direction and orientation of neurons is irrelevant.

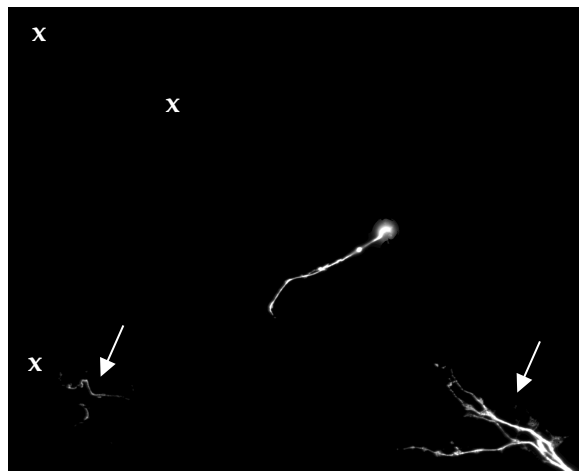
Table 7.1: Values used in the image augmenter for morphological transformations of the image dataset. For each transformation, a random value inside the presented ranges was chosen for individual input images. All transformations except rotation were applied to the x and y axes separately.

Transformation	Min	Max
Rotation	-90°	90°
X-axis Translation	-3	3
Y-axis Translation	-3	3
X-axis Shear	-90°	90°
Y-axis Shear	-90°	90°
X-axis Scale	0.5	4
Y-axis Scale	0.5	4

Finally, in order to proceed to classification, images have to be resized into the



(a)



(b)

Figure 7.4: The probability mask not always deletes all the background noise. Images of neurons before (a) and after (b) application of the probability mask obtained with Ilastik. Crosses denote the background that is not present in the second image, while arrows point to the ones that were not cleaned.

expected input sizes: 244x244 px. There were several attempts at classification using a simple resizing but, given the sizes of the images and how much background would be in each, a possible solution was splitting the image into same-size blocks and only analysing the ones with a certain number of viable pixels per block: >100 pixels in any block, or >10% and >30% of viable pixels in the block.

7.2.2 Classification

The main proponent of using supervised learning is that, by definition of the biological experiments, the data is already labelled as we know *a priori* which images belong to which genotype. In order to use as many images as possible, the experiment chosen has been repeated and includes three genotypes: wild-type, *khc*³ and *shot*³.

The dataset obtained with the clean-up is divided into a training set (complemented with augmented images) and a validation set, each used in the corresponding phase (Figure 7.5), 70% to 30% proportionality.

The training set is used by the algorithms to learn which images correspond to which labels - training phase (Figure 7.5a). After several iterations, the validation set is tested (Figure 7.5b) and the accuracy of the measurements with this set defines the accuracy of the overall pipeline.

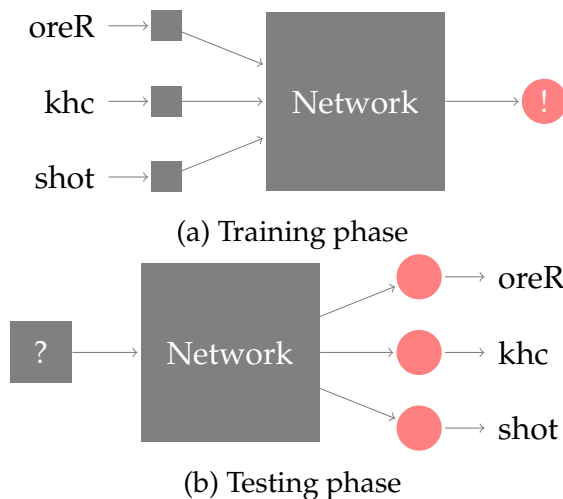


Figure 7.5: Schematics of the used classification architecture. During the training phase, labelled images are provided as input to the network that learns the predictive rule. In the testing phase, an unlabelled image is the input and the network will classify it into one of the learned categories.

But what does the network actually look like?

The most prominent network type in machine learning is a neural network (Figure 7.6).

When referring to feedforward neural networks, the information flows in one direction between the different layers: input layer (Figure 7.6, orange), hidden layer (Figure 7.6, blue) and output layer (Figure 7.6, red). In the training phase, labelled images are added to the input layer and the hidden layer will associate all the perceptible (to the algorithm) parameters seen on the images to the respective classification. In short, the hidden layer learns what distinguishes the labelled inputs.

However, a network can have several hidden layers. In this case, each layer receives the information already learned by the previous and propagates new knowledge forward. When there are multiple hidden layers, it becomes a deep neural network (Figure 7.7).

There is a specialised kind of neural network for processing grid-like data, particularly images: Convolutional Neural Networks, or CNN, which are networks that use convolution in at least one of their layers [102].

For images, and generally inputs that can be described as matrices, regular neural networks would share the inputs fully, i.e., the entire knowledge for the image.

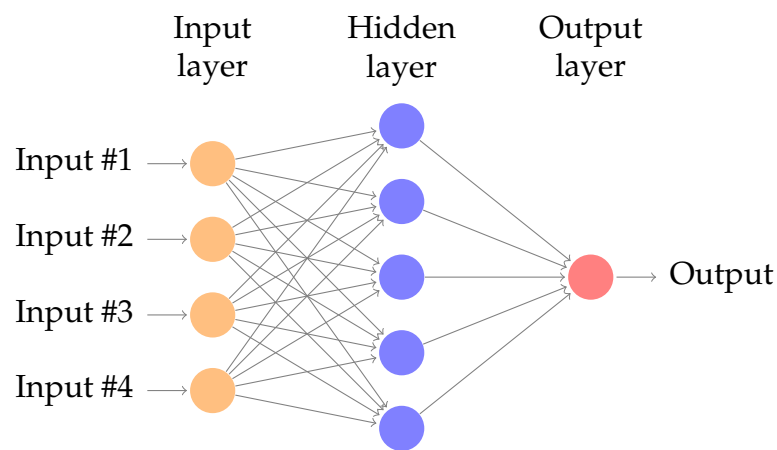


Figure 7.6: Schematics of a neural network architecture. There are three distinctive layers of "neurons" (circles) : input (orange), hidden (blue) and output (red). Arrows denote the direction of the information

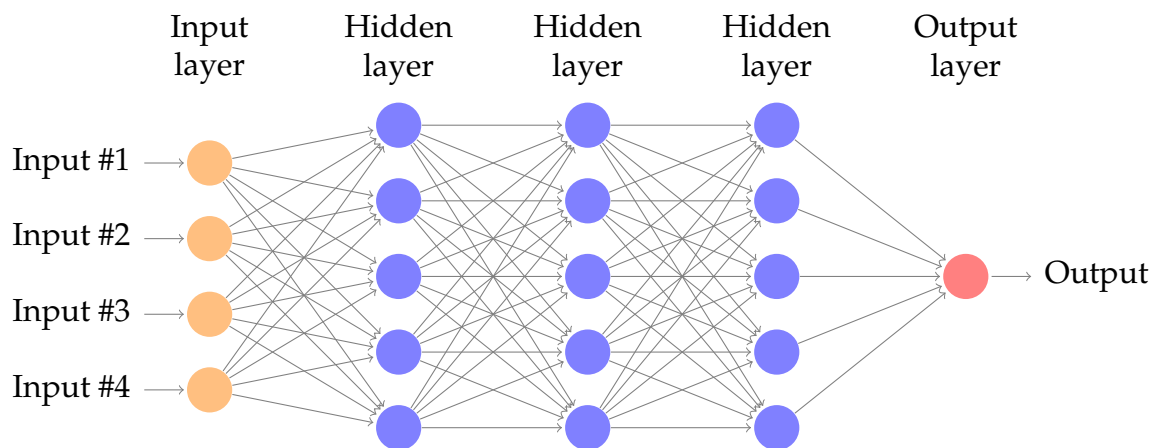


Figure 7.7: Schematics of a Deep Neural Network architecture. There are three types of layers: input (orange), hidden (blue) and output (red). The information flows in one direction, and each hidden layer learns with the previous.

However, with convolution (already referred in Chapter 3 for image filters) using small kernels, which decreases the information into smaller matrices. Further, given the weight of the information passed between layers, the parts of the image that do not provide any particular details will be dismissed. CNNs allow the detection of small and meaningful features in an otherwise large matrix.

The training of the convolutional layers is extensive, and computationally heavy as it has to learn how to interpret the input data. When it comes to image analysis, there are similarities between the data itself: for example, every RGB image is composed of pixels with certain intensities. As such, it is a futile exercise to train every network in how to detect image details such as colours, or image features such as edges. Consequently, the only layers that need to be trained to fit our dataset are the classification layers - the weighing of each node is adapted to the new, learned information. This process is called transfer learning. In image analysis, this translates into

networks previously trained on image feature extraction with a large image dataset.

The two types of CNNs used in this project were VGG16 and ResNet18, which are available MATLAB and pre-trained with the ImageNet [60] dataset.

VGG16 (Figure 7.8) [103] contains a total of 16 learning layers, out of which 13 are convolutional (Figure 7.8 dark gray), and the remaining 3 are fully connected layers (Figure 7.8, orange), which will feed the classification into the output.

The light gray dashed layers correspond to maxpooling layers, where the matrix has a size reduction to half by choosing the maximum value of each four elements. With the input image starting at 244×244 , the final layers receive a 7×7 input. These are denominated fully connected as they receive information from all the nodes in the previous layer and will associate the image features obtained to the intended classification.

Not depicted are auxiliary layers to prevent overfitting.

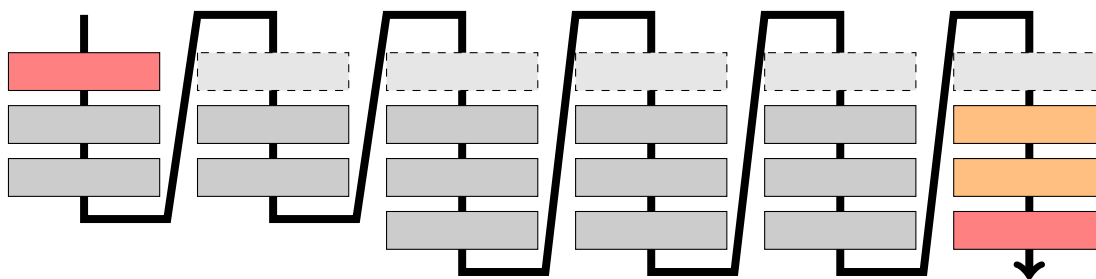


Figure 7.8: Schematics of a VGG16 architecture. Red layers are input and output, dark gray correspond to the 13 convolutional layers, dashed light gray correspond to pooling layers, orange are the fully connected layers. Adapted from [104].

The second network architecture used, ResNet18 [105], has a similar structure to VGG16 but has two extra layers and allows residual information to inform latter layers (Figure 7.9 blue). As such, small features (details that would be lost in the downsizing of image matrices) are propagated to latter layers.

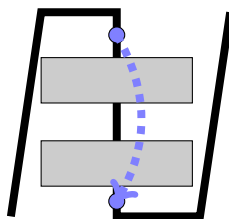


Figure 7.9: Schematics for the ResNet18 building block. Gray represents the convolutional layers, while blue represents the points between which residual information is propagated forward.

The last fully connected layer specified the classes in which to label the images. In both cases, the number was three: wild-type, shot3 and khc8.

In order to test the networks, an exploratory data analysis (EDA) was performed. Images were tested before and after being processed through ilastik, the size of the

blocks in which the image was split varied, and number of viable pixels. Further, the percentage of training/validation sets was changed which could inform whether a higher number of labelled data was needed.

MATLAB provides a user-interface to observe the training, where accuracy (the percentage of correct classifications in the validation phase) and loss (function to be minimised) are plotted per learning iteration. After going through all the epochs¹, the accuracy can be compared.

7.3 Classification Results

The tests were run during 30 epochs. Figure 7.10 shows an example of an interface for the network training in MATLAB.

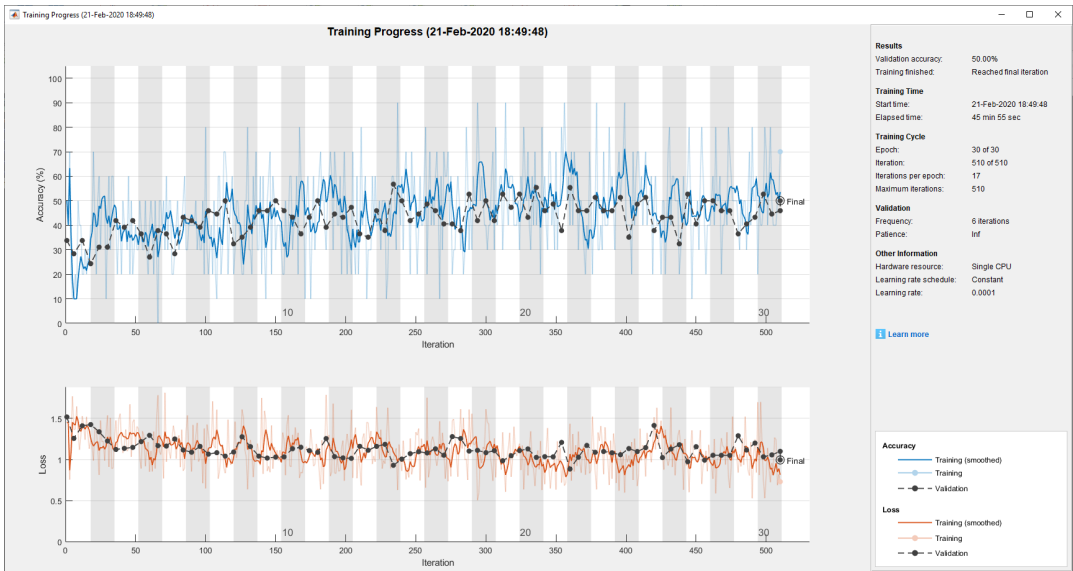


Figure 7.10: MATLAB user interface for network training, via Deep Learning Toolbox. The top graph shows the accuracy per iteration, the lower graph shows the loss per iteration. The right panel shows the evolution of results, training cycle and CPU usage.

With these interfaces, it is possible to keep track of the training evolution. The accuracy (Figure 7.10, top graph) and loss function (Figure 7.10, bottom graph) are calculated for each iteration and, ideally, accuracy would be at a high percentage while loss would be close to zero. In all the cases, loss was always around one and so the depicted value is the accuracy.

Further, the interface presents a panel with extra information such as the training cycle steps, CPU usage and time elapsed.

Table 7.2 shows the different combinations of dataset division, block size and how many viable pixels are necessary to consider a block to be valid as described in Section 7.2.1. The validation frequency is a parameter of the classification that performs

¹Epochs are the number of times the algorithm goes through the entire training dataset, there are several iterations per epoch.

validation after a number of iterations of training (in these cases, either 3 or 6).

Table 7.2: List of parameters explored and the average accuracy to two decimal places. The dataset % corresponds to the split between training and validation sets, block size applies when the images were individually split into blocks before being classification. Validation was performed every 3 or 6 iterations (Validation frequency).

Network	Dataset %	Block Size	Validation Freq	Viable Pixels	Accuracy %
VGG16	70/30	Full Image	6	N/A	13.4
VGG16	90/10	Full Image	6	N/A	14.1
VGG16	90/10	Full Image	3	N/A	16.8
ResNet18	70/30	Full Image	6	N/A	33.2
ResNet18	80/20	Full Image	6	N/A	39.9
ResNet18	90/10	Full Image	6	N/A	51.6
ResNet18	70/30	448x448	6	>30%	50.0
ResNet18	70/30	896x896	6	>30%	49.7
ResNet18	70/30	224x224	6	>30%	50.4
ResNet18	70/30	224x224	6	>10%	48.5
ResNet18	70/30	224x224	6	>100	35.5

The maximum value for the VGG16 classification accuracy was below 17%. Even with a large training set and risking over-fitting, the values remained low. With the introduction of ResNet, the value improved significantly, albeit not to a good classification standard (higher than 80%). Although there was a maximum of 67% when the percentage of training data was 90 to 10, the average was still $\approx 50\%$.

In practice, the classification between the three genotypes had such a low accuracy that no conclusions can be withdrawn.

As shown in Section 7.2.1, the images presented a number of image artefacts through an incomplete clean-up in Ilastik. This could translate into consideration of elements in the image that are not necessarily the neuron. Ideally, the classifier would be trained in the relevant region - the neurons.

The reduction of the images from 1376x1083 to 224x224 pixels may cause a high level of information loss due to the small details present in disorganisation areas. Given the increase of accuracy in ResNet, where the propagation of small details increases, it shows that the images might be too big for the classifier to properly distinguish them.

Chapter 8

Conclusions, discussion and future

8.1 Main Outcomes

8.1.1 Aim of the Thesis

Understanding how axons are maintained is an important step towards mechanistic insights into neuropathies [4]. One of the phenotypes present in such diseases is axon swellings: while axonal microtubules are normally arranged in parallel, mostly straight bundles, in the swellings they appear disorganised and forming loops. As explained in Section 1.3, the Local Axon Homoeostasis hypothesis [11, 14] states that microtubules would have an intrinsic tendency to get disorganised but are guided into parallel bundles by outside forces and additional proteins. These gave rise to the idea that the curling phenotypes could be quantitatively similar between mutant conditions in proteins of different pathways.

The biological key question behind this project is to identify whether areas of disorganisation differ between different genetic conditions or are similar. In order to lay the foundation to address this, the aim of the project was to (1) automate the image processing and analysis pipeline and (2) retrieve two new parameters to describe microtubules in image datasets: straightness of microtubule bundles/axons and curvature of microtubules within areas of disorganisation.

The three phenotypes chosen to analyse belong to three different pathways: kinesin (khc), a molecular motor transporting cargoes along microtubules; efa6, a factor preventing microtubule polymerisation and shot, a protein that guides microtubules parallel to the plasma membrane into bundles.

8.1.2 Image Processing and Analysis

There are a number of challenges when imaging of areas of microtubule disorganisation: microtubule disorganisation can be easily identified using epifluorescence, which is the most convenient, cheapest and quickest imaging method available in the laboratory. However, microtubule bundles and other fine microtubule structures

cannot be resolved into individual microtubules, or resolved as such - a single microtubule is thinner than the resolution limit of visual light. At this point it is also not clear if the observed structures in areas of disorganisation represent single microtubules or smaller subbundles of microtubules. This means that microtubules would need to be imaged with super-resolution methods (e.g. gSTED) to resolve ultrafine structures or ways would need to be found to get meaningful information from sub-resolution epifluorescence images. As the majority of the available images are epifluorescence images, the latter strategy was followed.

The first approach was to check whether the available software packaged at the Bioimaging Facility, University of Manchester, or free versions online provided the necessary functionality to retrieve the parameter we were most interested in - the curvature of the curling microtubules. However, none of them had the tool or even an algorithm available at the time - the development of a computational algorithm from the theoretical designs of curvature retrieval was necessary.

Analysing images is the process of retrieving information from the intensity matrices. Given the discrete nature of these, even the simple act of capturing an real scene onto an image (with any camera, from microscopes to phones) forces continuous, real values onto a grid.

From image to skeletonised shape

One way to accurately describe and retrieve the shape of such thin lines can be through the retrieval of its skeleton, the pixel-wide centred contour of the lines. The first part of the project was to transform the images from their RGB files into a binary skeleton.

First, finding an appropriate filter was pivotal to enhance the tube-like features (microtubule fluorescence images are projections of 3D "tubes") and depreciate the background. Applying the function *vesselness2D* to the microtubule channel allowed a better recognition of the lines, rather than directly applying a threshold to build the binary mask.

While this method is effective, the experimental conditions are not ideal and there could be gaps in the microtubules either as phenotype or because fixation did not work. Some improvement can be achieved by adding a predictive method that evaluates the probability of a certain pixel belonging to a tube, if the neighbours have a high certainty of being one.

From the enhanced image, a threshold is applied to retrieve the binary mask. At this stage, the user chooses the pixel intensity above which will consider the relevant parts of the image. This gives a choice to fine-tune both the virtualisation and mask construction in order to optimise the skeleton retrieval. While the user has to have some input on this stage, it would be ideal to find how to connect the threshold of the mask to the values of the monoscale image provided by the *vesselness* function,

while still allowing the user to fine tune the parameters. This would provide a first binary mask that conserves a certain percentage of pixels based on the distribution of monoscale intensities. These would be good indicators as to whether the mask needs to be more permissive (allow a higher percentage of values to be considered relevant) or the virtualisation to be repeated with a different intensity.

The next step was finding an appropriate method to retrieve the skeleton from the binary mask. It was important to keep certain conditions in mind: appropriate shape recognition, maintenance of Euler number and time. The functions tested had different algorithms and neighbouring considerations, which provided varying results. In the end, the function chosen was *bwmorph* as it was the quickest and took the most conservative approach by considering more points and bridges between parts of the image that separated by a couple of pixels in the binary mask. As mentioned above, the resolution cap will always create artefacts in the images - if the skeleton can minimise those, it performs better in delicate and small structures like microtubule disorganisation.

Skeletonisation is a method that can further be improved by applying learning techniques: for example, by having a user manually trace the images and use them as ground-truth for a skeleton-generating algorithm. This would be a time consuming task for the initial annotation. However, having functions generating skeletons (e.g. the ones mentioned in this project, *bwskel* and *bwmorph*), and the user then choosing the correct solutions could potentially lower the number of annotations (choose the closest skeleton instead of tracing the entire shape).

While the skeleton provides information on the shape, any intensity measurement is lost. Ideally, this could be considered further down the pipeline. For example, by tracking the intensity surrounding each skeleton pixel and analysing the distribution along the entire shape. The intensity comparisons would bias the skeleton towards the brighter pixels, which generally indicates a higher fluorescent signal - in this case, it would correspond to the main axon rather than a secondary branch.

Path recognition

Intensity is the value that traditional path recognition techniques recognise as the next possible point - by following to the closest neighbour with the highest intensity. However, given its extensive 8-connectivity neighbour search, this method can be very time consuming, especially with larger images, or with high levels of noise. Furthermore, intensity methods do not take advantage of the skeleton, as they would struggle with all pixels having either 1 or 0 intensity. These methods usually only have one input, the start point, and require the user to continuously monitor whether the path detection is correct.

Path recognition is important because generally we are only interested in a part of

the skeleton, in this case for two types of regions: axons and microtubule disorganisation.

This translates into the region of interest (for disorganisation) or the path (one single connected row) being all the pixels of interest between two end points. While there is a need for an extra initial input by the user, having the end point also allows different ways of searching. A quick mathematical way is by using graphs - where nodes are connected by edges, and a path is the connected nodes between the two ends. Each node in a graph is saved with its neighbours, which computationally translates into only searching the relevant points, reducing the search space.

There are functions to transform binary images into graphs: each pixel with value 1 will be considered a node, and will have edges to their 8-connectivity node neighbours. However, edges provide a weight to the path (i.e. the path is the shortest sum of weights between two points). If the edge construction had also intensity as a consideration, perhaps the path finding would be more accurate.

For the axon recognition, even in the presence of disorganisation regions, the path chosen is the shortest possible, crossing through the region, as this was the methodology used for the ImageJ analysis. For microtubule disorganisation recognition, all points connected between the two ends are considered of interest. This, however, causes higher discrepancies with manually picked paths, as every intricacy of the path and disorganisation is considered and not approximated. In reality, this translates into a more accurate description of the pixels in the image.

While these algorithms work for perfectly connected networks (i.e. each node has at least one path to any other node), the biological images will present several artefacts: gaps, other branches connecting the ends of the axon that are not part of the core axonal structure.

In the case of a gap that prevents the network from being fully connected, there are two strategies: either the user adds a line to complete the path, or selects one of the nodes surrounding the gap. Furthermore, the search for the closest node to the current recognised path could be automated if not for the edge case images where the closest connected point is still in the wrong path - which would need further input and corrections from the user. Finally, for axon recognition, if there is more than one possible path between the two ends, and the shortest one is not the correct, the user can add as many points as necessary to make sure the path is corrected.

If there were no time constraints on the method, there could be a radial search until it found the other end point to automatically cross the gap. However, for larger images this would mean a higher computational toll.

For microtubule disorganisation recognition, there are extra user inputs that can be added to remove unwanted branches from the region (such as small protrusions).

Analysis

Two important measurements can be retrieved just with the path recognition: axonal length (number of graph nodes, i.e. pixels, between cell body and the axon terminal), and disorganisation area (area enclosed by the other line of the region). However, the geometric description of the shapes is necessary, so straightness ratio and curvature require extra calculations.

The general method for straight segment calculation is based on the distances between the user clicks. By removing these, it is necessary to retrieve the geometrical description of the shape with other methods. The most widely used method is a Hough Transform, where the coordinates are parametrised according to the geometrical shape that one wants to retrieve. In this case, the shape is a simple straight line. The advantage is reducing any possible search into just one operation: the transformation into the parameter space. This reduces the time and computational resources needed, as well as providing an accurate description of the image. The Hough Transform method was applied recursively to the path: the algorithm will search the path until it finds the maximum number of peaks, or until the path has been saturated. While in theory it should work, the transform is heavily influenced by the angle in which the line is presented.

Having the straight line rotated in different angles will produce different results (Figure 4.10c). For example, diagonal lines in a binary matrix correspond to small straight ladders, rather than one long segment. This will produce a high number of small straight segments, rather than just a long one. The optimum result could be obtained if each skeleton is rotated 90° and the transform calculated for each rotation. However, this would be very time consuming, and performance heavy as it would require an increasingly larger amount of calculations.

Furthermore, the number of peaks in the Hough Transform space (i.e. the number of the highest frequency of pixels in the straight lines considered) is decided *a priori*, and as of the current version, the user can not change it. One way of having an adaptive number of peaks is by taking the number of straight segments and their length into account, as the calculations are happening. If all the straight segments found are roughly the same size, then potentially there are going to be more straight segments of that size in the image, so there is a need for a larger number of peaks. On the other hand, if the first segments are much larger than the last ones, then the search would stop. While it would improve the performance in most cases, large images with small segments would have a much longer search. This method provides a good comparison method between different phenotypes, but needs extra adjustments to accurately describe the straightness of real images.

The final calculated value is the curvature. The theoretical retrieval of curvature from images has been discussed for decades, but the implementations appear to suffer with the nature of the images themselves (discrete matrices). The most common

approaches involve the calculation of the curvature via the pixels themselves. However, given the resolution problems mentioned, these are not reliable methods. Fitting functions to the lines is less resolution dependent. There are several approaches. Fitting a function to the entire shape (whether it is polynomial function or Fourier transform), but these only provide a general curvature value per image. Furthermore, polynomials would have to be of higher degree (20+), which would introduce artefactual oscillations to the function. Using a Gaussian filter with a Gaussian window can provide a good fit with individual curvature values for each point in the line. However, there is the added caveat that the window size would be fixed per image: the correct window size, that captures the intricate details in one image, might calculate only a blur on the next. Finally, the method that seemed the most appropriate, is the Smoothing Spline fitting.

Using the disorganisation region selection, all cross points are removed (to avoid false curvatures), the resulting segments are split into the same-size splines and to each, a polynomial of third degree is applied.

In the synthetic images, chosen for their analytical curvature calculations (logarithmic spiral, where the curvature is proportional to the radius, and an ellipse, with a periodic curvature), the values obtained with the splines method are the closest approximation to the expected values from all the testing. Interestingly, a similar method has been recently developed [106] by applying a different type of function (Bezier curves, where a curve is defined by the end points and auxiliary points that act as magnets to the line, attracting the curve in their direction) to the segments.

8.1.3 ALFRED

While all the algorithms by themselves have been developed and tested, the entire pipeline is bundled in a MATLAB software package, ALFRED.

Every window was carefully chosen and tailored, with constant feedback from the user side. The presence of three different levels of windows correspond to the different levels of the image: full set of images, individual image and region of interest. The type of analysis performed in each window is also applicable to that level: all images are loaded in the first window, the whole image is processed in the second window (and the process is replicated in all images), each individual region is treated separately for its path recognition. Furthermore, if one of the panels malfunctions, the program continues to run and no progress is lost. This guarantees a visible consecutive flow, as well as robustness to unpredictable errors.

The order in which the buttons become available to the user corresponds to the functional pipeline: no button becomes available before it can be used (for example, a binary mask can not be obtained from the full RGB image). This is another robustness measure, as it prevents the program from crashing due to unexpected interactions. Furthermore, it helps the user with the flow of the analysis itself, preventing any step

being forgotten.

When analysing the regions individually, while the user sees the skeleton on the gray-scale image, all the calculations are happening on a graph level. Whenever the user clicks on the image, the program finds the closest possible node to the click (using Euclidean distance). Each of the mentioned alterations to the above algorithms (i.e. any user interaction or input such as extra clicks) are seamlessly in the software - the user only has to work with the interface.

The underlying mathematical principles could be implemented in different programming languages, but the calculations and flow are unique to ALFRED. Furthermore, certain parameters, such as the automated Microtubule Disorganisation Index calculation, are a characteristic of the software. Extra features are implemented as a consequence: by verifying first bounding box overlap, and then the actual skeleton overlap, the software guarantees that for each axon it finds all of the selected disorganisation regions, without any further input from the user. This allows the calculations to be automated, and the user receives the values at the end, without any manual gathering of data as is done currently (the index is calculated by organising the results manually).

The program was built to be used by anyone without programming experience. Furthermore, all algorithms will be published, and the code is freely available on Github [98] - this allows users from other languages to implement the algorithms in their respective platforms, and contributes to open source science. Finally, there will be a full user-manual available, explaining in detail how to use the program, and what each individual function does.

The software is, however, highly specific to the data gathered in the group. As the interface was optimised for the pipeline, it does not provide different features, such as localisation of parts of the image in sequences of images (i.e., tracking of cells per time frame), or intensity distributions. Furthermore, as it was developed in MATLAB, it is licensed and therefore not freely available to anyone. While the methods are modular, the implementation might not be trivial to non-programming users. Ideally, the algorithms in ALFRED should be adapted into plug-ins for more complex image analysis software (FIJI, or in Python, Napari [107]), using open-access languages and becoming available to a wider community.

8.1.4 Biological Interpretations from ALFRED

After the pipeline being built and the usability and results tested, the next key question can be addressed: are the disorganisation phenotypes quantitatively similar?

Before any considerations for curvature and straightness, it is important to confirm that the values obtained by ALFRED correspond appropriately to the manual analysis in length and disorganisation area. These values, however, could never be

the same, as the path recognised in ALFRED follow the axons more closely, especially in curved regions, than manual analysis with segmented lines can. Disorganisation carries another caveat: the extra curves will provide extra areas to the total (i.e. squared pixels), which will further enhance the differences.

To infer the similarities with the original ImageJ-obtained data, it is necessary to perform non-parametric statistical tests (as the distributions are independent). The main comparison value is going to be the probability that any difference between the two datasets is random, rather than systematic - p-value. The results are encouraging: in both length and disorganisation area, the p-value obtained by comparing the values of each dataset (ALFRED vs Manual) is above the threshold of 5% (value above which it is determined as random) for every genotype comparison.

Furthermore, even with ALFRED axonal length and disorganisation area values higher than the manual counterpart, the ALFRED-calculated value of the microtubule disorganisation index is not significantly different from the ImageJ-derived one. These phenotypes have been extensively studied and help understand the consequences of the different mutations in the *Drosophila* neurons.

Further analysis is needed, as length and area are not indication of the geometrical characteristics of the disorganisation itself.

Curvature can be used as the geometrical description of looped shapes, such as the ones present in disorganisation.

Given the nature of the spline method (each part of the skeleton was divided into same several smaller splines), the number of curvature points calculated was in the millions. As such, the time it takes to run is less than ideal - which could be improved with parallel processing by calculating several curvatures at the same time. Without knowing the actual measurements, the curvature values obtained are purely comparative between phenotypes within the same experiment. Manual acquisition of the radius in the image would also not provide an accurate comparison, as these are subjective: the circles that a human eye can see are not necessarily present as the images are 2D projections of 3D structures - the line continuation information is lost. On the other hand, the ALFRED calculations are generated per spline and would, at best, provide a range of values for the entire image.

Furthermore, there is always going to be noise in the curvature distribution as a lot of the smaller segments appear straight (curvature value of 0).

Another phenotype geometrical characterisation is by evaluating the axonal straightness, as the length does not provide the full picture as to how the axons have grown. For example, neurons under higher levels of tension would have straighter axons.

The straightness ratio calculations had results above 1, which would indicate that the sum of the segments is larger than the total length of the axon. There are different possible motives. Assuming the Hough Transform worked perfectly, the length

calculation might have been done erroneously due to the conversions from images to graphs, as the transforms are performed in the binary mask, while the length is calculated with the graph.

On the other hand, if the length calculation is correct (which is probable, given the similarity of the results with the ImageJ-derived analysis), then there might be some error with the Hough Transform conversions, such as approximation of straight segments (to be lenient to small angle changes) influencing the values, or the recursive algorithm not stopping at the correct number of peaks. To pin point the error, several runs would be necessary with the same image but, for example, rotated. If the values vary with the rotation, it is likely a Hough Transform problem, as mentioned above.

Notwithstanding, using the two different methods of measuring geometry the results appear to be that *shot*³ has a slightly different geometry to both *khc*^{27/27} and *efa6*^{ko}: the average curvature appears to be lower (with the distribution more isolated in Figure 6.9c, corresponding to larger loops in the disorganisation). Furthermore, regarding the axonal length and straightness, *shot*³ has also a lower distribution of values (the axon itself appears to be straighter).

The results are still consistent with the Local Axon Homoeostasis hypothesis. The wild-type results are generally different in disorganisation regions, being present in a much lower number of times (half or even a third of occurrences of the other phenotypes) and, when present, the curvature is higher - corresponding to lower radius and, consequently, smaller loops. Given how different the pathways of *khc* and *efa6* are, a similar geometry indicates that the curling of the microtubules is likely the same in both disorganisations. Furthermore, the small yet present variation with the *shot* phenotype could indicate the presence of another element at play that further influences the disorganisation of microtubules, even if just as a consequence of the mutation to *shot*, a guidance protein. However, this variation is far too small to draw any further conclusions.

The small differences observed need to be further explored, with more data. Other pathways have to be tested (mutations in other auxiliary proteins such as tau, or XMAP215/Msps), or even within different mutations of the same gene (comparison of phenotypes between different kinesin mutations).

Furthermore, the quantitative values could inform a mathematical model of axonal disorganisation. With this, parameters of biological experiments can be scrutinised in different ways, or even include the testing of parameters that would be otherwise impossible to study. In addition, simulations *in silico* can make predictions about the most relevant parameters to be regarded in the experiments. An accurate computational model of microtubules present in an axonal environment can also simulate physiological conditions for longer periods of time, to study the normal decaying of microtubules over the lifetime of the organisms. A symbiotic relationship

of simulated data and biological results can further inform the hypotheses, and focus the experimental time in parameters that are of interest.

8.1.5 Machine Learning

Regarding the machine learning applications, there were two main goals: Automated region selection and classification of phenotypes to compare between mutations in *shot* and *khc*. Furthermore, if any of these applications were successful, they were to be implemented into ALFRED.

The biggest setback for this project was the insufficient available data. While there are thousands of images from all different experiments, the conditions in which these were done are different: from genotypes used, to environmental conditions and days *in vitro*. As such, the hundreds of images produced in one experiment and its repeat (i.e. experiments with the same conditions, performed at different times) are not sufficient to accurately train models. Data augmentation can increase the number of input images, but might not be diverse enough to create an appropriate training and validation dataset. The experimental setting chosen, with *shot* and *khc*, allow the study of the two different pathways and has been performed more than once, increasing the data available.

Segmentation was partially successful - while there are a group of images that have been segmented correctly, a large majority still contained debris from other cells, or neurons that are not of interest. This remains a challenge in computer vision: how to ignore parts of the image that fit and correlate to the characteristics that define the regions of interest, albeit are subjectively not relevant - i.e. a human can decide which detected neuron is of interest, and which is to discard, while the program only detects two neurons. A negative training set could be introduced by eliminating cells that have only axon but lack cell body, or round- or oval-shaped cells. With *ilastik*, this can be implemented with the Object Classification workflow [108], which builds on the pixel classification.

The classification step, however, was not successful. The CNNs did not work: the images either are reduced from 1376x1083 to 224x224 and lose details that are needed, or are divided into blocks and do not represent correctly the neurons. As ResNet18 provided better results, perhaps a pre-trained network with a higher number of hidden layers (such as ResNet50) could be better, as this would increase the weights and consequently the details considered on the decision.

All the methods tried were supervised. A different approach, with an unsupervised method might also be a consideration: trying to cluster the images into different classes, without any *a priori* information, and analysing whether these clusters correspond to individual genotypes. However, images are large and have a high dimensionality for such little number of classes - the clustering would likely not be able to distinguish between the phenotypes.

The next step would be to train the classification layers only on the regions selected by ALFRED. This way, the user has already annotated what smaller parts of the image are of interest for each genotype, and the dimensionality problems are reduced. Furthermore, the clustering approach mentioned in the previous paragraph could also be applied to this new dataset.

As the classification step was not successful, no biological conclusions could be retrieved from the machine learning analysis.

8.1.6 Final Remarks

The overall goal of the thesis was to speed up and find a way to require fewer user inputs to the current analysis of the fluorescence images produced by my host group. In addition, the goal was also to find a way to measure two new parameters, axon straightness and curvature of microtubules in areas of disorganisation.

ALFRED provides a robust, reproducible, user-friendly, accurate pipeline to analyse phenotypes in a timely manner, with the new geometrical characteristics being retrieved. The analysis performed has less user input and provides results in accordance to the ImageJ-derived values obtained thus far. Furthermore, the geometrical results show similarities between *efa6* and *khc*, with small differences to shot. This might be a step towards obtaining quantitative evidence to sustain the Local Axon Homoeostasis hypothesis. The software is ready to be used to further analyse other genotypes or experimental conditions, for users with access to MATLAB.

The next step in the development of the software includes adding support for time series, to analyse live imaging and further understand the mechanisms of axonal swelling over time; data visualisation at the end, not just calculation and saving into a file, to allow the user to see the data in real time and adjust the calculations if needed and loading of previously saved analysis and processing.

As for the machine learning methods, as more data is generated, the datasets might approach a size that would be auspicious for classification. The most advantageous solution for a short term approach would be to save the regions selected by ALFRED and use these as the input for the learning methods. Furthermore, the next step is in finding the appropriate segmentation method that will distinguish the different parts within a neuron: cell body, axon and microtubule disorganisation. This way, the classification could be more specific to the objects, rather than the full images.

Bibliography

1. Prokop, A. Cytoskeletal organization of axons in vertebrates and invertebrates. *Journal of Cell Biology* **219**, e201912081. ISSN: 0021-9525. <https://doi.org/10.1083/jcb.201912081> (May 2020).
2. Marner, L., Nyengaard, J. R., Tang, Y. & Pakkenberg, B. Marked loss of myelinated nerve fibers in the human brain with age. *Journal of Comparative Neurology* **462**, 144–152. <https://onlinelibrary.wiley.com/doi/abs/10.1002/cne.10714> (2003).
3. Adalbert, R. & Coleman, M. P. Review: Axon pathology in age-related neurodegenerative disorders. *Neuropathol Appl Neurobiol* **39**, 90–108. ISSN: 1365-2990 (Electronic) 0305-1846 (Linking). <https://www.ncbi.nlm.nih.gov/pubmed/23046254> (2013).
4. Prokop, A. A common theme for axonopathies? The dependency cycle of local axon homeostasis. *Cytoskeleton* **78**, 52–63. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/cm.21657>. <https://onlinelibrary.wiley.com/doi/abs/10.1002/cm.21657> (2021).
5. Bradke, F., Fawcett, J. W. & Spira, M. E. Assembly of a new growth cone after axotomy: the precursor to axon regeneration. *Nat Rev Neurosci* **13**, 183–93. ISSN: 1471-003x (2012).
6. Curcio, M. & Bradke, F. Axon Regeneration in the Central Nervous System: Facing the Challenges from the Inside. *Annu Rev Cell Dev Biol* **34**, 495–521. ISSN: 1081-0706 (2018).
7. Pisciotta, C. & Shy, M. E. Neuropathy. *Handb Clin Neurol* **148**, 653–665. ISSN: 0072-9752 (Print) 0072-9752 (2018).
8. Kevenaar, J. T. & Hoogenraad, C. C. The axonal cytoskeleton: from organization to function. *Front Mol Neurosci* **8**, 44. ISSN: 1662-5099 (Print) 1662-5099 (Linking). <https://www.ncbi.nlm.nih.gov/pubmed/26321907> (2015).
9. Conde, C. & Caceres, A. Microtubule assembly, organization and dynamics in axons and dendrites. *Nat Rev Neurosci* **10**, 319–32. ISSN: 1471-0048 (Electronic) 1471-003X (Linking). <https://www.ncbi.nlm.nih.gov/pubmed/19377501> (2009).

10. Akhmanova, A. & Steinmetz, M. O. Tracking the ends: a dynamic protein network controls the fate of microtubule tips. *Nature Reviews Molecular Cell Biology* **9**, 309–322. ISSN: 1471-0080. <https://doi.org/10.1038/nrm2369> (2008).
11. Hahn, I., Voelzmann, A., Liew, Y.-T., Costa-Gomes, B. & Prokop, A. The model of local axon homeostasis - explaining the role and regulation of microtubule bundles in axon maintenance and pathology. *Neural Development* **14**, 11. ISSN: 1749-8104. <https://doi.org/10.1186/s13064-019-0134-0> (2019).
12. Mitchison, T. & Kirschner, M. Dynamic instability of microtubule growth. *Nature* **312**, 237–42. ISSN: 0028-0836 (Print) 0028-0836 (Linking). <http://www.ncbi.nlm.nih.gov/pubmed/6504138> (1984).
13. Matamoros, A. J. & Baas, P. W. Microtubules in health and degenerative disease of the nervous system. *Brain Res Bull* **126**, 217–225. ISSN: 1873-2747 (Electronic) 0361-9230 (Linking). <https://www.ncbi.nlm.nih.gov/pubmed/27365230> (2016).
14. Voelzmann, A., Hahn, I., Pearce, S. P., Sanchez-Soriano, N. & Prokop, A. A conceptual view at microtubule plus end dynamics in neuronal axons. *Brain Res Bull* **126**, 226–237. ISSN: 1873-2747 (Electronic) 0361-9230 (Linking). <https://www.ncbi.nlm.nih.gov/pubmed/27530065> (2016).
15. Prokop, A. The intricate relationship between microtubules and their associated motor proteins during axon growth and maintenance. *Neural Dev* **8**, 17. ISSN: 1749-8104 (Electronic) 1749-8104 (Linking). <http://www.ncbi.nlm.nih.gov/pubmed/24010872><https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3846809/pdf/1749-8104-8-17.pdf> (2013).
16. Datar, A. *et al.* The Roles of Microtubules and Membrane Tension in Axonal Beading, Retraction, and Atrophy. *Biophysical Journal* **117**, 880–891. ISSN: 0006-3495. <https://doi.org/10.1016/j.bpj.2019.07.046> (2019).
17. Hellal, F. *et al.* Microtubule stabilization reduces scarring and causes axon regeneration after spinal cord injury. *Science* **331**, 928–31. ISSN: 1095-9203 (Electronic) 0036-8075 (Linking). <http://www.ncbi.nlm.nih.gov/pubmed/21273450> (2011).
18. Ruschel, J. *et al.* Axonal regeneration. Systemic administration of epothilone B promotes axon regeneration after spinal cord injury. *Science* **348**, 347–52. ISSN: 1095-9203 (Electronic) 0036-8075 (Linking). <http://www.ncbi.nlm.nih.gov/pubmed/25765066> (2015).
19. Lu, W., Lakonishok, M. & Gelfand, V. I. Kinesin-1-powered microtubule sliding initiates axonal regeneration in *Drosophila* cultured neurons. *Mol Biol Cell* **26**, 1296–307. ISSN: 1939-4586 (Electronic) 1059-1524 (Linking). <http://www.ncbi.nlm.nih.gov/pubmed/25657321> (2015).

20. Bernier, G. & Kothary, R. Prenatal onset of Axonopathy in Dystonia musculorum mice. *Developmental Genetics* **22**, 160–168 (1998).
21. Hurd, D. D. & Saxton, W. M. Kinesin mutations cause motor neuron disease phenotypes by disrupting fast axonal transport in *Drosophila*. *Genetics* **144**, 1075–85. ISSN: 0016-6731 (1996).
22. Tarrade, A. *et al.* A mutation of spastin is responsible for swellings and impairment of transport in a region of axon characterized by changes in microtubule composition. *Hum Mol Genet* **15**, 3544–58. ISSN: 0964-6906 (2006).
23. Liu, L., Tuzel, E. & Ross, J. L. Loop formation of microtubules during gliding at high density. *J Phys Condens Matter* **23**, 374104. ISSN: 1361-648X (Electronic) 0953-8984 (Linking). <https://www.ncbi.nlm.nih.gov/pubmed/21862840> (2011).
24. Gupta, K. K., Alberico, E. O., Nathke, I. S. & Goodson, H. V. Promoting microtubule assembly: A hypothesis for the functional significance of the +TIP network. *Bioessays* **36**, 818–26. ISSN: 1521-1878 (Electronic) 0265-9247 (Linking). <https://www.ncbi.nlm.nih.gov/pubmed/24943963> (2014).
25. Hahn, I. *et al.* Tau, XMAP215/Msps and Eb1 co-operate interdependently to regulate microtubule polymerisation and bundle formation in axons. *bioRxiv* (2021).
26. Alves-Silva, J. *et al.* Spectraplakins promote microtubule-mediated axonal growth by functioning as structural microtubule-associated proteins and EB1-dependent +TIPs (tip interacting proteins). *J Neurosci* **32**, 9143–58. ISSN: 1529-2401 (Electronic) 0270-6474 (Linking). <https://www.ncbi.nlm.nih.gov/pubmed/22764224> (2012).
27. Voelzmann, A. *et al.* *Drosophila* Short stop as a paradigm for the role and regulation of spectraplakins. *Seminars in Cell and Developmental Biology* **69**. Spectraplakins, versatile roles in physiology and pathology Protocadherins and other atypical cadherins, 40–57. ISSN: 1084-9521 (2017).
28. Qu, Y., Hahn, I., Webb, S., Pearce, S. P. & Prokop, A. Periodic actin structures in neuronal axons are required to maintain microtubules. *Mol Biol Cell*. ISSN: 1939-4586 (Electronic) 1059-1524 (Linking). <http://www.ncbi.nlm.nih.gov/pubmed/27881663> (2016).
29. McNally, F. J. & Roll-Mecak, A. Microtubule-severing enzymes: From cellular functions to molecular mechanism. *Journal of Cell Biology* **217**, 4057–4069. ISSN: 0021-9525. <https://doi.org/10.1083/jcb.201612104> (Oct. 2018).

30. Brouhard, G. J. & Rice, L. M. The contribution of alphabeta-tubulin curvature to microtubule dynamics. *J Cell Biol* **207**, 323–34. ISSN: 1540-8140 (Electronic) 0021-9525 (Linking). <http://www.ncbi.nlm.nih.gov/pubmed/25385183> (2014).
31. Akhmanova, A. & Steinmetz, M. O. Control of microtubule organization and dynamics: two ends in the limelight. *Nat Rev Mol Cell Biol* **16**, 711–26. ISSN: 1471-0080 (Electronic) 1471-0072 (Linking). <https://www.ncbi.nlm.nih.gov/pubmed/26562752> (2015).
32. Brady, S. T. & Siegel, G. J. Basic Neurochemistry Principles of Molecular, Cellular and Medical Neurobiology Eighth Edition. <GotoISI> : // WOS : 000320652300002 (2012).
33. Kawaguchi, K. Role of kinesin-1 in the pathogenesis of SPG10, a rare form of hereditary spastic paraplegia. *Neuroscientist* **19**, 336–44. ISSN: 1089-4098 (Electronic) 1073-8584 (Linking). <http://www.ncbi.nlm.nih.gov/pubmed/22785106> (2013).
34. Crimella, C. *et al.* Mutations in the motor and stalk domains of KIF5A in spastic paraplegia type 10 and in axonal Charcot-Marie-Tooth type 2. *Clin Genet* **82**, 157–64. ISSN: 1399-0004 (Electronic) 0009-9163 (Linking). <http://www.ncbi.nlm.nih.gov/pubmed/21623771> (2012).
35. Ahmad, F. J. *et al.* Effects of dynactin disruption and dynein depletion on axonal microtubules. *Traffic* **7**, 524–37. ISSN: 1398-9219 (Print) 1398-9219 (Linking). <http://www.ncbi.nlm.nih.gov/pubmed/16643276> (2006).
36. Ferreira, A., Niclas, J., Vale, R. D., Banker, G. & Kosik, K. S. Suppression of kinesin expression in cultured hippocampal neurons using antisense oligonucleotides. *J Cell Biol* **117**, 595–606. ISSN: 0021-9525 (Print) 0021-9525 (Linking). <http://www.ncbi.nlm.nih.gov/pubmed/1533397> (1992).
37. Roote, J. & Prokop, A. How to design a genetic mating scheme: a basic training package for Drosophila genetics. *G3 (Bethesda)* **3**, 353–8. ISSN: 2160-1836 (Electronic) 2160-1836 (Linking). <http://www.ncbi.nlm.nih.gov/pubmed/23390611> (2013).
38. Prokop, A. Fruit Flies in Biological Research. English. *Biological Sciences Review* **28**, 10–14. ISSN: 0953-5365 (Apr. 2016).
39. Prokop, A. Why funding fruit fly research is essential for the biomedical sciences. <https://www.openaccessgovernment.org/fruit-fly-research/52396/> (2018).

40. Sanchez-Soriano, N., Tear, G., Whittington, P. & Prokop, A. *Drosophila* as a genetic and cellular model for studies on axonal growth. *Neural Dev* **2**, 9. ISSN: 1749-8104 (Electronic) 1749-8104 (Linking). <http://www.ncbi.nlm.nih.gov/pubmed/17475018> (2007).
41. Pandey, U. B. & Nichols, C. D. Human disease models in *Drosophila melanogaster* and the role of the fly in therapeutic drug discovery. *Pharmacol Rev* **63**, 411–36. ISSN: 1521-0081 (Electronic) 0031-6997 (Linking). <http://www.ncbi.nlm.nih.gov/pubmed/21415126> (2011).
42. Reiter, L. T., Potocki, L., Chien, S., Gribskov, M. & Bier, E. A systematic analysis of human disease-associated gene sequences in *Drosophila melanogaster*. *Genome Res* **11**, 1114–25. ISSN: 1088-9051 (Print) 1088-9051 (Linking). <http://www.ncbi.nlm.nih.gov/pubmed/11381037> (2001).
43. Debanne, D., Campanac, E., Bialowas, A., Carlier, E. & Alcaraz, G. Axon physiology. *Physiol Rev* **91**, 555–602. ISSN: 1522-1210 (Electronic) 0031-9333 (Linking). <https://www.ncbi.nlm.nih.gov/pubmed/21527732> (2011).
44. Gu, C. Rapid and Reversible Development of Axonal Varicosities: A New Form of Neural Plasticity. *Frontiers in Molecular Neuroscience* **14**, 1. ISSN: 1662-5099. <https://www.frontiersin.org/article/10.3389/fnmol.2021.610857> (2021).
45. Beaven, R. *et al.* *Drosophila* CLIP-190 and mammalian CLIP-170 display reduced microtubule plus end association in the nervous system. *Mol Biol Cell* **26**, 1491–508. ISSN: 1939-4586 (Electronic) 1059-1524 (Linking). <http://www.ncbi.nlm.nih.gov/pubmed/25694447> (2015).
46. Sanchez-Soriano, N. *et al.* Mouse ACF7 and *drosophila* short stop modulate filopodia formation and microtubule organisation during neuronal growth. *J Cell Sci* **122**, 2534–42. ISSN: 0021-9533 (Print) 0021-9533 (Linking). <http://www.ncbi.nlm.nih.gov/pubmed/19571116> (2009).
47. Schindelin, J., Arganda-Carreras, I. & Frise, E. e. a. Fiji: an open-source platform for biological-image analysis. *Nature Methods* **9** (2012).
48. Software, G. *version 9.0.0 for Windows* www.graphpad.com.
49. Bitplane. *Imaris for Neuroscientists* <https://imaris.oxinst.com/products/imaris-for-neuroscientists>.
50. McQuin, C. *et al.* CellProfiler 3.0: Next-generation image processing for biology. *PLOS Biology* **16**, 1–17 (July 2018).
51. Usov, I. & Mezzenga, R. FiberApp: An Open-Source Software for Tracking and Analyzing Polymers, Filaments, Biomacromolecules, and Fibrous Objects. *Macromolecules* **48**, 1269–1280 (2015).

52. Meijering, E. *et al.* Design and validation of a tool for neurite tracing and analysis in fluorescence microscopy images. *Cytometry A* **58**, 167–76. ISSN: 1552-4922 (Print) 1552-4922 (Linking). <https://www.ncbi.nlm.nih.gov/pubmed/15057970> (2004).
53. Mathworks. *MATLAB GUIDE* Web Page. 2015. <https://uk.mathworks.com/discovery/matlab-gui.html>.
54. Mathworks. *tic* Web Page. 2006. <https://uk.mathworks.com/help/matlab/ref/tic.html>.
55. Mathworks. *toc* Web Page. 2006. <https://uk.mathworks.com/help/matlab/ref/toc.html>.
56. Linkert, M. *et al.* Metadata matters: access to image data in the real world. *Journal of Cell Biology* **189**, 777–782. ISSN: 0021-9525. <GotoISI>://000278177500003 (2010).
57. Eddins, S. *Image Graphs* <https://www.mathworks.com/matlabcentral/fileexchange/53614-image-graphs>.
58. Jerman, T., Pernus, F., Likar, B. & Spiclin, Z. Beyond Frangi: an improved multiscale vesselness filter. *Medical Imaging 2015: Image Processing* **9413**. ISSN: 0277-786X. <GotoISI>://000355653800079 (2015).
59. Mathworks. *Deep Learning Toolbox* Web Page. 2018. <https://uk.mathworks.com/help/deeplearning/>.
60. Deng, J. *et al.* *ImageNet: A Large-Scale Hierarchical Image Database* in CVPR09 (2009).
61. Berg, S. *et al.* ilastik: interactive machine learning for (bio)image analysis. *Nature Methods*. ISSN: 1548-7105. <https://doi.org/10.1038/s41592-019-0582-9> (Sept. 2019).
62. Ilastik. *Ilastik Pixel Classification Documentation* <https://www.ilastik.org/documentation/pixelclassification/pixelclassification>.
63. Haubold, C. *et al.* Segmenting and Tracking Multiple Dividing Targets Using ilastik. *Adv Anat Embryol Cell Biol.* (2016).
64. Prokop, A., Kuppers-Munther, B. & Sanchez-Soriano, N. *Using Primary Neuron Cultures of Drosophila to Analyze Neuronal Circuit Formation and Function* ().
65. Saxton, W., Hicks, J., Goldstein, L. & Raff, E. Kinesin heavy chain is essential for viability and neuromuscular functions in *Drosophila*, but mutants show no defects in mitosis. *Cell* **64**, P1093–1102 (1991).
66. Brendza, K., Rose, D., Gilbert, S. & Saxton, W. Lethal Kinesin Mutations Reveal Amino Acids Important for ATPase Activation and Structural Coupling. *Journal of Biological Chemistry* **274**, 31506–31514 (1999).

67. Nguyen, M., Stone, M. & Rolls, M. Microtubules are organized independently of the centrosome in *Drosophila* neurons. *Neural Development* **6** (2011).
68. Kolodziej, P., Jan, L. & Jan, Y. Mutations That Affect the Length, Fasciculation, or Ventral Orientation of Specific Sensory Axons in the *Drosophila* Embryo. *Neuron* **15**, 273–286 (1995).
69. Kupperts-Munther, B. *et al.* A new culturing strategy optimises *Drosophila* primary cell cultures for structural and functional analyses. *Developmental Biology* **269**, 459–478. ISSN: 0012-1606. <https://www.sciencedirect.com/science/article/pii/S0012160604000892> (2004).
70. Zhang, Y., Tsinghua University, P. & DeGruyter. *Image Engineering* (2017).
71. Mendonca, A. M. & Campilho, A. Segmentation of retinal blood vessels by combining the detection of centerlines and morphological reconstruction. *IEEE Trans Med Imaging* **25**, 1200–13. ISSN: 0278-0062 (Print) 0278-0062 (Linking). <https://www.ncbi.nlm.nih.gov/pubmed/16967805> (2006).
72. Grel, C., Gol Mohammedzadeh, H. & Erden, A. Rose Stem Branch Point Detection And Cutting Point Location For Rose Harvesting Robot (July 2016).
73. Jiang, W. *et al.* A feature based method for trajectory dataset segmentation and profiling. *World Wide Web* **20**, 5–22. ISSN: 1573-1413. <https://doi.org/10.1007/s11280-016-0396-y> (2017).
74. Jerman, T., Pernu, F., Likar, B. & piclin, i. *Beyond Frangi: an improved multiscale vesselness filter* in. **9413** (), 94132A–94132A–11. <http://dx.doi.org/10.1117/12.2081147>.
75. Tabachnick, B. G. & Fidell, L. S. *Using multivariate statistics* (2018).
76. Frangi, A. F., Niessen, W. J. & Viergever, M. A. Three-dimensional modeling for functional analysis of cardiac images: a review. *IEEE Trans Med Imaging* **20**, 2–25. ISSN: 0278-0062 (Print) 0278-0062 (Linking). <https://www.ncbi.nlm.nih.gov/pubmed/11293688> (2001).
77. Longo, A. *et al.* Assessment of hessian-based Frangi vesselness filter in optoacoustic imaging. *Photoacoustics* **20**, 100200. ISSN: 2213-5979 (Print) 2213-5979 (Linking). <https://www.ncbi.nlm.nih.gov/pubmed/32714832> (2020).
78. Mathworks. *bwmorph* Web Page. 2006. <https://uk.mathworks.com/help/images/ref/bwmorph.html>.
79. Mathworks. *bwskel* Web Page. 2018. <https://uk.mathworks.com/help/images/ref/bwskel.html>.
80. Howe, N. R. *Skeletonisation Function* Computer Program. 2006. <http://www.science.smith.edu/~nhowe/research/code/>.

81. Telea, A. & Wijk van, J. *An augmented Fast Marching Method for computing skeletons and centerlines* in *Proceedings of the symposium on Data Visualization 2002 (VisSym'02, Barcelona, May 27-29, 2002)* (Eurographics, 2002), 251–259. ISBN: 1-58113-536-X.
82. Mathworks. *bweuler* Web Page. 2006. <https://uk.mathworks.com/help/images/ref/bweuler.html>.
83. Mathworks. *imbinarize* Web Page. 2016. <https://uk.mathworks.com/help/images/ref/imbinarize.html>.
84. Eddins, S. *Image Graphs* 2021. <https://www.mathworks.com/matlabcentral/fileexchange/53614-image-graphs>.
85. Mathworks. *hough* Web Page. 2006. <https://uk.mathworks.com/help/images/ref/hough.html>.
86. Mathworks. *houghpeaks* Web Page. 2006. <https://uk.mathworks.com/help/images/ref/houghpeaks.html>.
87. Mathworks. *houghlines* Web Page. 2006. <https://uk.mathworks.com/help/images/ref/houghlines.html>.
88. Hlavac, V., Pajdla, T. & Sommer, M. Improvment of the Curvature Computation. *IEEE*. ISSN: 1051-4651 (1994).
89. Yang, H., Luo, J., Shen, Z. & Wu, W. A local voting and refinement method for circle detection. *Optik* **125**, 1234–1239. ISSN: 0030-4026. <https://www.sciencedirect.com/science/article/pii/S0030402613011613> (2014).
90. Yao, Z. & Yi, W. Curvature aided Hough transform for circle detection. *Expert Systems with Applications* **51**, 26–33. ISSN: 0957-4174. <https://www.sciencedirect.com/science/article/pii/S0957417415008210> (2016).
91. Djekoune, A. O., Messaoudi, K. & Amara, K. Incremental circle hough transform: An improved method for circle detection. *Optik* **133**, 17–31. ISSN: 0030-4026 (2017).
92. Jiang, L., Wang, Z., Ye, Y. & Jiang, J. Fast circle detection algorithm based on sampling from difference area. *Optik* **158**, 424–433. ISSN: 0030-4026. <https://www.sciencedirect.com/science/article/pii/S0030402617317011> (2018).
93. Fornberg, B. & Zuev, J. The Runge phenomenon and spatially variable shape parameters in RBF interpolation. *Computers and Mathematics with Applications* **54**, 379–398. ISSN: 0898-1221. <https://www.sciencedirect.com/science/article/pii/S0898122107002210> (2007).

94. Schmidt, J., Evans, I. S. & Brinkmann, J. Comparison of polynomial models for land surface curvature calculation. *International Journal of Geographical Information Science* **17**, 797–814 (2003).
95. Tarolli, P., Cavalli, M. & Masin, R. High-resolution morphologic characterization of conservation agriculture. *CATENA* **172**, 846–856. ISSN: 0341-8162. <https://www.sciencedirect.com/science/article/pii/S0341816218303503> (2019).
96. Yang, Z.-B., Radzienski, M., Kudela, P. & Ostachowicz, W. Two-dimensional modal curvature estimation via Fourier spectral method for damage detection. *Composite Structures* **148**, 155–167. ISSN: 02638223 (2016).
97. Kerautret, B. & Lachaud, J. O. Curvature estimation along noisy digital contours by approximate global optimization. *Pattern Recognition* **42**, 2265–2278. ISSN: 00313203 (2009).
98. Web Page. 2021. <https://github.com/mooniean/ALFRED>.
99. Pearce, S. P., Heil, M., Jensen, O. E., Jones, G. W. & Prokop, A. Curvature-Sensitive Kinesin Binding Can Explain Microtubule Ring Formation and Reveals Chaotic Dynamics in a Mathematical Model. *Bull Math Biol* **80**, 3002–3022. ISSN: 1522-9602 (Electronic) 0092-8240 (Linking). <https://www.ncbi.nlm.nih.gov/pubmed/30267355> (2018).
100. Murphy, K. P. & Proquest. *Machine learning : a probabilistic perspective* (2012).
101. Holmes, S. & Huber, W. *Modern statistics for modern biology* (2019).
102. Goodfellow, I., Bengio, Y. & Courville, A. *Deep learning* xxii, 775 pages. ISBN: 9780262035613 (hardcover alk. paper) 0262035618 (hardcover alk. paper). <https://mitpress.mit.edu/books/deep-learningMITPressOpenAccess> (MIT Press, Cambridge, MA, 2016).
103. Simonyan, K. & Zisserman, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition in International Conference on Learning Representations* (2015).
104. Web Page. <https://github.com/MartinThoma/LaTeX-examples/blob/master/tikz/vgg-16/vgg-16.tex>.
105. He, K., Zhang, X., Ren, S. & Sun, J. *Deep Residual Learning for Image Recognition in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), 770–778.
106. Mary, H. & Brouhard, G. J. Kappa (): Analysis of Curvature in Biological Image Data using B-splines. *bioRxiv* (2019).
107. Napari contributors. *napari: a multi-dimensional image viewer for python* 2019.
108. Ilastik. *Ilastik Object Classification Documentation* <https://www.ilastik.org/documentation/objects/objects>.

109. Mathworks. *bwlabel* Web Page. 2006. <https://uk.mathworks.com/help/images/ref/bwlabel.html>.
110. Mathworks. *regionprops* Web Page. 2006. <https://uk.mathworks.com/help/images/ref/regionprops.html>.

Appendix A

ALFRED: Software Implementation and User Manual

A.0.1 Loading Window layout and functionality

The first window opening when running ALFRED is the Loading Window, as seen in Figure A.1. As mentioned, it allows the user to load the images into the program.

By pressing the button `Load`, the user can choose the folder from which to load the image(s). A small loading pop-up will keep track of how many images have been loaded and how many are left. In Figure A.1b, there is an example of a panel loaded and cropped in six columns and 5 rows.

The files are read using BioFormat (see section 2.1.2). This provides flexibility regarding the type of image files that can be opened, ranging from simple .jpg to more complex microscope-related stacks (e.g., STED images). In the latter case, ALFRED will separate the stacks into its single images.

In the case that a panel is loaded onto the software, it can be divided into same-size rectangles, simply by inserting the intended number of rows and columns and pressing `Update`. There is an additional white border checkbox, for the cases where the images are separated by a defined border of white pixels that are accounted for in the cropping. The button `Restore` undoes the division and the preview will show the original panel.

Each image is numbered, with a number that will also be saved in the backup and final files. If more than 30¹ images are loaded, or if a panel has more than 30 divisions, then a new page is created and the user can navigate between them by using the `Previous Page` and `Next Page` buttons.

After images have been loaded, the analysis can begin. To open the next window, the user simply needs to click a certain preview, which then opens the full image in the main window which pops-up (details in Section A.0.2).

As buttons only become available once an image is presented in the preview, the

¹The number 30 was chosen to facilitate the study of panels for the experiments described.

platform is robust (i.e. does not cause an error when the user does something unexpected) to user input and interaction: from checking the file input, guaranteeing the images load accordingly, to their preview or the requests to split the image, as well as the specific order of buttons to be clicked.

To verify robustness, the tests performed ranged from opening different types of valid and invalid imaging formats, attempting to press buttons that are otherwise unavailable or even cancelling the loading of images.

A.0.2 Main Window layout and functionality

The main window will open with the image on the left hand side (Figure A.2a). If the image has 3 channels or less, the image will show them overlapped as an RGB image. However, if there are more, only the first channel will be shown.

Before any of the options are selected, the program saves the images on a backup .mat file (MATLAB data file), saved under the name of the image. From this point onwards, any changes are shown live in the left hand panel.

After each option on the right hand side menu (Figure A.2b) is newly selected, the progress is saved on the backup file to ensure that no work is lost in case a program error occurs. Furthermore, the original image matrices are kept intact in memory, so any progress can be undone up to the starting image, even before virtualisation.

Furthermore, running the same step repeatedly with the same values will not influence the resulting image, for example, virtualisation is always applied to the channel and not to an already virtualised image.

The `View Original Image` button at the top allows a quick comparison between the image at any stage with the initial image, by overlaying the images and the user can click in and out. This allows the user to keep the real image in mind while working with its monoscale or binary versions.

The first option on the control panel (Figure A.2b) is a checkbox of whether the image has a dark (normal) or light background (inverse). In the usual fluorescence imaging, the background is dark. Normalisation in this case is to reduce the images to 8bit (each pixel is normalised to values between 0 and 255), which usually allows a faster analysis than higher values (such as 16bit). Furthermore, if the input image is 8bit, there will be no difference from the original.

The Channel Selection drop-down menu allows choosing the desired channel for analysis (Figure A.3). In this example, and with RGB images, the channels have the correspondence of 1-Red, 2-Green and 3-Blue. If all the images loaded to the program were RGB, the channels could be named after the colours. However, the program does not really recognise colours, but rather the overlapping matrices. As such, if the image type has a different number of channels, these would not be assigned to any colour, but would appear numbered.

Upon channel selection, the image instantly goes through the virtualisation/vesselness process (Section 3.2.1). The function sensitivity (parameter τ mentioned in the method by Jerman, Section 3.2.1) can be tuned with the slider and the `Virtualise` button, which will always apply vesselness to the original image channel with the new specified value.

The Greyscale Threshold slider enables the user to set the greyscale threshold of the binary mask, which can be compared at all times with the original by clicking the `View Original Image` button at the top.

As mentioned in Chapter 3, the ensuing step of the image processing is skeletonisation. However, this is a computationally demanding process, scalable with the size of the image. As such, the skeletonisation is only applied after the user has selected a ROI, and a new window opens (Section A.0.3) with only the relevant selected region.

The region selection is generally done with a rectangular tool: highlighting a rectangle of the larger image to be analysed individually. However, some images have high background noise, or where cells are positioned close together. There is a free hand tool which allows a more detailed selection around the regions. The image cut-off is defined by the bounding box (dashed lines in Figure A.4) - the coordinates of the smallest possible rectangle that contains the selection area. In the case of free hand tool (Figure A.4b), it is the minimum bounding box that encompasses the selection, and everything outside the selected shapes is set to zero for analysis purposes of selection.

As soon as the selection is completed, a new pop-up window is opened with the skeletonised version of the region (ROI window, further described in Section A.0.3), which has been calculated from the mask in an automated interim step (see Flowchart, Figure 5.2).

After closing the ROI Window, if the selection is saved, the properties of the path selection are further saved on the main backup file, and the bounding box of the region is highlighted with a dashed rectangle on the image (Figure A.5, green for microtubule disorganisations, magenta for axons).

Furthermore, whenever the user changes between images in the same panel, the processed regions will still be highlighted, and the user can keep track of which parts have been done.

The button `Discard All Regions` allows the user to erase all ROIs selected and restart the selection, without having to process the images again.

An important detail in analysing the images is the scaling factor between pixels and the measurements of the original image. The scaling factor is formatted as pixel per unit. To set the conversion factor, the user needs to click the `Change` button that prompts a pop-up window (described in Section A.0.4). If no conversion factor has been input before, the scale change is retroactively applied to all the previously processed images. The default calculation is 1 pixel per μm .

The final group of buttons in the control panel of the Main Window represent general actions:

- `Undo` - retracts the last action performed by the user, at any step.
- `Previous` - opens the previous image in the group.
- `Continue` - opens the next image in the group.
- `Calculate All` - triggers the calculations (Section A.0.5) for all the images processed thus far.

When changing between images with `Previous` and `Continue`, the image processing steps are automatically applied even if the image has been previously processed, as images are generally analysed in succession and have been obtained using the same microscope. The channel selection and virtualisation will be the same throughout all images of a panel or stack, as well as the conversion factor.

To improve performance, the calculations should be done after all ROIs in all images have been selected.

A.0.3 ROI Window

The ROI window is prompted with two inputs provided by the main window: the selected region of the selected channel and its skeleton, calculated after the bounding box selection has been done by the user (Figure A.6). When the window opens, it displays the skeleton (white line) overlapping the image (section from the original image matrix) to allow the user to judge as to whether the program has recognised the shape accurately. This is the point where the methods in Section 3.3.2 are applied on the background, as a graph is created from the binary image.

The type selection provides a dropdown menu, which allows the selection between the two types of regions contained in the ROI: either microtubule disorganisation (Figure A.6b (1)), or axon (Figure A.6c (1)). This selection triggers the availability of the adequate set of selection tools (Figure A.6b,c (2)).

This selection will enable the necessary buttons for each type of analysis ahead (present in Figure A.6b (2)). Firstly, both will enable the `Start Selection` button. This will allow the user to click the end points (beginning and end of the region) of the relevant section of the skeleton.

As the skeleton is a pixel-wide line on a black background, it would be difficult for the user to click exactly on a specific pixel of the skeleton. As such, there is a search scope around each click with a certain radius, searching for the closest node to the selected coordinates (using the function `checkCoordinates.m`). There is a constant check and conversion between the (x, y) coordinates of the image and the nodes in the graph, to guarantee the closest real image values at any point.

The aforementioned radius for the search scope is controlled by the jump slider and the algorithm is allowed to recognise between two unconnected nodes in the graph. This is also to account for image artefacts where there is not a fully connected path but it is still close enough to recognise it.

If the selection is out of the scope of any point in the graph, a prompt appears to warn the user and asks for a new coordinate. The selected coordinates will then be displayed in green on the skeleton (Figure A.7a). After, the algorithms presented in Section 3.3 are called to extract the path between the two end points, which will be shown in blue. Further selection points can be added.

On the other hand, the checkbox for the entire path allows the user to mark all nodes in the graph as part of the path without any user input (Figure A.7b), which is particularly useful when applying to the test images used in testing, or when the user selection is very comprehensive.

The amount of points to be added at one particular step can be changed in the editable text box. The type of points to add to the graph differs between regions, and the changes are automatically applied and shown in the left hand-side panel.

If the region was selected as axon, and the suggested path does not match the user's interpretation, `Inner Points` can be added, which will determine crucial nodes of the skeleton along which the path needs to go through. This might be needed also if there is a significant gap. Furthermore, the user can `Add Lines` to circumvent any imaging problems (e.g., image presents a dotted line instead of a continuous one).

If, on the other hand, the selected region was microtubule disorganisation, the user can add `Outer Points`, which will delete connections to regions that don't belong to the disorganisation, thus trimming the region and removing any unwanted branches such as, for example, parts of the axon (Figure A.7c,d).

The user can press `Undo` to eliminate any wrong selections or unwanted points, in the reverse order they were added. This is done by deleting from the saved internal matrices the most recently saved values. If the user wants to restart the selection and clear the path completely (i.e., resetting the window), there is the `Reset Path` button.

At any point, the user can make changes to the options above and press `Apply` to update the path selection.

After the selection, the user can either `Save and Close`, where the values are saved to the main dataset (connecting all windows of ALFRED), or `Cancel` and discard the changes. However, before doing so, there's a prompt to confirm with the user that it doesn't want to save any changes or information gathered in that region.

A.0.4 Microscope Specifications Window

Before ALFRED calculates all of the values mentioned in Chapter 4, it needs a value for the conversion from pixels to another unit of length. As such, when pressing the `Image Specs Change` button, a pop up appears, as seen in Figure A.8.

There are three options that a user can choose from in order to select the conversion.

- **Metadata** (Figure A.8a): Certain types of files, such as images from a STED microscope, will include metadata with each image, provided by the image acquisition software and automatically loaded by BioFormats. This usually includes the pixel/unit conversion needed. As such, with a simple checkbox, that value will be accessed.
- **Microscope Scaling Factors** (Figure A.8(b)): If the images do not provide the correct needed conversion, then the user can select from predefined scaling factors. Currently, this can only be altered manually previous to running the program. On the subfolder `\documents`, there is an Excel spreadsheet named `scalingfactors.xlsx`:

On the inside, there is a table with the conversion values for the selected microscope, as in Figure A.9. When the user selects the microscope, the Objective will automatically update with the available options from the sheet, as well as Optovar and Binning options after each has been sequentially selected. Finally, ALFRED will get the designated value from the provided table, as soon as the user presses `Apply`.

- **Manual Insertion** (Figure A.8(c)): the default unit is 1 pixel per micrometers (μm), but can be changed with the drop-down menu. If the user wishes to obtain the values in pixels, just insert manually 1 and no conversion will be calculated.

After each selection, the user can simply press the button `Apply` and continue the analysis. On the other hand, if the user does not want to apply any changes, the `Cancel` button can be pressed.

A.0.5 Analysis and Calculations

After all the ROIs have been selected, the last step their individual analysis. In order to save computational efforts and time, all the calculations are performed after the button `Calculate All` is selected. As such, the user can stop the region selection at any point and perform the calculations afterwards.

Furthermore, even though all values are calculated and saved on the back up file, a pop-up shows up (Figure A.10) that allows the user to have selected results

saved on a separate comma-separated value (csv) file. These can later be opened in spreadsheet analysers such as Excel.

Before any calculation is saved, it is important to check whether all the images have had the conversion factor updated. If none was changed, all are set to the default value of $1\text{px}/\mu\text{m}$. These are added retroactively to any image whose conversion factor was not changed.

The regions considered are the blue paths selected, as in Figure A.7. The straightness index is calculated for both axons and microtubule disorganisation regions by dividing the sum of all straight segment lengths obtained with the Hough Transform per the total length of the path. While in the axon the path tends to be linear, in microtubule disorganisation the total region of the skeleton selected by the user is considered.

As morphologically both regions are different, some calculations are handled separately for the microtubule disorganisation.

Most of the calculations for axon regions, such as straightness, are calculated when ROIs are selected. This happens when the calculations are seamless and, therefore, will not influence the work-flow. If the calculations were slow, they could be moved to this section.

Furthermore, the only calculation needed at this stage is the conversion of all the values (particularly axon length) with the scaling factor.

Microtubule Disorganisation

For the disorganisation regions, an additional step is needed. The best way to analyse these complex shapes is by dividing them into different segments of connected components. Each region is converted into a binary mask (as in Chapter 3), and each connected component is labelled.

MATLAB provides the function needed to perform the labelling (*bwlabel* [109], Figure A.11), whose output is an image where each point in a connected component has the value attributed.

This labelled image will then be the input to another MATLAB function to analyse the properties of each individual region (*regionprops* [110]). While a lot of values can be obtained, only two provided the needed morphological insight: eccentricity and area.

Eccentricity is the “un-circularity” of a shape, defined as its difference to circle: a larger eccentricity corresponds to a less symmetrically curved shape. As the loops in disorganisation appear to be circular, this could be an indication of their shapes. Furthermore, area of each individual swelling is also calculated, which provides an insight on the average size of swellings.

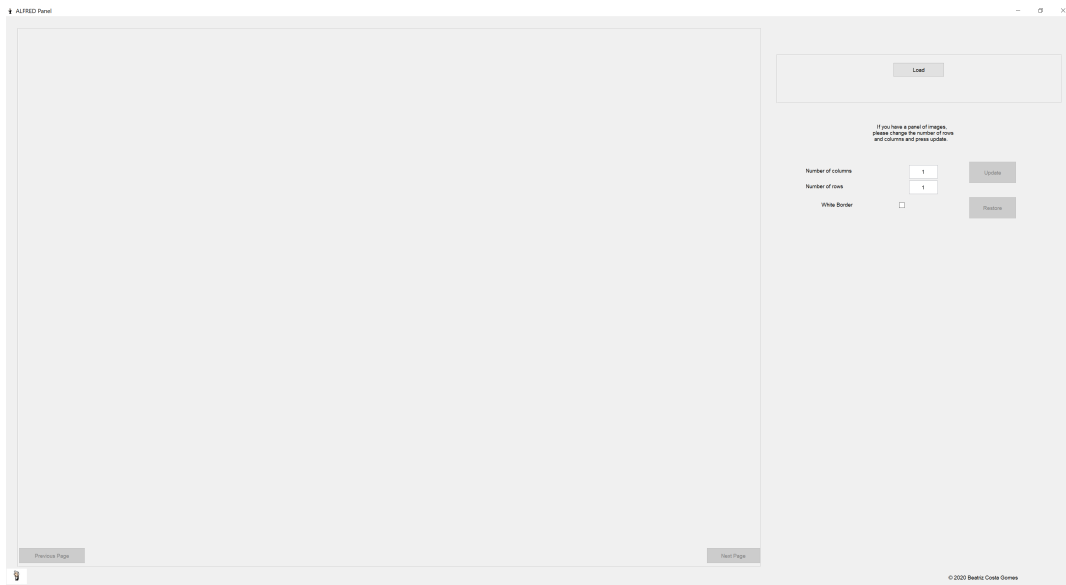
There are two other different types of areas of interest: microtubule disorganisation area and the effective area occupied by microtubules in the swelling. These are

used to calculate the density of the disorganisation region. As the calculations are done in binary masks, the area is the sum of all values in the selected points binary matrix. The density becomes, therefore, the area of the microtubules divided by the total area of the swelling.

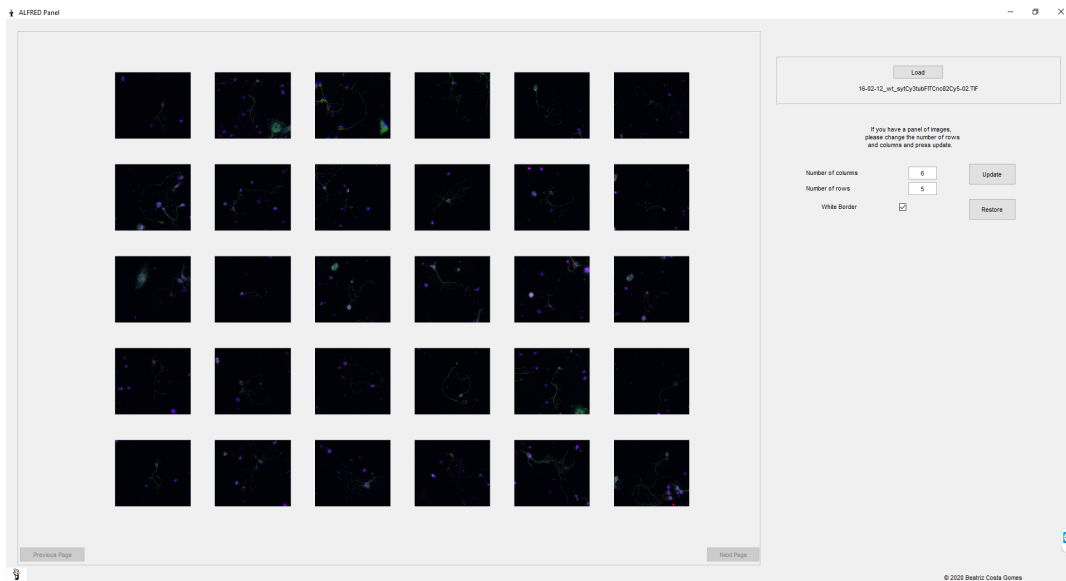
Most of the computational time is spent on the next calculation: the curvature (Section 4.3).

First, each individual skeleton for each region goes through the pre-processing needed for the curvature calculation, i.e., the removal of branch points and general cleaning of smaller branches. The remaining matrix can now go through the calculation, and the final output are radius and arc length for each spline.

After all the values are calculated, the .csv file can be created (if selected by the user). The user can now proceed for bulk analysis of their biological images.

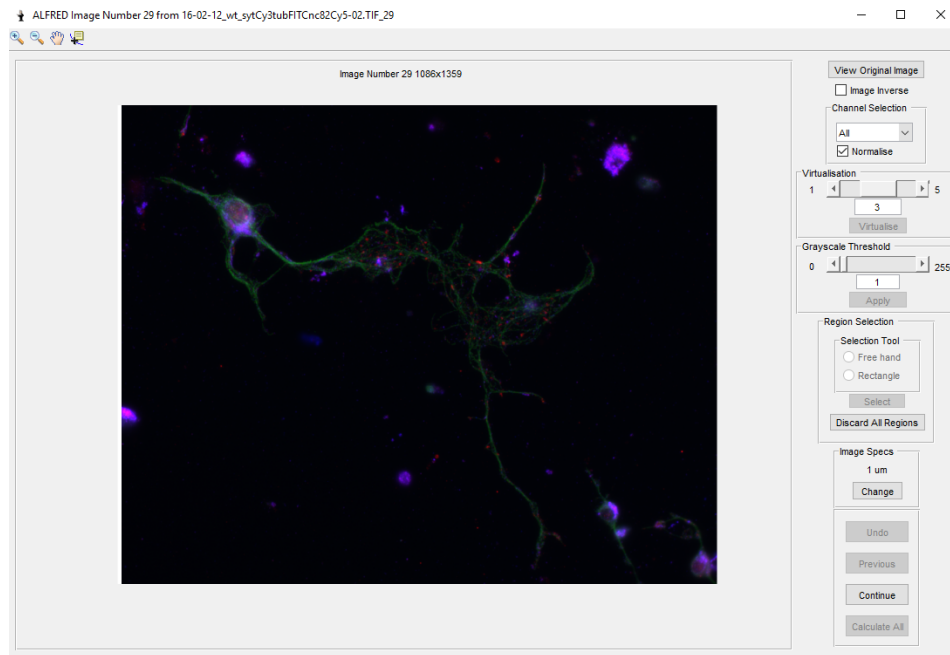


(a)

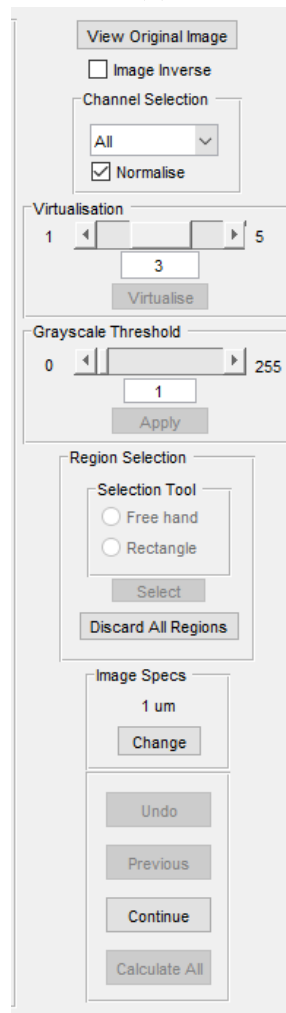


(b)

Figure A.1: Loading Window of ALFRED: (a) View of the window when it opens. (b) View of the window after loading and cropping a panel. The majority of the window is the region where the images will be previewed. On the right, there is the `Load` button; the option to divide it into same sized blocks in case a panel was loaded (with an input for the intended number of rows and columns). The images can be previewed on the left side of the window.



(a)



(b)

Figure A.2: Main Window of ALFRED. (a) Full view with image number 29 from the panel in Figure A.1b loaded. Image shown on the left hand side, with MATLAB Figure tools for zoom and positioning above; control panel on the right is shown as close-up in (b), and further explained in the main text.

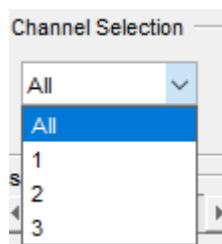


Figure A.3: Example of channel selection options if the Main Window of ALFRED. When clicking the chevron, the selection options are shown as arbitrary numbers: in this case, 1 corresponds to red, 2 to green and 3 to blue. The program does not identify the colour of the channel, only the numbers and these can be different between different types of images.

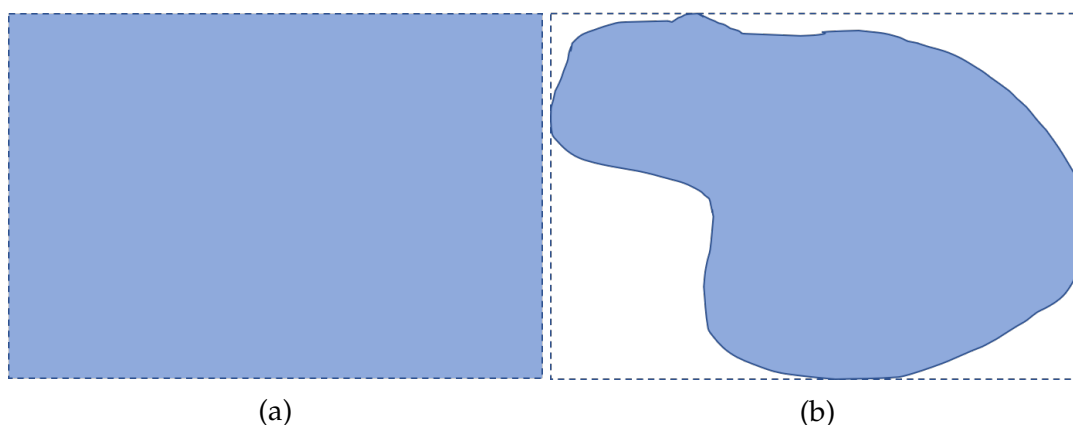
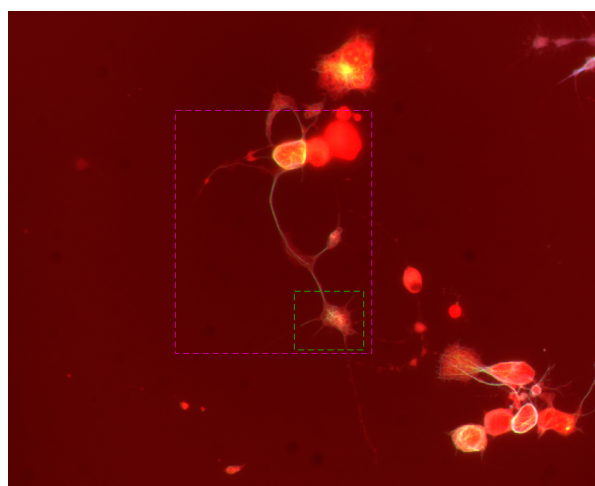
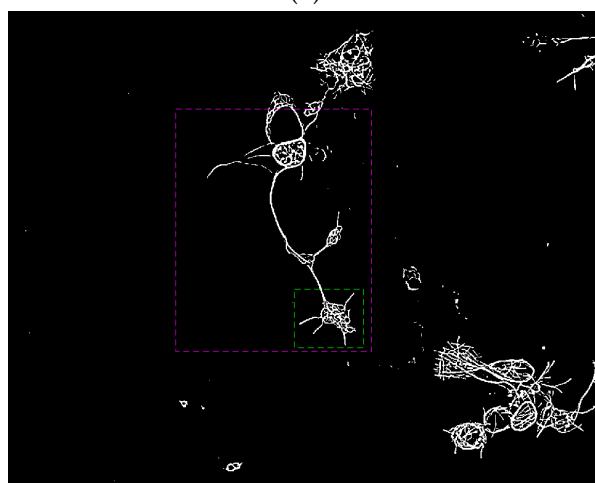


Figure A.4: The region selection options in the Main Window of ALFRED. (a) A rectangular selection, (b) a free hand selection (blue). Both have the same bounding box (dashed line), but in (b) the pixels outside the selected area are set to 0.

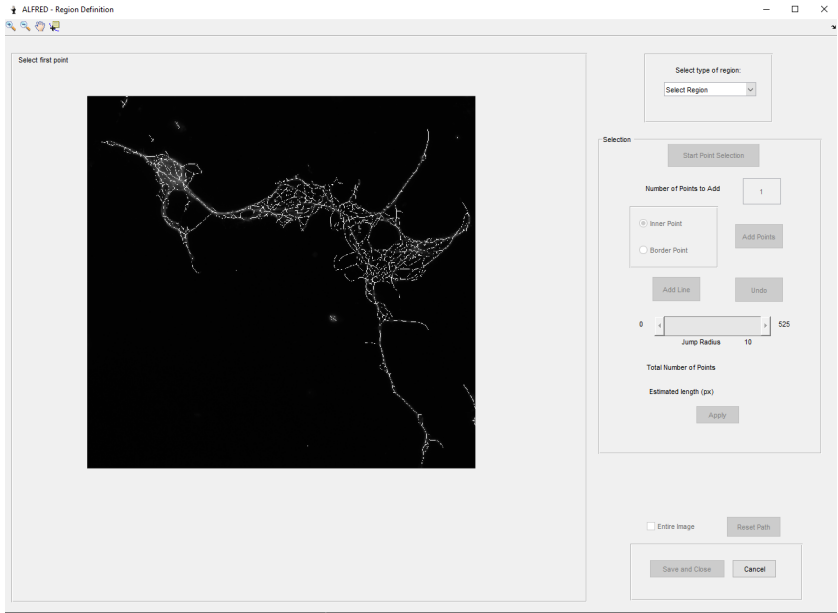


(a)

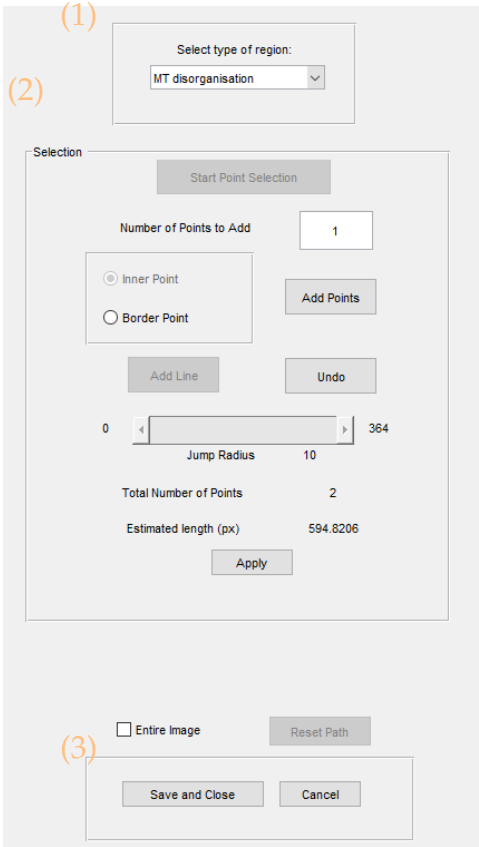


(b)

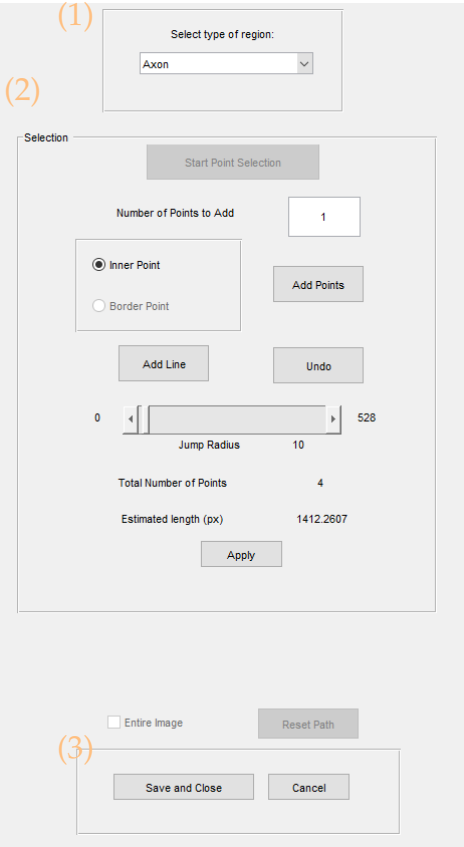
Figure A.5: Highlighting processed regions in ALFRED in the RGB image (a) and binary mask (b). Two different bounding boxes of processed regions are highlighted with dashed lines: magenta for axon, green for microtubule disorganisation.



(a)

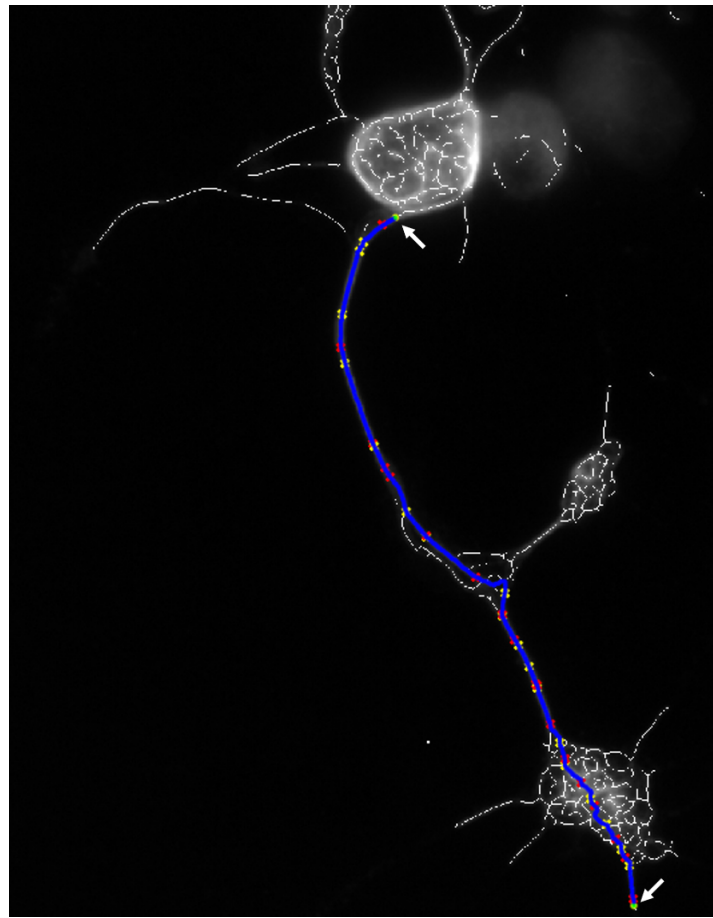


(b)

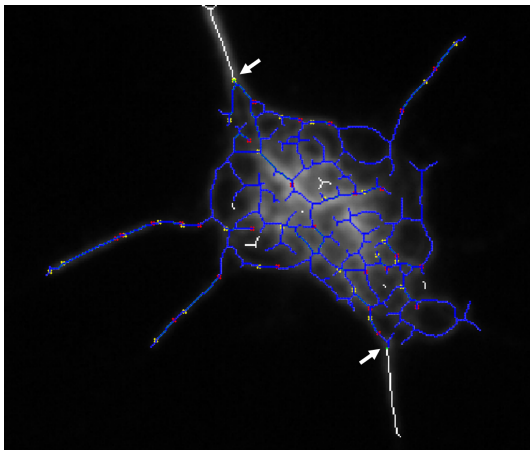


(c)

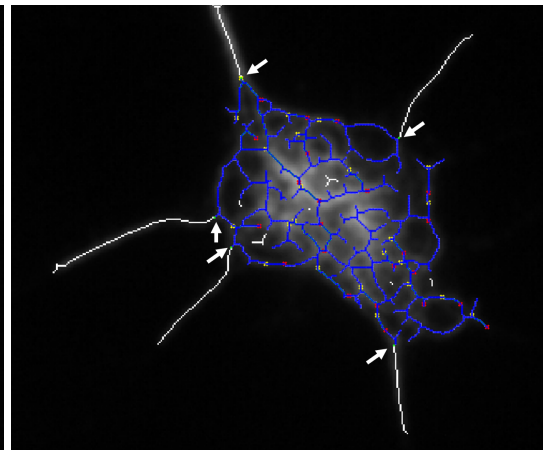
Figure A.6: Opening of the ROI window of ALFRED. (a) Overview of the skeletonised image on the left and a control panel on the right. Close up of control panel for microtubule disorganisation (b) and axon (c): (1) dropdown menu to select which type of region it is, whether microtubule disorganisation or axon. (2) Selection control, where the relevant points for path definition in the image can be selected. (3) Saving and closing options to keep or discard the calculated values.



(a)



(b)



(c)

Figure A.7: Example of path selection in the ROI window of ALFRED. (a) Axon recognition with two input points; (b) Close up and recognition of full microtubule disorganisation within the two selected points. (c) Trimming of branches in the disorganisation. Blue is the recognised regions/paths for calculation. Red and yellow dots represent the extremity of straight segments calculated with Hough Transform (Section 4.2). Arrows are the user input points.

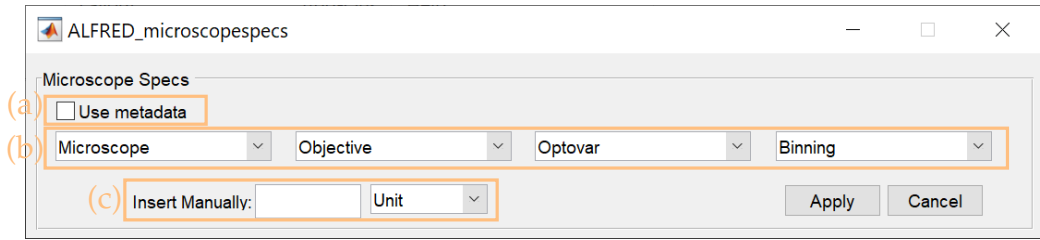


Figure A.8: Microscope specifications window of ALFRED. Orange embossed areas indicate the type of conversion available: (a) use metadata from the images provided by the image acquisition software, (b) scaling factors associated with microscopes, from Figure A.9 and (c) insert units manually.

Camera: MatrixVision mvBlueFox3-M2 2124G (right side)					Camera: MatrixVision mvBlueFox3-M2 2124G (left side)				
pixel per µm	binning	1x1	2x2	4x4	pixel per µm	binning	1x1	2x2	4x4
objective	optovar				objective	optovar			
10 air	1 10 air	1	2.88377	1.441885	0.720943	10 air	2.900686	1.450343	0.725172
10 air	1.25 10 air	1.25	3.607372	1.803686	0.901843	10 air	3.625858	1.812929	0.906464
10 air	1.6 10 air	1.6	4.599615	2.299808	1.149904	10 air	4.641098	2.320549	1.160274
10 air	2 10 air	2	5.760132	2.880067	1.440033	10 air	5.801372	2.900686	1.450343
40 air	1 40 air	1	11.58808	5.794039	2.897019	40 air	11.60274	5.801372	2.900686
40 air	1.25 40 air	1.25	14.49196	7.245979	3.62299	40 air	14.50343	7.251716	3.625858
40 air	1.6 40 air	1.6	18.45524	9.227621	4.613811	40 air	18.56439	9.282196	4.641098
40 air	2 40 air	2	23.07378	11.53689	5.768446	40 air	23.20549	11.60274	5.801372
40 sil oil	1 40 sil oil	1	11.65763	5.828817	2.914409	40 sil oil	11.60274	5.801372	2.900686
40 sil oil	1.25 40 sil oil	1.25	14.51095	7.255476	3.627738	40 sil oil	14.50343	7.251716	3.625858
40 sil oil	1.6 40 sil oil	1.6	18.46834	9.234169	4.617084	40 sil oil	18.56439	9.282196	4.641098
40 sil oil	2 40 sil oil	2	23.02615	11.51308	5.756538	40 sil oil	23.20549	11.60274	5.801372
60 air	1 60 air	1	17.1559	8.577949	4.288974	60x H2O	17.6251	8.812549	4.406275
60 air	1.25 60 air	1.25	21.43385	10.71692	5.358462	60x H2O	22.03137	11.01569	5.507843
60 air	1.6 60 air	1.6	27.35694	13.67847	6.839236	60x H2O	28.20016	14.10008	7.050039
60 air	2 60 air	2	34.12821	17.0641	8.532051	60x H2O	35.2502	17.6251	8.812549
100 oil	1 100 oil	1	29.21154	14.60577	7.302885	100 oil	29.33333	14.66667	7.333334
100 oil	1.25 100 oil	1.25	36.41026	18.20513	9.102564	100 oil	36.66667	18.33333	9.166667
100 oil	1.6 100 oil	1.6	46.69661	23.3483	11.67415	100 oil	46.93333	23.46667	11.73333
100 oil	2 100 oil	2	58.56612	29.28306	14.64153	100 oil	58.66667	29.33333	14.66667

Figure A.9: Example of scalingfactors.xlsx. The table is organised per microscope, optovar and binning options.

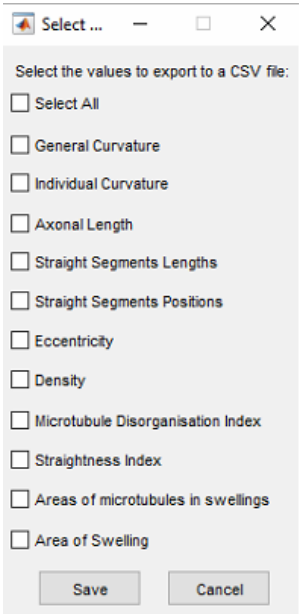


Figure A.10: Saving options pop-up of ALFRED. The user chooses the values to save on the csv file.

1	1	1	0	0
1	1	0	0	1
1	0	0	1	0
0	0	1	0	0
0	1	0	0	1

(a) Skeleton

1	1	1	0	0
1	1	0	0	2
1	0	0	2	0
0	0	2	0	0
0	2	0	0	3

(b) Labelled Image

Figure A.11: Image Matrix Labelling. (a) Binary image (skeleton) where points are either 1 or 0. (b) Labelled image, where each connected component (8-connectivity) is attributed a number, the label, by substituting 1 for that value.