Inaugural dissertation
for
obtaining the doctoral degree
of the
Combined Faculty of Mathematics, Engineering
and Natural Sciences
of the
Ruprecht - Karls - University
Heidelberg

Presented by
B.A. Kimberly Meechan
Born in: Dorchester, United Kingdom
Oral examination: 28th March 2022

Tools for multimodal atlases: bridging morphology and gene expression

Referees: Jun.-Prof. Dr. Steffen Lemke
Dr. Anna Kreshuk

Summary

Maps combine different types of data, from a number of different sources, into the uniform grid of their coordinate system. They are invaluable resources as they allow the connections between data to be understood in their spatial context. We have been making biological maps for decades of all kinds of systems - for example, model organisms like *Drosophila melanogaster* and *Caenorhabditis elegans* have large atlases of neuronal cell types available.

These large multimodal atlases (i.e. datasets that combine multiple data sources into one coordinate system) tend to undergo a similar 'lifecycle'. First, a standard coordinate system is formed, followed by collection of the required datasets. These are then integrated into the common coordinate system, before they are analysed, and finally shared as a resource to the biological community.

Electron microscopy (EM) offers many advantages when integrated into atlases of this kind. Firstly, it can achieve very high resolution, making it ideal for visualising small features like the fine structure of different organelles. It also provides the full spatial context of a tissue - allowing the locations and orientations of cells to be understood in a comprehensive manner. This comes at the cost of a number of challenges though. For one, EM is quite low throughput, meaning that often only a single sample can be imaged, or only small regions of a sample. Also, EM data can be extremely large (many terabytes in size) making it difficult to store, analyse, and share with other researchers.

In my thesis, I aim to ease the integration of large-scale volume EM into multimodal atlases at various points within this lifecycle. First, at the acquisition stage, I look at increasing the speed, ease and precision of targeted EM acquisition. I develop two different methods - one focused on trimming resin-embedded blocks with an ultramicrotome, and the other with a Serial Block Face Scanning Electron Microscope (SBEM). Both these methods are provided as user-friendly Fiji plugins, making them accessible to those with no programming experience.

Second, at the analysis stage, I create an analysis pipeline for comparing morphology (from EM) with gene expression of cell types in a large multimodal atlas. This focuses on a large atlas of the organism *Platynereis dumerilii* that combines volume EM with gene expression patterns for over 200 genes. This analysis provides full body clustering of cells based on morphology and gene expression, and shows a clear correspondence between gene expression and morphological tissue boundaries from the EM.

Finally, at the resource sharing stage, I look at improving the ease of sharing and exploration of these massive EM datasets. To do this, I contribute to the software MoBIE that allows interactive browsing of very large, remotely stored image data. I focus on making the creation of MoBIE projects accessible to those with no programming experience, adding a user interface for creating projects via the Fiji plugin.

Zusammenfassung

Karten, im Allgemeinen, stellen verschiedene Datentypen aus einer Reihe verschiedener Quellen in einem einheitlichen Koordinatensystem dar. Sie sind unschätzbare Ressourcen, da sie es ermöglichen, die Verbindungen zwischen Daten in ihrem räumlichen Kontext zu verstehen. Wir erstellen seit Jahrzehnten Karten für biologische Daten verschiedenster Systeme – für Modellorganismen wie *Drosophila melanogaster* und *Caenorhabditis elegans* gibt es beispielsweise große Atlanten neuronaler Zelltypen.

Diese großen multimodalen Atlanten (d. h. Datensätze, die mehrere Datenquellen in einem Koordinatensystem kombinieren) durchlaufen in der Regel einen ähnlichen „Lebenszyklus“. Zuerst wird ein Standardkoordinatensystem gebildet, gefolgt von der Sammlung der erforderlichen Datensätze. Diese werden dann in das gemeinsame Koordinatensystem integriert, bevor sie analysiert und schließlich als Ressource an die naturwissenschaftliche Gemeinschaft weitergegeben werden.

Elektronenmikroskopie (EM) bietet viele Vorteile, wenn sie in solche Atlanten integriert wird. Erstens erreicht sie eine sehr hohe Auflösung, ideal um kleine Merkmale wie die Feinstruktur verschiedener Organellen zu visualisieren. Ausserdem liefert EM den vollständigen räumlichen Kontext eines Gewebes – und ermöglicht so ein umfassendes Verständnis der Lage und Ausrichtung von Zellen. Dies geht jedoch mit einer Reihe von Herausforderungen einher. Zum einen hat EM einen recht geringen Durchsatz, was bedeutet, dass oft nur eine einzelne Probe oder nur kleine Bereiche einer Probe abgebildet werden können. Außerdem können EM-Daten extrem groß sein (mehrere Terabyte), was es erschwert sie zu speichern, zu analysieren und mit anderen Wissenschaftlern zu teilen.

In meiner Dissertation möchte ich die Integration von large-scale EM in multimodale Atlanten an verschiedenen Stellen dieses Lebenszyklus erleichtern. Zunächst, innerhalb der *Acquisition Stage*, arbeite ich an Geschwindigkeit, Einfachheit und Präzision der getargeten EM-Akquisition, d. h. der gezielten und selektiven Aufnahme von Zielstrukturen. Ich entwickle zwei verschiedene Methoden: bei der Einen werden in Kunstharz eingebettete Blöcke mittels Ultramikrotom und bei der Anderen direkt im Serial Block Face Scanning Electron Microscope (SBEM) getrimmt. Beide Methoden habe ich als benutzerfreundliche Fiji-Plugins implementiert, sodass sie für Personen ohne Programmiererfahrung zugänglich sind.

Des Weiteren erstelle ich in der *Analysis Stage* eine Analysepipeline zum Vergleich von Morphologie (gemäß EM) und Genexpression von Zelltypen in einem großen multimodalen Atlas. Im Fokus steht ein großer Atlas des Organismus *Platynereis dumerilii*, der Volumen-EM mit Genexpressionsmustern für über 200 Gene kombiniert. Diese Analyse liefert ein Zell-Clustering für den gesamten Organismus, das auf Morphologie und Genexpression basiert und eine klare Übereinstimmung zwischen Genexpression und morphologischen Gewebegrenzen zeigt.

Schließlich betrachte ich in der *Resource Sharing Stage* die Vereinfachung der gemeinsamen Nutzung und Untersuchung dieser riesigen EM-Datensätze. Dazu beteilige ich

mich an der Entwicklung der Software MoBIE, die ein interaktives Navigieren sehr großer, entfernt gespeicherter Bilddaten ermöglicht. Ich konzentriere mich unter Anderem darauf, die Erstellung von MoBIE-Projekten für Personen ohne Programmiererfahrung zu ermöglichen, indem ich eine Benutzeroberfläche zum Erstellen von Projekten über das Fiji-Plugin entwickle.

Acknowledgements

First of all, I would like to thank my PhD supervisor Yannick Schwab for all his support throughout these long four years. His EM knowledge and endless enthusiasm have been greatly appreciated, especially during these strange coronavirus times.

I'm also very thankful to all the members of the Schwab lab and EMCF, who have supported this project throughput. Special thanks to everyone who stepped in to help with lab work when I could not always be there in person. This PhD would not be finished without you! Thanks to Rachel Templin for her help as the fellow Schwab lab *Platynereis* expert, and to Nicole Schieber, Rosa Pipitone and Inés Romero Brey for their help with the ultramicrotome targeting project.

Also, thanks to all the members of the Arendt lab who introduced me to *Platynereis dumerilii*, and provided their knowledge and enthusiasm throughout. Also, of course, to everyone who worked on the PlatyBrowser project - Hernando M. Vergara, Constantin Pape, Valentyna Zinchenko, Kevin Nzumbi Mutemi, Rachel Templin, Anna Kreshuk, Detlev Arendt, Christian Tischer and everyone else.

In addition, thanks to Christian Tischer and Constantin Pape for their help on numerous coding and image analysis projects throughout my PhD. Working on MoBIE together has been one of the most fun parts of my time at EMBL, and I've learnt so much about programming from both of them. Continuing with the coding projects, big thanks to Benjamin Titze for introducing me to SBEMimage and helping me get started with developing for the SBEM. Also, thanks to Julian Hennies as the fellow Schwab lab computational expert - your python knowledge was greatly appreciated!

I would also like to thank the members of my thesis advisory committee - Detlev Arendt, Anna Kreshuk and Steffen Lemke - for their guidance on my project. Also, Alfons Riedinger, Vera Stankova, Helmuth Schaar and Arthur Milberger from the EMBL electronic and mechanical workshops for designing and making the semi-automated ultramicrotome system. Also, Maxim Polikarpov and Gleb Bourenkov for their excellent X-ray imaging of the *Platynereis* samples.

Finally, a big thanks to the people outside of my science life that have supported me throughout my entire PhD. In particular, my mum and my sister, who have been incredibly supportive, and now know more about *Platynereis* than they ever wanted to. A final thanks to my wonderful friends, who have been cheering virtually from the UK the entire time. Get your D20s ready, because we're all playing D&D when this is finally over.

GENETIC MAP OF PLATYNEREIS DUMERILII
Illustrated by Kimberly Meechan
0°
1°
2°
55°
54°
EXPLANATION OF MAP UNITS
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Cluster 10
Cluster 11
Cluster 12
Cluster 14

Genetic clustering of *Platynereis dumerilii* from chapter 4, illustrated in the style of an old geological map.

Contents

Chapter 1

Introduction

1.1 Maps and Atlases

If you ask me to imagine a map, the first thing that appears in my mind is a bright orange OS Explorer hiking map. For one, there's a large stack of them in the bookshelf next to my desk and second, they are one of the most popular maps in the UK. If you hike any of the big walking destinations here - like Dartmoor, Exmoor or the Lake District - you'll find they are a common sight sticking out of backpacks, stuffed in coat pockets or trodden into muddy ditches. Why are they so popular? The main reason is that they stuff a mountain of information into one convenient, pocket sized unit. Yes, they have the walking routes marked, but also camp sites, museums, pubs... It manages to bring together all these pieces of data and lock them into the uniform grid of its coordinate system. In this way, the spatial relationship between these data can be understood and used to plan out a really excellent hike.

While hiking maps are some of the most common maps to encounter, the definition of a map can be much broader. For example, there are climate maps that focus on the amount of rainfall in different parts of the world, or geological maps that focus on the type and orientation of rocks in different regions. Fundamentally, a map can include any kind of data at all, as long as it is being tied into a spatial coordinate system. Its purpose is to provide a resource that allows data to be interpreted within its spatial context.

Moving to biology, we've been making maps for decades of all kinds of systems. Here, the spatial coordinates change from being that of the Earth, to that of a particular model organism. North and South become anterior and posterior, east and west, dorsal and ventral. Our data layers become information from different methods e.g. light microscopy, electron microscopy, or sequencing. The aim remains the same though, to provide a resource that integrates data in a spatial way. The advantage, as with any traditional map, is to allow the relationships between different kinds of data to be understood in their spatial context. In a hiking map, this lets you link the location of car parks, paths and pubs to plan a good hike. In a biological map, this allows you to link e.g. morphology, location and gene expression to understand the formation of different cell types.

1.2 Biological atlases

Biological maps and atlases have been created for many different organisms, at many different resolution levels. For example, the Virtual Fly Brain[1] is an atlas at the tissue scale, focusing on the *Drosophila melanogaster* brain. It is provided as a browsable online resource[2], integrating data from many sources e.g. confocal images of GAL4 lines[3] and traced neurons from FIB-SEM datasets[4] and TEM datasets[5]. The spatial coordinate system for this atlas is provided by a number of brain 'templates' that represent the averaged size and morphology of the fly brain.

Moving to another brain focused atlas, the Allen Brain Atlas provides a browsable, online resource[6] for a number of species including mice and humans. For example, for the mouse, they provide spatial gene expression patterns for more than 20,000 genes (via in-situ hybridisation)[7]. This is tied to a standard spatial coordinate system provided by their mouse 'reference atlas'. This atlas is formed from histological sections taken at regular intervals through a mouse brain, annotated with details of the positions of different mouse brain structures.

Both these atlases are focused on the tissue scale of a particular model organism, but atlases can be produced at any scale. For example, the 'mitocheck' atlas focuses on the scale of individual cells, mapping the position of 28 different proteins over the course of human mitotic cell division[8]. Proteins were fluorescently labelled in HeLa cell lines, and imaged via 3D confocal microscopy. These spatial, and temporal gene expression patterns were then mapped to their standard 4D coordinate system, representing the average shape of a mitotic cell over time. This reference was formed by the averaging of images of many hundreds of cells undergoing mitosis. Again, as with the other atlases, this is provided as an interactive online resource[9].

Moving to a final example at a much larger scale, there are ongoing efforts to make atlases at the scale of entire organisms, rather than individual tissues. For example, the Human Cell Atlas is an international consortium of many different labs that aim to map cell types in different parts of the human body[10]. Their aims are wide-ranging, but again focus on combining different modalities (e.g. single cell RNA sequencing and multiplexed in-situ hybridisation) onto a standard coordinate system. The formation of this standard coordinate system is in fact one of their key goals, and to this end they've already published an entire paper discussing the challenges of defining coordinates for such a large and variable system as the human body[11].

To summarise, all of these atlases involve the integration of different kinds of data onto a standard, reference coordinate system. The resulting atlas is provided as a resource to the community, often in the form of a website or database, allowing others to use it as a map to guide their future work.

1.3 Benefits of electron microscopy integration

Electron microscopy (EM) uses a beam of accelerated electrons to produce high-resolution images of samples. It comes in two main types: transmission electron microscopy (TEM) and scanning electron microscopy (SEM). In TEM, samples must be sectioned very thinly, so the beam of electrons can pass through the sample and be detected on the other side. In SEM, samples can be much larger, and electrons are instead scattered (backscattered electrons), or trigger the emission of further electrons (secondary electrons). Converse to TEM, these electrons are detected on the same side of the sample i.e. they don't pass through the sample.

Why would we be interested in integration of EM into atlases? First of all, the main advantage of EM is the level of resolution it can achieve. SEMs can image at resolutions up to a few nanometres, while TEMs can achieve even sub-nanometre resolutions[12][13]. Therefore, for imaging small structures like individual synapses, or the fine structure of different organelles, EM is often the only technique with sufficient resolution.

The second advantage of EM is that it can provide the full spatial context of a tissue in an unbiased way - i.e. when you image you see the locations and orientations of all the cells present there. This is converse to something like fluorescence microscopy, where you image very precisely where a particular labelled structure is (e.g. one that expresses a certain gene), but often don't see the context around that structure.

This being said, EM also has a number of disadvantages that can be ameliorated by integrating it with other data in an atlas. For example, the strength I mentioned above, of seeing the full spatial context of a tissue, is in some ways also EM's greatest disadvantage. EM is unbiased, but this can make interpretation of images very difficult. Structures must be interpreted on their morphology alone, with no information as to what genes they express, or what proteins are present there. As a testament to this, the EMofCellsTissuesOrganisms slack group (which contains hundreds of electron microscopists), has an entire channel dedicated to identifying 'mysterious objects' in EM images[14]. Therefore, integration of EM with techniques that provide e.g. gene expression information, can be extremely powerful. The combination of these techniques can provide the spatial context of the tissue organisation and morphology, alongside details of the cells' molecular identity and function.

Another major disadvantage of EM, is the speed with which images can be acquired. For high resolution, EM sacrifices its field of view and acquisition times. This means that, in general, only a small number of individual samples can be imaged with EM, or even only small parts of a particular sample (depending on its size). For example, imaging an entire *Drosophila* brain with serial section TEM took about 16 months, and generated about 106 terabytes of data[5]. Therefore, integration of EM with other datasets in an atlas, can also help reduce this disadvantage. Higher throughput light microscopy methods can provide information about larger sample structures, in more individuals, with EM providing higher resolution information for select regions.

In summary, combining EM with other methods in an atlas can give you the best of both worlds. You gain the high resolution, and unbiased spatial and morphological data of EM, alongside information about cells' gene expression and functions. Light microscopy (and other techniques) can provide a higher throughput overview of a large number of individuals, over large tissue areas, supplemented by the highly resolved detail of EM where it is required.

1.4 Challenges of electron microscopy integration

As useful as EM can be when integrated with other data, there are many challenges that make this process very difficult. I mentioned one already above: the speed of acquisition. This means EM is usually limited to imaging only small regions, or only a few individuals.

Since acquisition speeds are so slow, we are usually required to be very targeted in which regions we acquire. This process of targeting a region for EM is also time consuming, complex, and involves many manual steps. In brief, the usual process involves trimming a resin-embedded sample with a diamond knife using a machine called an ultramicrotome. The orientation of the cutting is determined manually, by rotating different parts of the machine. This means that the precision is limited, and a reasonable amount of training is required to complete this process (see Chapter 2 for details).

Once the EM is acquired the next challenge begins, how to integrate this data? This depends entirely on the kind of data you are trying to integrate with. In general, the more similar the type of data, and the closer their resolutions, the easier the integration will be. For example, integration with X-ray imaging methods is often possible via intensity based image registration. This works as the kind of data they provide are quite similar - both provide unbiased information about the structure of a sample. Conversely, integration with some fluorescence imaging data can be much more difficult. In general, the resolution gap will be much larger, making it harder to identify corresponding structures between the two datasets. Also, the type of data is very different, with fluorescently labelled structures appearing as bright spots with no context about the structures around them. This means more complex means of integration are often necessary e.g. segmentation of structures that are present in both the EM and fluorescence images (nuclei are a commonly used example). For even more disparate data, e.g. single cell sequencing data, it may be necessary to bridge across even more modalities - for example, matching sequencing data to fluorescence images of in-situ hybridisations, and then to EM.

Once everything is integrated and the atlas is complete, the challenge of the large size of EM data becomes readily apparent. The *Drosophila* brain EM dataset I already mentioned[5], was 106 terabytes in size, making it very difficult to store and work with. These massive image sizes present challenges at all steps in the process, from initial storage, to data analysis, to finally sharing these images as a resource to the community. There is no way that members of the community can download such large datasets - so remote methods must be improved to allow them to access and browse these data.

1.5 Integrating EM into the lifecycle of a multimodal atlas

Most multimodal atlases (i.e. those that integrate more than one kind of data) undergo a similar 'lifecycle' (Figure 1.1). First, a standard coordinate system is formed, usually by averaging many images of the structure of interest (e.g. the templates I mentioned earlier for Virtual Fly Brain and the Allen Brain Atlas). Then, the required data is collected, and integrated into the common coordinate system. This integrated data is then analysed e.g. identifying cell types, or interesting spatial structures, before finally being shared as a resource to the community. At this point the life cycle becomes closed i.e. with a good reference coordinate system, data can be continuously collected, integrated and analysed, allowing the atlas to grow over time.

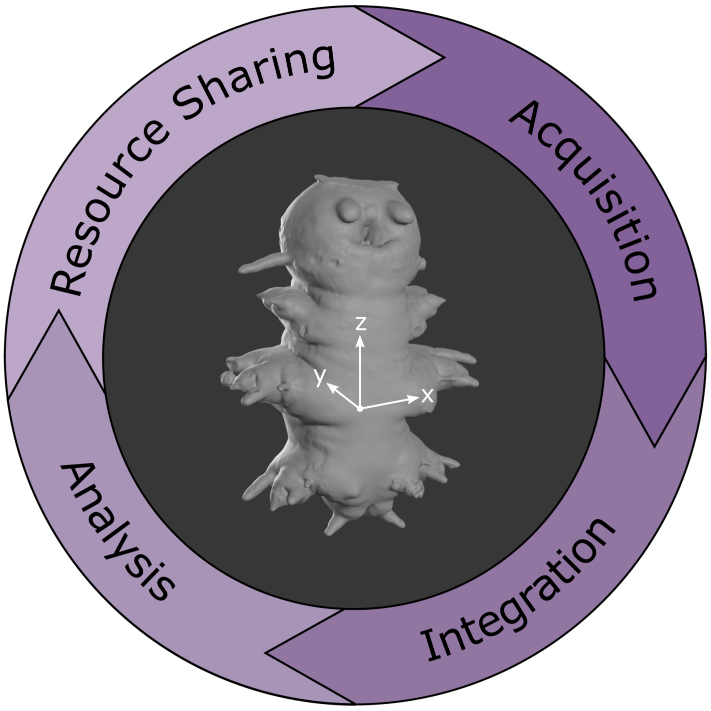


Figure 1.1: The lifecycle of a multimodal atlas

The benefit of these atlases is that they allow multiple forms of data to be brought together and understood in their spatial context. The addition of EM offers a number of advantages (including resolution, and unbiased structural data), at the cost of a number of challenges (e.g. the speed of acquisition and size of datasets). The aim of this thesis is to ease the integration of large-scale volume EM into multimodal atlases at various points within this lifecycle. First at the acquisition stage, I look at increasing the speed, ease and precision of targeted EM acquisition. This will allow easier selection of regions of interest, and faster acquisitions to allow more individuals or conditions to be compared. It will also ease integration into the reference coordinate system, by leveraging lower resolution X-ray data as an intermediate. Second, at the analysis stage, I look at making an analysis pipeline for comparing morphology (from EM) and gene expression for cell types in a large multimodal atlas. This will provide re-useable code for quantification of cell morphologies from EM, and comparison to genes on a large scale. Finally, at the resource sharing stage, I look at improving the sharing and exploration of these massive EM datasets, by contributions to the software MoBIE that allows interactive browsing of very large, remotely stored image data.

Chapter 2

Acquisition: faster targeting with an ultramicrotome

2.1 Background

2.1.1 Why target?

As discussed in chapter 1, EM is a time consuming process. For example, for the *Platynereis dumerilii* atlas I will discuss in chapter 4, collecting one entire individual with serial block face SEM took 7 weeks of continuous acquisition[15]. This means it is rarely feasible (in terms of time and cost) to image the whole specimen with EM.

It also means that our sample sizes are usually very small. Indeed, often only one individual is imaged, especially for very large datasets. For example, it is very common for connectomics datasets that map large brain regions to be released with only one individual imaged[5][4]. These small sample sizes can make it difficult to assess how much variation may be present between individuals or conditions, and also creates difficulty in quantifying observations in a statistically significant way. Therefore, we need to find ways to reduce the acquisition times for EM.

There are many ways to do this, but perhaps the most common is targeted acquisition i.e. rather than attempting to acquire the entire sample, we instead focus on smaller regions of interest for EM. Of course, this speed up relies on there being scientific questions that can be answered with a smaller region of interest. For example, the connectomics datasets that I mentioned previously would not be good candidates for this - as they wish to understand the connectivity of the brain *as a whole*. In contrast, questions that focus on particular cell types or tissues, could benefit greatly from targeted acquisition.

2.1.2 How to target with an ultramicrotome

For 3D EM at room temperature, samples usually undergo a series of preparation steps (including chemical fixation and the addition of heavy metals), before being embedded into resin blocks (Figure 2.1). These blocks are then trimmed to the region of interest using a machine called an ultramicrotome.

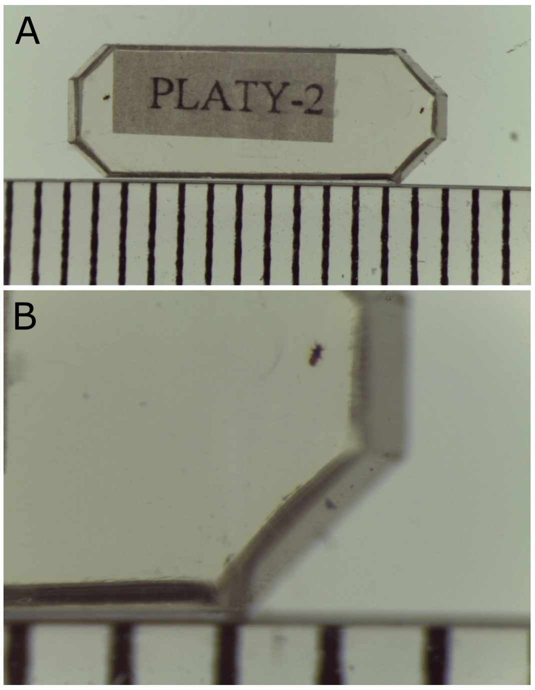


Figure 2.1: Resin embedded samples. ***A*** - two *Platynereis* embedded at opposite ends of a resin block. Ruler at the bottom has divisions of 1mm. ***B*** - zoom of right end of block in *A*

Ultramicrotomes are devices used to cut very thin slices from samples with a diamond knife (Figure 2.2). They are produced by a number of manufacturers (with slight differences), but the main components remain the same for all. First, there is a knife holder - this holds the diamond knife firmly in place during cutting. It can be adjusted to an optimal position by moving forwards and backwards (referred to as North and South) and left and right (referred to as West and East). It can also be rotated to change the angle of the knife. Second, there is a sample holder - this holds the resin-embedded sample firmly in place during cutting. It can also be adjusted to the optimal position - here by rotating two different knobs to rotate the holder along two separate axes (see Figure 2.2). This sample holder is attached to a moving arm, which performs the cutting motion. To cut, the knife remains at a fixed position (secured by the knife holder), and the sample is moved up and down (via the arm) to cut slices. Each time the arm moves down, and cuts a slice, it moves South slightly. The amount it moves South, and therefore the thickness of each slice, can be adjusted very precisely (in the tens of nanometre range). The arm can move South a certain total distance (termed the feed length) before it must be reset to its initial position. The movement of the arm can be controlled manually by rotating a wheel at the side of the ultramicrotome, or via an electronic control panel. This panel allows you to set a cut thickness and desired depth, and have the microtome automatically cut that distance. The final key part of the ultramicrotome is a binocular microscope. This is a normal light microscope that allows you to see the sample and knife during cutting at high magnification.

With this setup, the operator places their block in the sample holder, and uses the view down the binoculars to manually adjust the 3 axes: knife tilt, sample tilt and sample rotation. Once the orientation is set, they move the knife close to the sample, and clamp it in place, before starting the cutting cycle at the desired thickness.

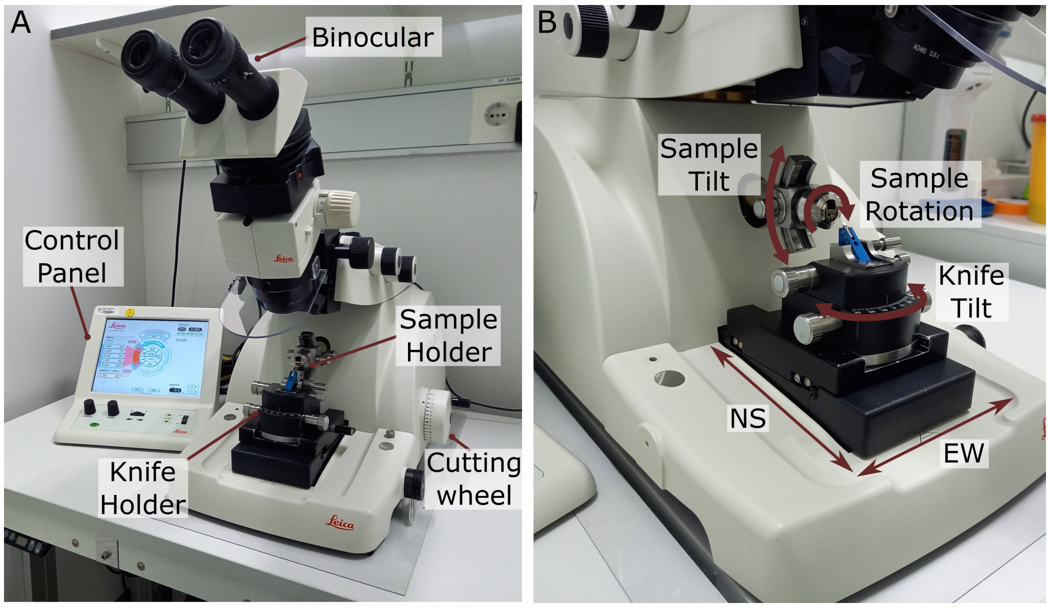


Figure 2.2: Leica UC7 Ultramicrotome. ***A*** - labelled parts of a Leica UC7 Ultramicrotome. ***B*** - Zoom of the same ultramicrotome as *A* showing directions of movement and rotation of different parts.

2.1.3 Difficulties of targeting with an ultramicrotome

Targeting a specific region of interest with an ultramicrotome is a difficult process, and requires a lot of time and training to master. The first major difficulty is that you must rely on the features you can see down the binocular to guide how deep, and at what angle to cut. For example, if you are targeting a structure deep inside your sample, then the external features of the resin block or the sample itself may give you few clues as to the position of your region of interest. Also, in the sample preparation process, the addition of the heavy metal osmium can make the sample turn a very dense black colour, which also makes it difficult to see sample features that could guide your cutting. In addition, the possible viewing angles with the binocular are quite limited, making it difficult to assess the position of the sample in the block.

Another difficulty is that while the cutting motion of the microtome is automated, the initial positioning of the knife and the sample is not. Therefore, even with good features visible down the binocular, the operator must manually adjust each axis to reach the right orientation to hit the region of interest. This kind of 3D thinking is very difficult, and takes a lot of time and training to become proficient with. The example I always give is that of the rubik's cube: here the difficulty also lies in understanding how rotations about 3 different axes will affect the position and orientation of individual pieces. This becomes second nature with time, but is a great challenge to newcomers.

All these aspects mean that targeting with an ultramicrotome has limited precision and requires a lot of training and practice.

2.1.4 Targeting with external 3D maps

There are two main ways that have been explored to make ultramicrotome use easier, faster and more precise. The first is the use of external 3D maps of the sample (e.g. from X-ray or light microscopy) to guide the targeting. The second is automation of the microtome itself.

Let's first consider the use of X-ray imaging. X-ray computed tomography (CT) can provide detailed 3D maps of a sample quickly, and in a non-destructive way[16][17]. For example, laboratory micro-CT scanners are becoming ever more common, and can reach voxel sizes of around one micron, on a timescale of a few hours (depending on the system). In addition, they don't require any extra sample preparation steps - resin blocks prepared for EM can be scanned directly[18][19][20][21], as long as they are trimmed small enough to fit inside the particular micro-CT scanner you are using.

In brief, micro-CT works by taking 2D projections of a sample at many different angles. To enable this, the sample is usually placed on a rotating stage, and alternately rotated and imaged to collect a series of 2D images. These 2D images are then reconstructed into your final 3D stack (often referred to as a tomogram).

If you require higher resolution than a laboratory scanner can offer, there is also the possibility of using a synchrotron facility to image samples. Synchrotrons are massive, ring-shaped particle accelerators that accelerate electrons to very high speeds. They are extremely powerful sources of X-rays, and can provide resolutions far superior to laboratory X-ray sources (in the range of hundreds or even tens of nanometres) [16][21][19]. They can also be much faster (depending on the required resolution), scanning in the range of seconds or minutes[16]. This being said, they come at the disadvantage that there are only a few of these facilities worldwide, and access is much more limited than laboratory sources.

X-rays have been combined with EM for a number of uses. For example, for long EM acquisitions they can be used as an initial quality control check that the sample is intact and undamaged. Scheffer et al.[22] use X-ray imaging to check each piece of prepared *Drosophila* brain before imaging with FIB-SEM. It has also been used to aid targeting of regions of interest. For example, Karreman et al.[18], [23] use a laboratory micro-CT machine to create 3D maps to guide their targeting at an ultramicrotome. They correlate light microscopy imaging of fluorescent tumour cells in a mouse brain, to X-ray images of the same sample prepared for EM and embedded in resin. This allows them to precisely measure the depth from the surface of their resin block to the cancer cell (their region of interest), and use this to guide their trimming at the ultramicrotome. Yet, as they don't take into account how the block will be mounted in the microtome, or the final angles of each of its axes, this distance is not the same as the actual cutting distance (i.e. N-S in the microtome). Therefore, while cutting they had to pause at regular intervals to take light microscopy images of thick sections of the sample to check their progress. This allowed good precision (<5 micron), but slowed down the process and required more manual steps. Also, while they measure the distance from the X-ray, there is no automation of the angle of trimming, so this still must be set manually by an experienced operator.

Another example is Xu et al.[24] who again use a laboratory X-ray source to scan resin-embedded samples and select their regions of interest. They again use this to to guide trimming with the ultramicrotome, by trimming in steps and re-scanning with X-ray at intervals to check their progress. In addition, Musser et al.[25] use both a laboratory X-ray source and X-ray imaging at a synchrotron to guide trimming at an ultramicrotome and imaging with FIB-SEM of neuroid cells in a sponge.

There are also many examples of using 3D light microscopy imaging data to guide ultramicrotome trimming. For example, Ronchi et al.[26] use confocal imaging of fluorescently labelled samples embedded in resin blocks. They then use measurements of the depth to the region of interest to guide their trimming at the ultramicrotome. Again, this is done as an iterative process, of trimming and re-scanning in the confocal to monitor the progress towards the region of interest.

Another example is Brama et al.[27] who also image fluorescently labelled samples embedded in resin blocks. Here, they integrate a miniaturised light microscope directly onto the ultramicrotome to allow direct imaging of the fluorescence. In this way they could visualise fluorescently labelled nuclei and monitor their depth from the surface as the microtome cuts. This allows very precise targeting in depth.

To summarise, all these examples use 3D imaging (X-ray or light) to guide trimming at an ultramicrotome. Usually, this is done by measuring the depth of regions of interest from the surface of the resin block, followed by manual trimming at the ultramicrotome. As most of these methods don't take into account the final position of the sample in the ultramicrotome or its orientation, accuracy is limited and trimming must be done in iterative steps. This slows the process down and requires more rounds of X-ray or light imaging. In addition, all these methods focus mainly on the depth to a region of interest and not the orientation of cutting - this must still be set manually by the operator.

2.1.5 Automation of the ultramicrotome

The second way to increase the ease and speed of targeting with an ultramicrotome is to introduce more automation. So far, most attempts at ultramicrotome automation have focused on serial sectioning for either TEM or array tomography. Serial sectioning involves the collection and imaging of a series of 2D sections from a sample to reconstruct a 3D volume. It is a very difficult and manual process, where fragile thin sections must be collected by hand and their sectioning order maintained to allow final reconstruction in 3D.

For example, ATUMtome[28] allows cut sections to be automatically collected onto reels of tape for later imaging with an SEM. Conversely, MagC[29] uses magnetic resin to allow sections to be automatically concentrated with a magnet and collected onto silicon wafers for imaging with SEM. The section order is retrieved automatically by comparison of fluorescent beads embedded in the magnetic resin of each slice. Finally, LASSO[30] uses manual control of a motorised setup to collect sections in a metal loop, followed by automatic deposition onto silicon/silicon nitride TEM substrates.

All these methods speed up the collection of serial sections but, so far, few have focused on automation of the initial block trimming or targeting of regions of interest. One of the few examples is the study I previously mentioned[27] where a mini fluorescent microscope was integrated with an ultramicrotome to provide greater automation of targeting. Again though, this focuses only on depth of targeting, and requires fluorescently labelled samples.

2.1.6 Aims

My aim was to make targeting with an ultramicrotome easier, faster and more precise. To do this, I aimed to make an integrated workflow to go from X-ray imaging of a sample to a list of angles for the three microtome axes, and a cutting distance. This would be implemented as software to allow precise measurement of the X-ray, and calculation of the required angles and distances. To increase the precision of targeting, I also aimed to introduce more automation in the form of motors for each of the three axes: knife tilt, sample tilt and sample rotation.

2.1.7 *Platynereis* as a model system

For this chapter, and all following, I have chosen to use the marine annelid *Platynereis dumerilii* as the model system. *Platynereis* is an established model system for development, evolution and neurobiology, and can be bred easily in laboratory conditions[31][32][33]. It offers many advantages as a test case for developing tools to integrate EM into multimodal atlases. For one, the 6 day old *Platynereis* (the stage I use throughout) is of a relatively small size, containing around 11,000 cells in total[15]. This means it has an interesting variety of different cell types, while still being of a manageable size for EM. In addition, its development is highly stereotypical[34][35] - making it possible to integrate data from many different individuals into one common coordinate system. Finally, there are already many resources available for *Platynereis* - for one, a large atlas of in-situ hybridisations (for $>$ 200 genes) known as 'ProSPr' (Profiling by Signal Probability mapping)[36].

2.2 Results

2.2.1 A semi-automated microtome

First, a number of adaptations were made to a standard Leica UC7 Ultramicrotome to allow more precise angular movement (Figure 2.3). These changes were designed and implemented by: Alfons Riedinger, Vera Stankova, Helmuth Schaar and Arthur Milberger (of the electronic and mechanical workshops at EMBL), with guidance from Yannick Schwab and myself. A motor was added to each of the rotational axes (knife tilt, sample tilt and sample rotation) controlled by a small raspberry pi computer. The angle of each of these axes can be precisely read, and controlled via a touchscreen interface.

These axes were calibrated by measuring the angular movement in degrees against the total number of motor steps. In this way, the motors can be used to precisely go to any angle.

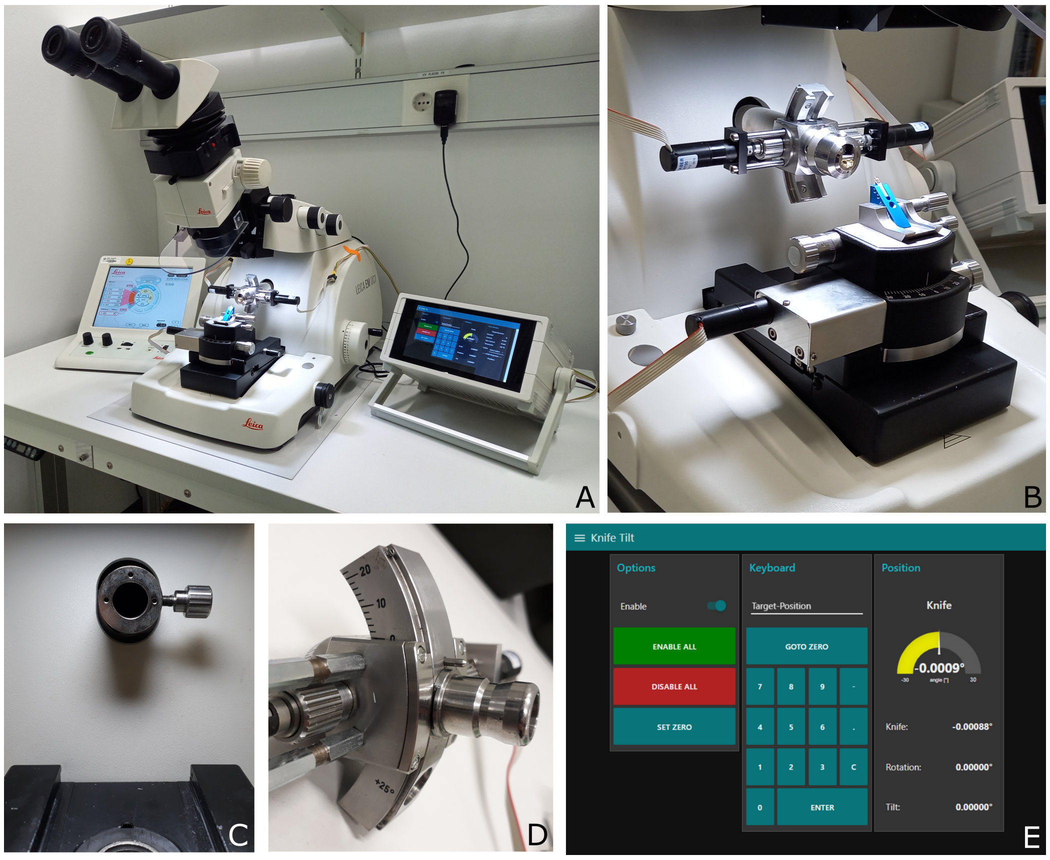


Figure 2.3: Semi-automated Leica UC7 Ultramicrotome. ***A*** - overview of semi-automated ultramicrotome. ***B*** - Zoom of the same ultramicrotome as *A* showing the motors attached to each of the rotational axes. ***C*** - ultramicrotome arm with sample holder removed. This shows the additional hole that was drilled (here there are 3 holes, but the top one is the only one required) ***D*** - zoom of back of the sample holder, showing the small metal pin that was added to fit into the hole in *C*. ***E*** - Knife tilt page of the user interface on the touchscreen to the right of the ultramicrotome. Motors can be enabled/disabled from here, and angles entered to move the motors to different positions.

2.2.2 Targeting workflow

The targeting problem can be broken down into two steps: that of orientation, and that of distance. For the orientation problem, we must find a combination of 3 angles (knife tilt, sample tilt, sample rotation) that bring the target plane to be vertical, and parallel to the knife. For the distance problem, we must find the depth to cut from the sample to reach the target plane.

Coordinate frames

To solve both of these problems, I first needed a way to describe coordinates in a standard way for both the microtome, and the sample. For the microtome, I therefore defined orthogonal coordinate frames for each of the components - one for the microtome as a whole (referred to as the world coordinate frame), one for the knife, one for the arc piece, and one for the sample holder (figure 2.4). The world coordinate frame is fixed, and defines x as parallel to the east-west movement of the microtome, y as parallel to the north-south movement, and z as vertical. The other coordinate frames can be imagined as attached to their respective components, moving with them as they move (figure 2.4B-D).

For the sample, the coordinate frame was more difficult to define. Resin blocks can be many different shapes and sizes, and contain a wide variety of samples. This means it is difficult to define a standard coordinate system that can be applied to them all. To solve this, I required an initial trimming step at the microtome to produce a reference face that could define the coordinate system. Here, a flat face is trimmed into the block (with a diamond knife), and trimmed to have four perfectly straight sides. These edges can be in any orientation, allowing some flexibility to the microtome operator. The coordinate frame is then defined with x running from left to right along the bottom edge of this reference face, z vertical within the reference face, and y pointing away from the surface (i.e. the reference face's normal) (Figure 2.4E). This trimmed face is very similar to the standard procedure for trimming resin embedded blocks for sectioning at the ultramicrotome, so should be easy to execute and familiar to anyone who has used a microtome previously.

Finally, I needed a coordinate frame for the target plane within the sample. This I defined with a set of two rotations (θ_{to} and θ_{tr}) with respect to the block coordinate frame. See figure 2.5 for details of how these are defined.

Orientation

Finding a general solution to the orientation, is actually a very similar problem to that which must be solved in robotics to e.g. control a robotic arm. Here, they construct a 'forward kinematics' equation that describes the position of the end effector (e.g. a gripper) in terms of the input angles (i.e. the angles of each individual part of the arm). They then create an 'inverse kinematics' equation that can calculate the input angles, given the desired final position of the end effector.

In effect, we must do the same. Our input angles are the knife tilt, sample tilt and sample rotation, and we must understand how these 3 angles work together to give the final position of our target plane (forward kinematics). Equally, we must then be able to, given a target plane, calculate the three input angles that can allow us to reach it (inverse kinematics).

The forward kinematics equation is constructed by calculating the rotation matrices that relate each of the individual coordinate frames (Figure 2.4). Here, I only focus

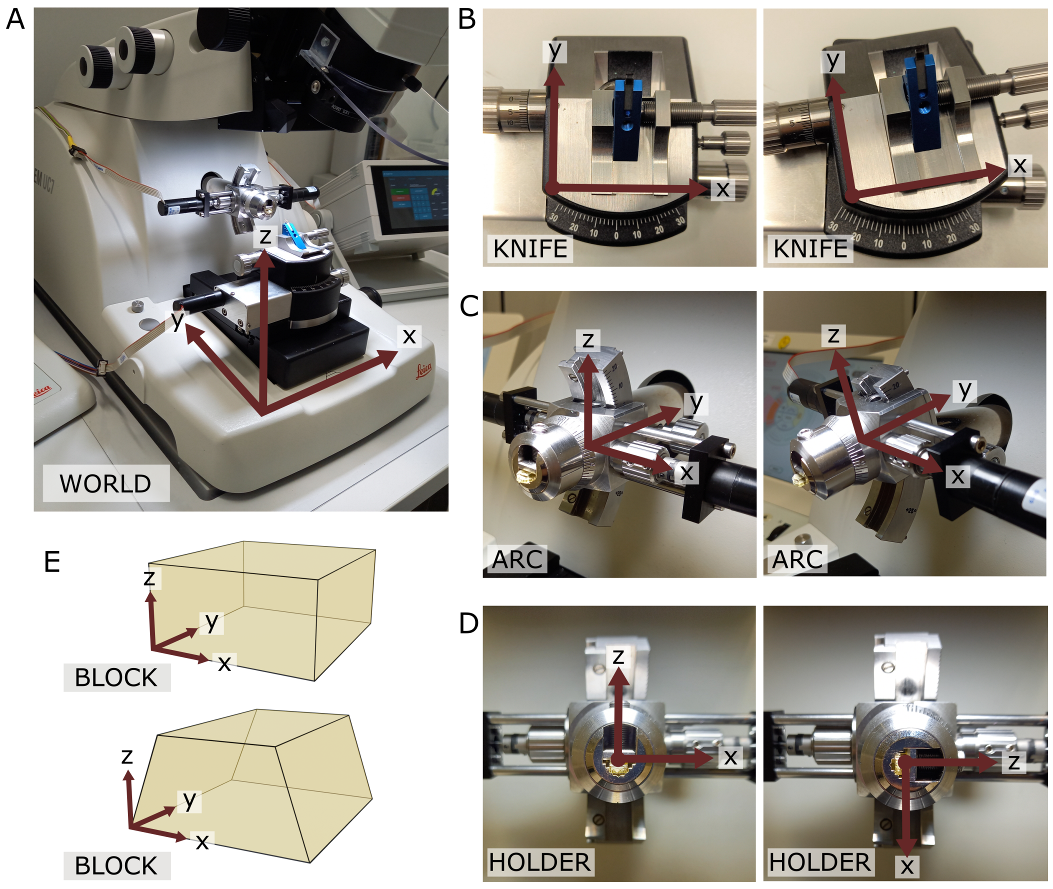


Figure 2.4: Coordinate frames. ***A*** - world coordinate frame. X is parallel to EW, y is parallel to NS, z is vertical. ***B*** - Knife coordinate frame at 0 degrees (left) and 10 degrees(right). Z points out of the page, and is parallel to the world coordinate frame z. Note that at 0 degrees the knife frame has the same orientation as the world frame. ***C*** - Arc coordinate frame at 0 degrees (left) and 14 degrees (right). X is parallel to the world coordinate frame x. Note that at 0 degrees the arc frame has the same orientation as the world frame. ***D*** - Holder coordinate frame at two positions separated by a 90 degree rotation. Note that here the 0 position depends on the particular targeting run, as it is defined as the rotation when the knife and block face are aligned. The y axis points into the page and is parallel to the arc coordinate frame y axis. ***E*** - Block coordinate frame for a rectangular block face (top) and a trapezium block face (bottom). X is parallel to the bottom edge of the block face.

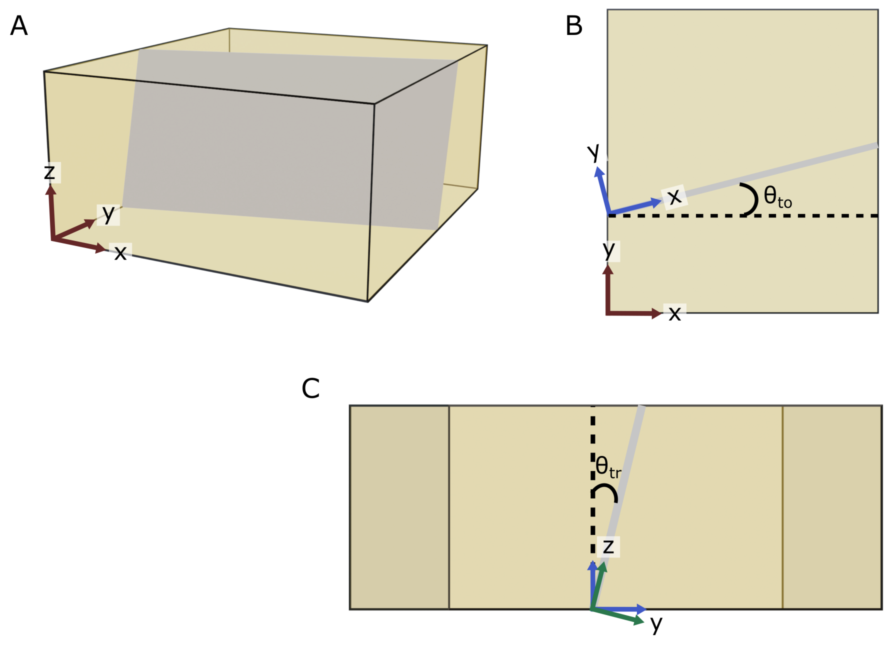


Figure 2.5: Block and target coordinate frames. ***A*** - diagram of a rectangular block, with the block coordinate frame shown in red. The gray plane inside the block is the target plane. ***B*** - Considering the xy plane only, the target plane intersects along the shown gray line. θ_{to} is then defined as the angle between the block frame x axis and the blue x axis shown i.e. the rotation angle about the block frame z. ***C*** - Now considering the zy plane of the blue axes from *B* (i.e. looking down the line of intersection), θ_{tr} is defined as the angle between the blue and green z axes i.e. the rotation angle about the blue frame x. The green axes are the final target coordinate frame.

on the orientation, and not the relative position - as the position will be solved in a separate process. I've constructed a pose diagram that shows the relation between these coordinate frames in figure 2.6A. This process is complicated for us, as the rotation to move from the sample holder to the block face is unknown. The sample holder contains a simple clamp that holds the resin block in place - the orientation of the block will therefore be slightly different every time it is placed inside.

How do we find the relation between the holder and the block face then? This can be solved by adding an additional step where the knife is aligned to the block face - this is a standard procedure when using an ultramicrotome, so again should be easy and familiar to anyone who has a used a microtome previously. This process is done manually by looking down the microtome binoculars, and adjusting the orientation of the three axes to bring the knife parallel to the flat block face, and the bottom edge of the block parallel to the knife edge.

Once this is complete, I know that the orientation of the knife coordinate frame, and the block coordinate frame are the same. This means I can calculate the rotation matrix between the holder and the block by going the 'long way round' i.e. from the Holder

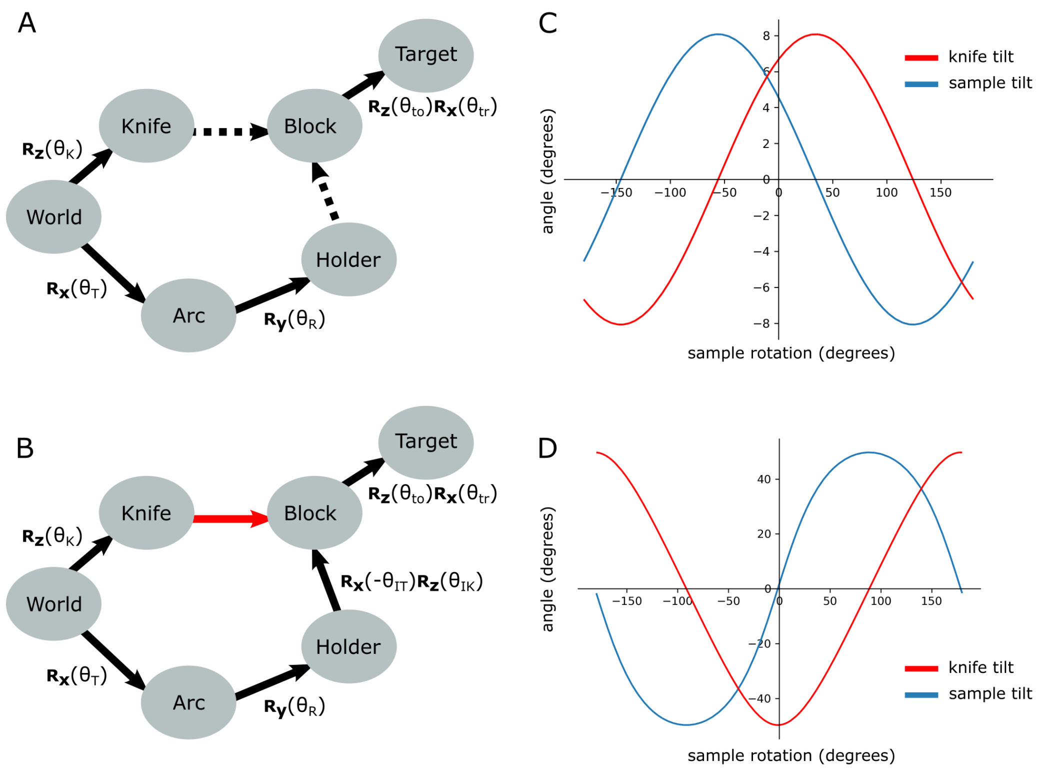


Figure 2.6: Pose diagrams and solutions. ***A*** - pose diagram showing relations between coordinate frames. $\boldsymbol{R_x}$, $\boldsymbol{R_y}$ and $\boldsymbol{R_z}$ are rotation matrices about the x, y and z axis respectively. θ_T is the sample tilt angle, θ_R the sample rotation angle, θ_K the knife angle, θ_{to} the target offset angle, and θ_{tr} the target rotation angle. Dashed arrows are unknown relations. ***B*** - pose diagram when the knife is aligned to the block face. Here, the knife coordinate frame is in the same orientation of the block (red arrow), and therefore we can infer the Holder to Block relation that was unknown in *A*. θ_{IT} is the initial sample tilt angle and θ_{IK} is the initial knife tilt angle, when the knife and block are aligned. We define θ_R to be zero at this aligned orientation. ***C*** - graphs of knife and sample tilt solution for all sample rotation angles. This used values of θ_{IK} = 10, θ_{IT} = 10, θ_{to} = -3.3 and θ_{tr} = 5.4. ***D*** - solution graphs with more extreme initial angles. θ_{IK} = 10, θ_{IT} = 10, θ_{to} = -60 and θ_{tr} = 5.4.

to the Arc, to the World, to the Knife and then to the Block (Figure 2.6B). I can make this even simpler by defining the holder rotation to be zero in this aligned position.

As such, this means that the full forward kinematics equation, describing the rotation from the World coordinate frame to the target plane becomes:

$$\boldsymbol{F} = \boldsymbol{R_x}(\theta_T)\boldsymbol{R_y}(\theta_R)\boldsymbol{R_x}(-\theta_{IT})\boldsymbol{R_z}(\theta_{IK})\boldsymbol{R_z}(\theta_{to})\boldsymbol{R_x}(\theta_{tr})$$

where $\boldsymbol{R_x}$, $\boldsymbol{R_y}$ and $\boldsymbol{R_z}$ are rotation matrices about the x, y and z axis respectively. θ_T is the sample tilt angle and θ_R is the sample rotation angle. θ_{IT}, θ_{IK}, θ_{to}, and θ_{tr} are constants for a particular targeting run representing the initial sample tilt angle, the initial knife tilt angle, the offset angle between the block face and the target plane, and the rotation angle between the block face and the target plane respectively (see Figure 2.5 for details of how θ_{to}, and θ_{tr} are calculated).

This equation allows us to accurately describe the orientation of the target plane for any set of input angles. Now, we must calculate the required inverse kinematics equation from this. For us, the constraint to satisfy is that the target coordinate frame's y axis (which is the same as the target plane's normal) must be perpendicular to the world z axis. This would mean that the plane is vertical and, as the knife cuts vertically, can be reached by the knife (as long as it is within the angle range it can reach). This can be framed mathematically as:

$$(\boldsymbol{Fy}) \cdot \boldsymbol{z} = 0$$

were F is the forward kinematics equation defined above, $\boldsymbol{y}$ is the y vector $\left(\begin{smallmatrix}0\\1\\0\end{smallmatrix}\right)$ and $\boldsymbol{z}$ is the z vector $\left(\begin{smallmatrix}0\\0\\1\end{smallmatrix}\right)$. $\boldsymbol{Fy}$ gives the vector for the target plane y axis in terms of the world coordinate frame. Constraining this so its dot product with the world z axis must be 0, means it must be perpendicular to world z i.e. the plane must be vertical.

Solving this equation gives the solution:

$$\theta_T = \arctan\Big(C_1\cos(\theta_R) + C_2\sin(\theta_R)\Big)$$

which, as expected, means there are many possible solutions. In fact any point on this line (see figure 2.6C) is a possible solution (as long as each of the angles stays within the constraints of the microtome e.g. within +/- 20 degrees for the sample tilt).

Here, C_1 and C_2 are constants for a particular targeting run, depending on the exact location of the target plane, and the position of the block in the microtome. See section A.1 in the appendix for details.

We can also solve for the knife angle in a similar manner, giving a solution of the form:

$$\theta_K = \arctan\left(\frac{C_1\Big(C_2\cos(\theta_R) + C_3\sin(\theta_R)\Big)}{\left(\sqrt[2]{{C_1}^2 + \Big(C_2\sin(\theta_R) - C_3\cos(\theta_R)\Big)^2}\right)|C_1|}\right)$$

where C_1, C_2 and C_3 are constants for a particular targeting run (note: these are different constants to the sample tilt equation above). See section A.2 in the appendix for details.

These two equations allow the calculation of the tilt and knife angle, given a particular rotation, and therefore provide a general solution to the orientation problem. Note that there will be some situations where no solution is possible due to the constraints of the microtome axes e.g. in figure 2.6D, the sample tilt of some solutions exceeds the maximum tilt of 20 degrees. In these situations, the operator would need to remount the specimen, or re-trim it to aim for an angular difference within the microtome constraints.

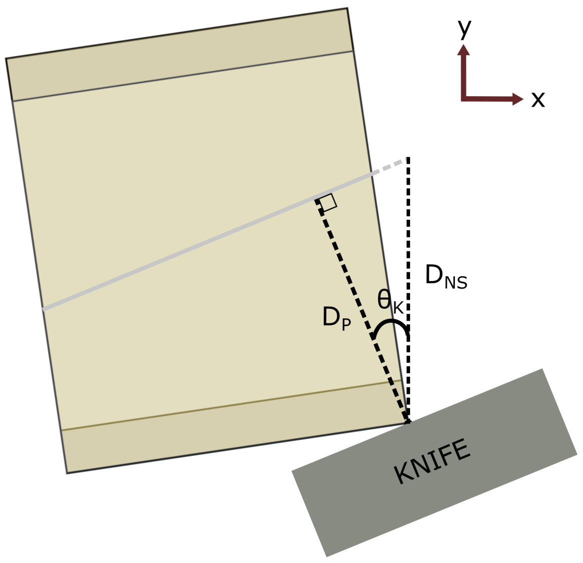


Figure 2.7: Distance calculation. Here, the same block and target plane as in figure 2.5 are shown in an example solution orientation. In red is the world coordinate frame, showing we are looking down the world z axis. The knife is aligned to the vertical target plane, but moves in the world y direction (NS) to cut. D_P is the perpendicular distance between the target plane and the furthest surface point, D_{NS} is the corresponding NS distance, and θ_K is the knife angle.

Distance

Once the orientation is solved, I need to calculate the distance to cut from the sample to reach the target plane. As the cutting direction (NS) may not be parallel to the target plane normal, this must be done in two steps. First, the perpendicular distance is calculated from the target plane to the furthest point of the block face, then I compensate for any offset between this and NS.

When the orientation is solved, I know that the target plane must be vertical. Therefore, to find the NS distance I only need to compensate for the current knife angle (Figure 2.7).

This means the distance is:

$$D_{NS} = \frac{D_P}{cos(\theta_K)}$$

where D_{NS} is the NS distance, D_P is the perpendicular distance between the target plane and the furthest surface point, and θ_K is the knife angle.

2.2.3 Crosshair

The calculations in the previous section allow the orientation and distance to be solved in a general manner. The next step was to make this workflow accessible to anyone who works in electron microscopy - even those with no programming or image analysis experience.

To enable this, I developed a Fiji[37] plugin called Crosshair that integrates all the individual steps into one workflow. This plugin builds on top of BigDataViewer[38] (to allow browsing of large images as 2D slices), and the imageJ 3D Viewer[39] (to allow browsing of images in 3D).

The user interface is made up of three main panels: the controls, a 2D viewer (BigDataViewer) and a 3D viewer (ImageJ 3D Viewer) - see figure 2.8.

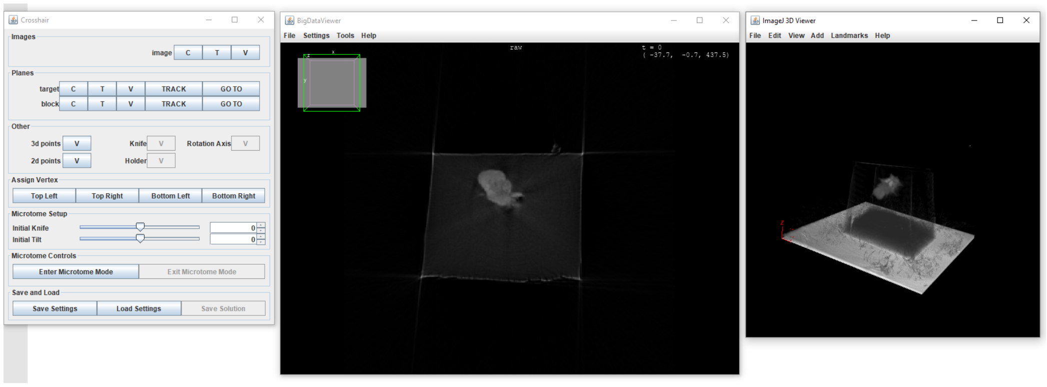


Figure 2.8: Crosshair ui - the left panel is the controls, the middle panel is the 2D Viewer and the right panel is the 3D viewer.

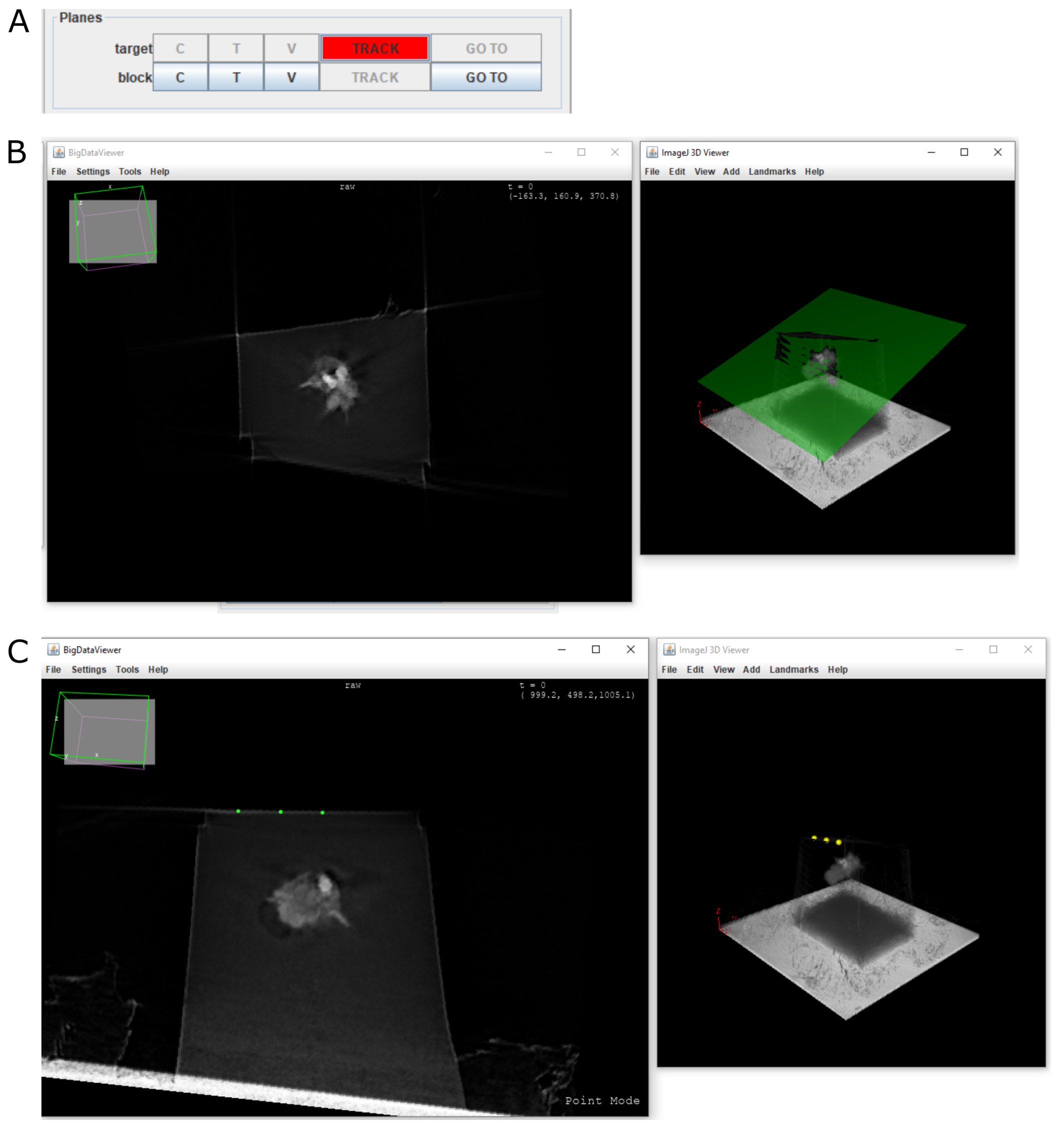


Figure 2.9: Planes in Crosshair. ***A*** - planes can be created by using the 'TRACK' buttons in the planes menu. ***B*** - when 'TRACK' is pressed, navigating in the 2D viewer (left), creates a corresponding plane in the 3D viewer (right). ***C*** - points can be placed by clicking in the 2D viewer (left), and will be displayed in the 3D viewer (right). Here points are placed on the block surface to fit a plane. All panels use an example X-ray of a *Platynereis dumerilli* embedded in a resin block.

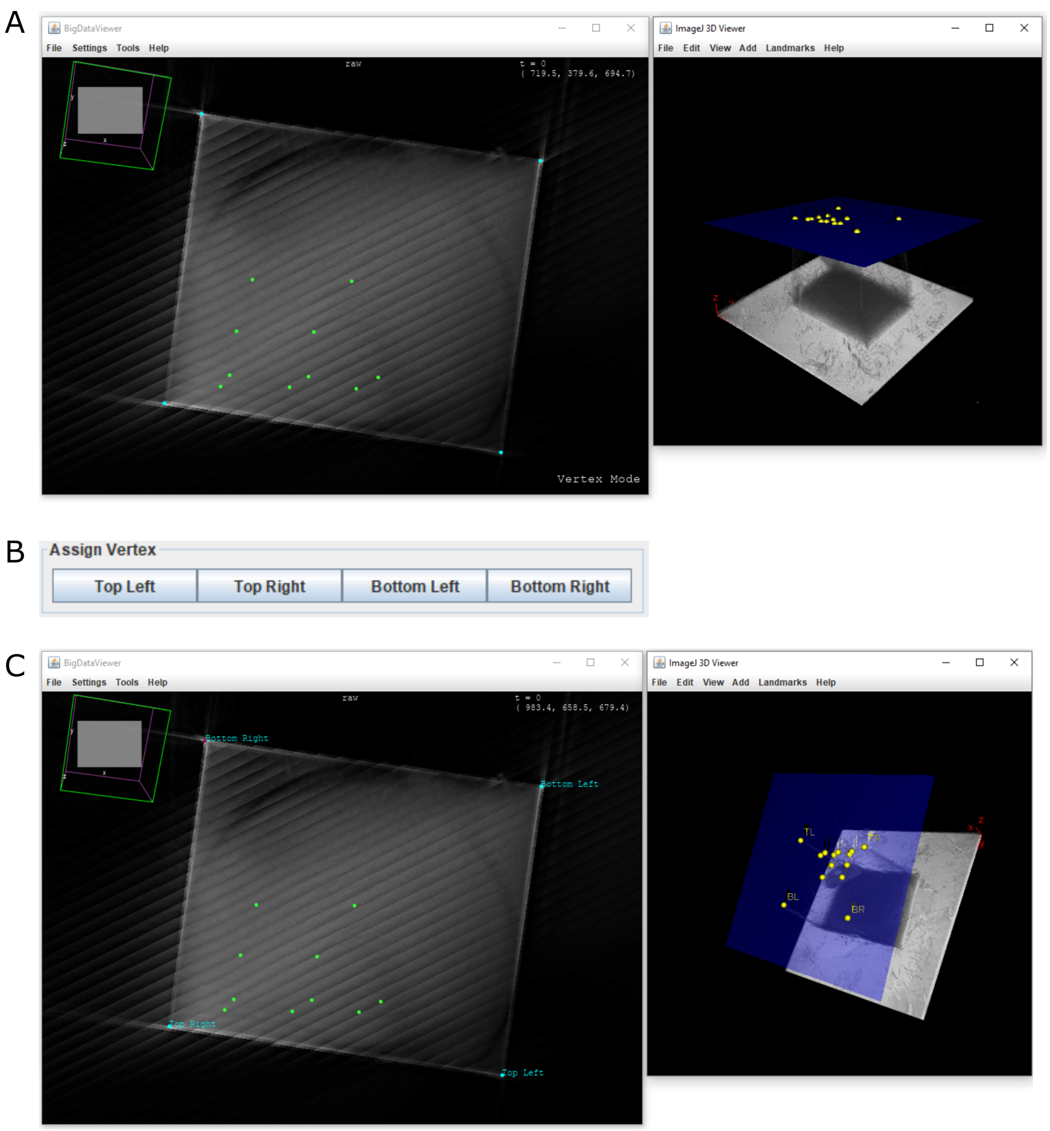
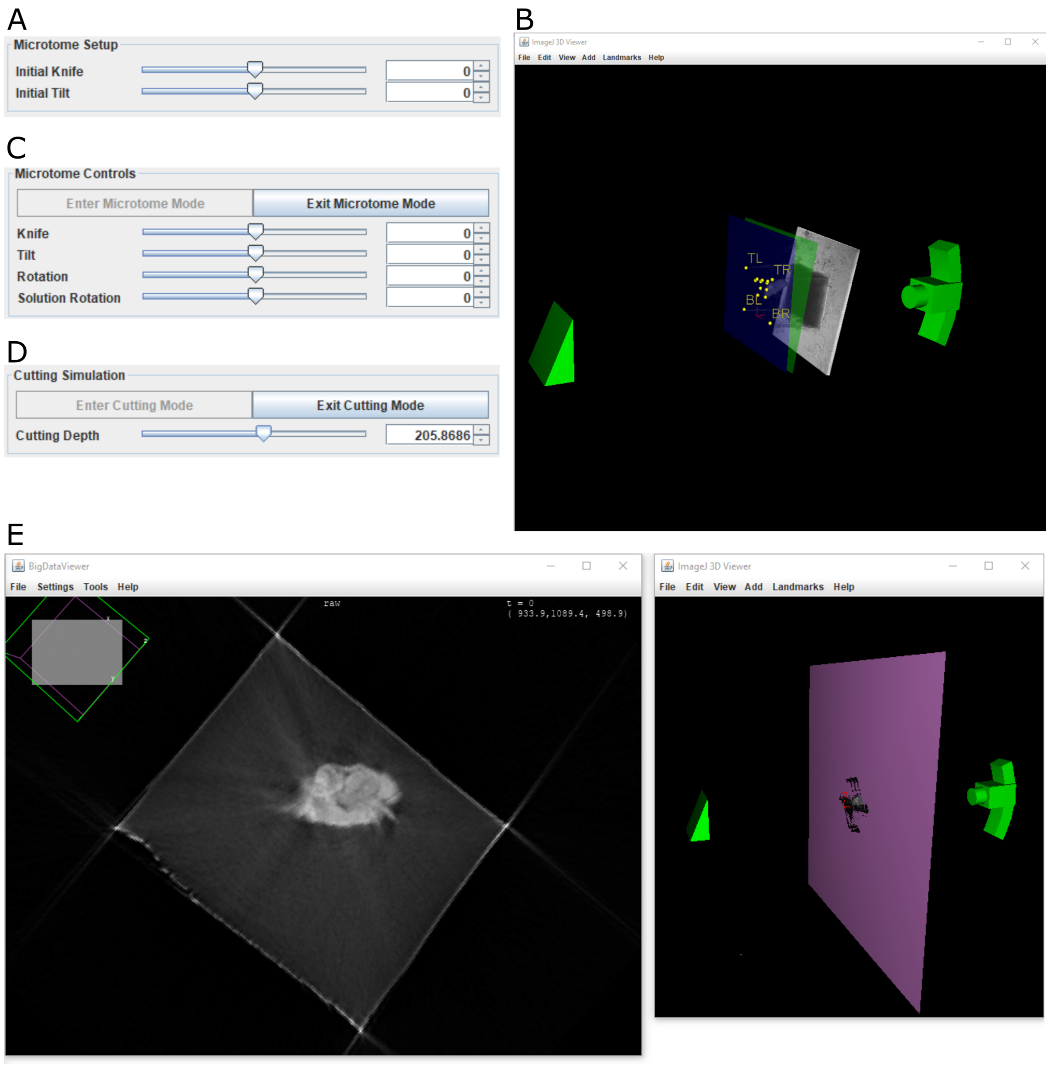


Figure 2.10: Vertices in Crosshair. ***A*** - block surface shown in 2D (left) with points used to fit the plane in green, and vertices marked in blue. Right - same block in 3D, with block face as a blue plane. ***B*** - the orientation of the block can be set by selecting vertices and assigning them via the Assign Vertex menu. ***C*** - same as *A*, with orientation of vertices marked.



Initial Knife 0
Initial Tilt 0

Knife 0
Tilt 0
Rotation 0
Solution Rotation 0

Cutting Depth 205.8686

```
t = 0
( 933.9,1089.4, 498.9)
```

Figure 2.11: Crosshair microtome and cutting mode. ***A*** - the initial knife angle θ_{IK} and initial sample tilt angle θ_{IT} can be entered via sliders. ***B*** - 3D viewer in microtome mode. A very simple representation of the microtome is shown in green (left is the knife, and right is the arc and sample holder), along with the X-ray scan and the points/planes. ***C*** -controls for microtome mode. The three axes of the microtome - knife tilt, sample tilt and sample rotation can be controlled via sliders. The final slider, 'solution rotation' allows possible solutions to be cycled through and selected. ***D*** - controls for cutting mode. ***E*** - 2D and 3D viewer in cutting mode. Left shows an example slice through the X-ray of a *Platynereis*, and right shows the corresponding cut position (pink plane).

It allows X-ray images to be viewed in both 2D and 3D and both the block face and target plane to be marked easily. Planes can be created in two ways: by navigation in the 2D viewer, and by fitting to a series of points (Figure 2.9). The corners (vertices) of the block face can also be easily marked, and the orientation of the block in the microtome indicated (Figure 2.10).

Once the various planes and points are marked on the X-ray, you can enter 'microtome mode' which allows you to interact with a simple 3D representation of the microtome (Figure 2.11). This has the constraints of the various axes built in, and shows the movement of the block in 3D space. A slider can be moved to cycle through all the different solutions and see their corresponding positions in the 3D viewer. This also gives the angles of the three axes and the distance to cut at the microtome.

Finally, there is a 'cutting mode' which allows you to simulate the progression of cutting through the block for the chosen solution (Figure 2.11). This allows the user to easily predict which features they will encounter at different cutting depths, and their orientations, by viewing both in 2D and 3D.

Crosshair is freely available on github (`https://github.com/K-Meech/crosshair`), and can be easily installed via a fiji update site.

2.2.4 Workflow

Once these solutions were calculated, I worked to establish a step by step workflow to go from an X-ray scan of a block to the final target. I summarise this in figure 2.12, and provide a detailed step by step in the appendix section B.

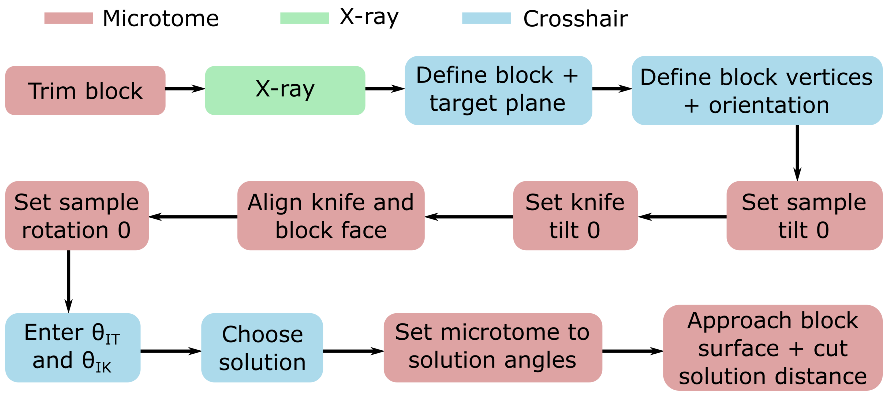

Microtome
X-ray
Crosshair
Trim block
X-ray
Define block + target plane
Define block vertices + orientation
Set sample tilt 0
Set knife tilt 0
Align knife and block face
Set sample rotation 0
Enter θ_{IT} and θ_{IK}
Choose solution
Set microtome to solution angles
Approach block surface + cut solution distance

Figure 2.12: Steps in the targeting workflow, colour coded by whether they are microtome, X-ray or Crosshair steps.

2.2.5 Accuracy measures

Once the workflow and software were established, I targeted a series of *Platynereis* samples to measure the accuracy. To do so, 5 *Platynereis* were prepared by Rachel Templin, and then trimmed and X-rayed by Rosa Pipitone. I then set up the target planes in Crosshair, aiming in each sample to cut through both of the anterior dorsal cirrus that protrude from either side of the *Platynereis* head. All samples were trimmed so that the target was less than one microtome feed length from the block surface (i.e. 200 microns), to ensure the greatest accuracy. This is necessary, as the start of cutting must be assessed manually by looking down the binoculars, and will always incur a slight error. Every time the feed length is exceeded, the microtome must be reset, and this manual approach redone. Therefore, the more times the microtome is reset, the greater the cumulative distance error will be.

I then targeted the samples with the semi-automated microtome and workflow described above. After targeting, these samples were X-rayed again by Inés Romero Brey and registered to the previous scan by myself. I then compared the target plane (from the before scan), to the block surface (from the after scan). See section 7.1.4 of the methods for details of how the accuracy measures were calculated.

Sample	Angle Error (degrees)	Distance error (microns)
a	0.31	3.01
b	1.27	2.37
c	0.44	1.07
d	3.08	9.22
e	0.18	6.83

This gave a mean angular error of 1.06 degrees, and a mean distance error of 4.50 microns (see table above and figure 2.13). Note that there is some variation in the individual accuracies, with one sample (d) having much lower accuracy than the rest. Also, our before and after scans show some slight variations which will introduce some error in this registration process - this is a work in progress, and will be improved in the coming months.

Nevertheless, comparing the before and after X-ray scans, showed that all samples hit the target of slicing through the two anterior dorsal cirrus, and gave very similar cross-sections to the predicted target (figure 2.14). This demonstrates that the method can be used to target structures in a quick and precise way.

2.3 Discussion

In summary, I have developed a method that allows quick, and precise targeting of a particular target plane in a sample. This was demonstrated with 5 separate *Platynereis* samples that all reached their target of slicing through both anterior dorsal cirrus. A mean angular error of 1.06 degrees, and a mean distance error of 4.50 microns was achieved. All targeting was done in one step i.e. we didn't stop at intervals to re-scan or take thick sections, allowing the method to be much faster than the current state of the art. As all steps are calculated via a user-friendly software, it also makes positioning a sample in 3D much easier, allowing easy targeting at a chosen orientation.

There are a number of places this method could be improved in future though. First, more samples should be targeted to better assess the accuracy, and its variability. Also,

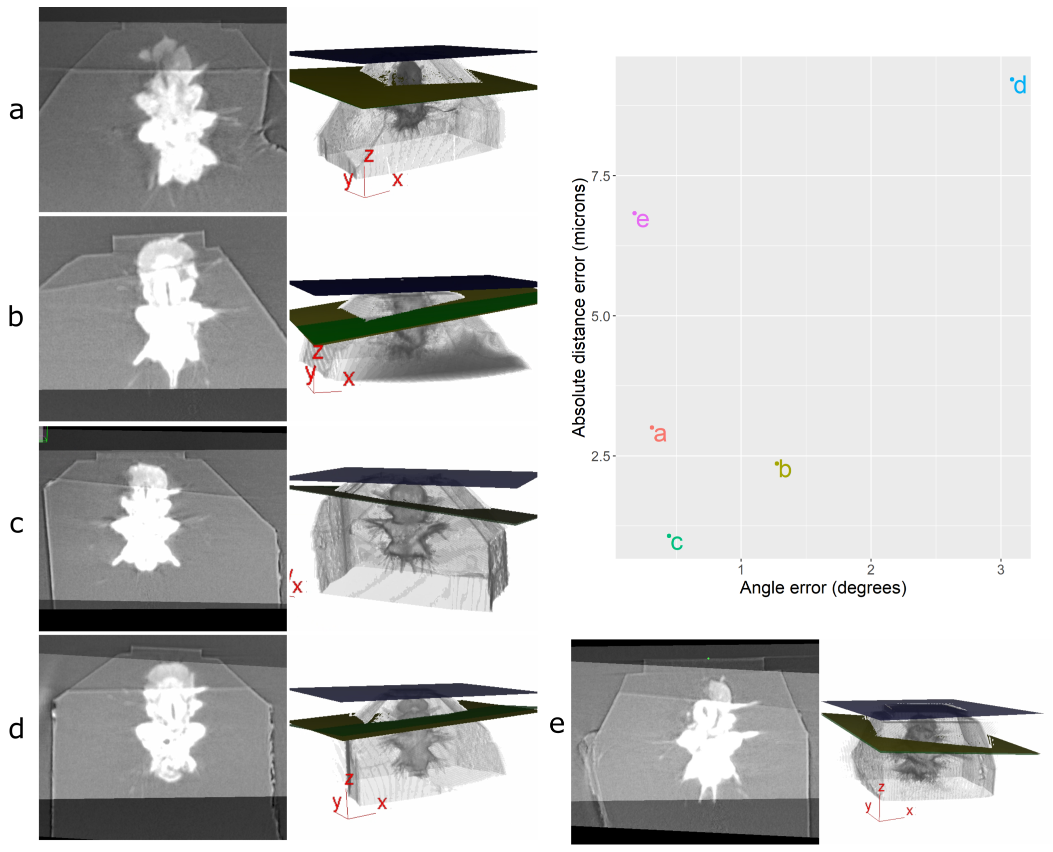


Figure 2.13: Accuracy tests results. In the top right, is a graph of the 5 samples (a-e) and their angle error vs distance error. Around the outside are images for samples a-e, with the left showing the registered before and after X-ray scans, and the right a snapshot of the 3D viewer in Crosshair. In the 3D, the block surface plane is shown in blue, the target in green, and the plane reached in yellow. In most cases, the green and yellow planes are too close together to distinguish well. The axes shown in the 3D view are the 3D viewer coordinate system.

our before and after X-ray scans should have their reconstructions improved, to make them more consistent, and provide a more precise registration. This will minimise any error introduced via this registration step.

In addition, any manual steps in the workflow should be minimised. For example, the setting of the zeros, the alignment of the knife and the block face, and assessment of when the knife reaches the block face are all done manually by the operator. This means that a more experienced ultramicrotome user will achieve better accuracy with this workflow than a less experienced one. Also, there will always be some variation in accuracy due to human error.

For the setting of the zeros, it would be useful to find a more accurate way to do this, that requires less human intervention. For example, the arc zero could be set by precise measurement of the arc piece, rather than be eye. The knife is more difficult, as the knife is removed and replaced each time, and different knives are used for different

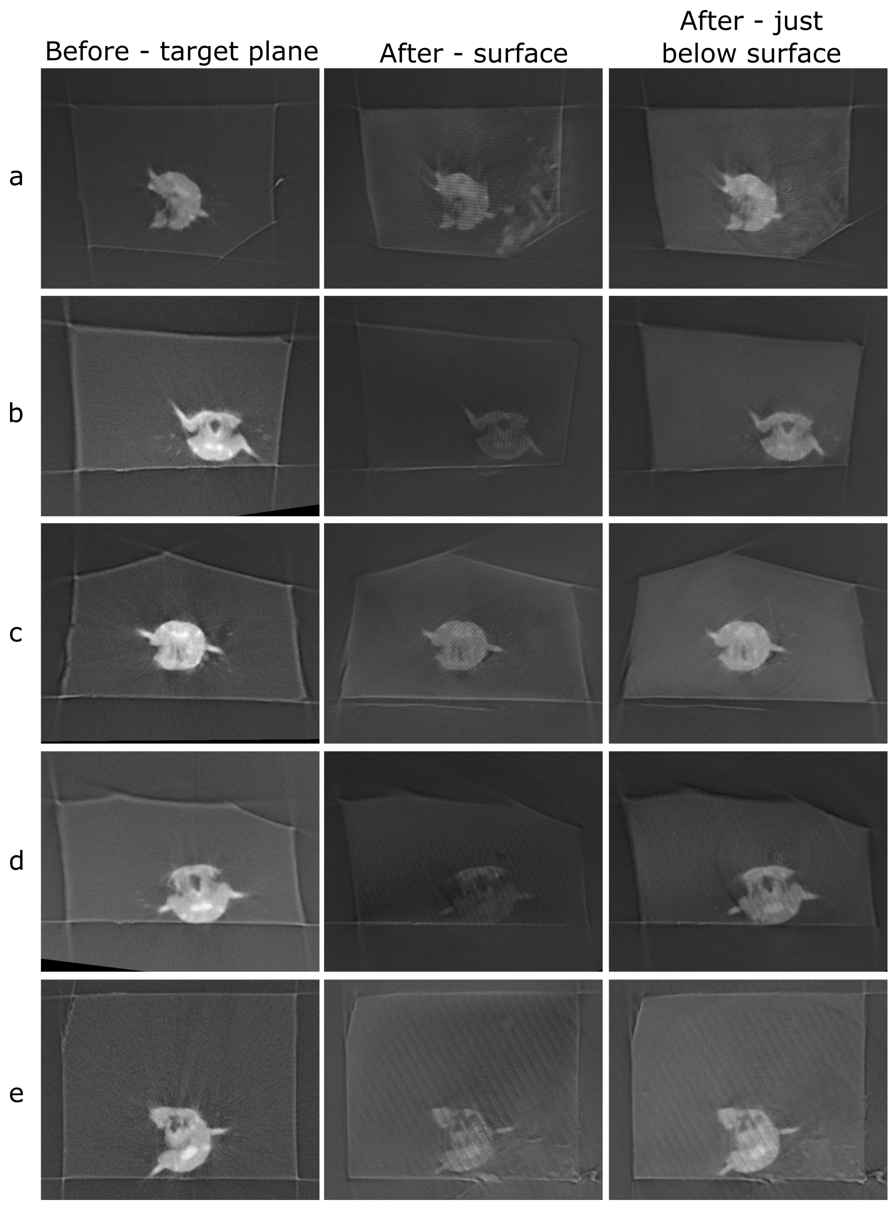


Figure 2.14: Comparison of target plane (from the before scan), to the block surface (after scan). Each sample has a row with three images. Left column - the target plane from the before targeting scan. Middle column - the block surface from the after targeting scan. Right column - cross-section parallel to block surface, but slightly inside the block. The middle column images can be quite dark / noisy as they slice right along the surface of the block, so I also provide the right column so the cross-section can be seen more easily.

purposes. Even so, it would be desirable to find a method to determine the zero that does not require manual alignment of the knife and block.

Next, for the manual alignment of the knife and block face, it would be useful to try and remove this step entirely. This step is only necessary because the relation between the block surface and the sample holder is unknown - the sample holder is a simple clamp which means the sample is in a different position each time it is inserted. This could be resolved by X-raying the sample directly inside the sample holder, so the orientation could be calculated from the scan. This is challenging in X-ray systems that use geometric magnification though, due to the large size of the sample holder. In these systems, the closer the sample is to the X-ray source, the greater the magnification is. Scanning a large sample holder would therefore greatly reduce the possible resolution, and decrease the accuracy of any targeting. This would be possible in other X-ray systems though that do not have this constraint. Another solution would be to design, for example, a thin pin that the sample can be scanned on top of. If this pin was made so that it would fit tightly inside the sample holder at a fixed orientation, then this would also allow the orientation to be calculated.

Finally, for assessing when the knife reaches the block face, it would be useful if there was a more automated way to do this. Assessing when the cutting begins is difficult, as the sample is often far from parallel to the knife (to hit the target), meaning that cutting starts from one corner. It is easy to cut a number of microns before the debris is easy to see on the knife, resulting in errors in the final distance cut. This is perhaps the most difficult manual step to automate, and would require some solution that allows the absolute distance between the knife and block surface to be known.

Moving away from the manual steps, it would be interesting to expand the technique to different kinds of data. For example, 3D light microscopy data that shows fluorescently labelled regions in the block should also be compatible. As long as the block surface can also be seen, then the same steps could be used for this. Also, it would be interesting to expand the method to more than one region of interest. For example, if your sample contains a series of structures you want to acquire, it would be useful to calculate a solution that allows you to hit each one in turn, without having to re-trim and X-ray the block again.

Another improvement would be to expand to targeting not just a plane, but a specific area within that plane. This is common for trimming samples for FIB-SEM, for example, where you can't image the entire block face. This would require integration of precise East-West movement of the microtome, and additional steps in the solution.

In its current state though, the method already provides a useful method for quick and precise targeting. As the code is open-source, and the motor system can be adapted to any standard Leica ultramicrotome, it can also be re-used in other labs. It has already been used by members of the Schwab lab and EMBL Electron Microscopy Core Facility for targeting particular regions/orientations in a sample. Also, there has been interest from the general EM community and industry when it has been shown at a number of workshops and courses.

Chapter 3

Acquisition: faster targeting with SBEM

3.1 Background

3.1.1 Layers of targeted acquisition

As discussed in the previous chapter, we need to find ways to make EM acquisition faster, easier and more precise. The work in the previous chapter provides a user-friendly solution to precisely hit a certain plane, at a certain orientation, in a resin-embedded sample. But what steps come after this, and how can they be made more efficient?

Depending on how large an area you wish to image, and at what resolution, there are a number of options after you have hit your region of interest. The first is to take thin sections of the sample at the microtome for TEM, or array tomography. This can be done directly at the microtome after the Crosshair workflow. The second is to transfer the block to a volume SEM - either a FIB-SEM (Focused Ion Beam Scanning Electon Microscope) or SBEM (Serial Block Face Scanning Electron Microscope). Both these microscopes work by removing thin layers from the surface of the block, then imaging its surface with SEM in an iterative loop. FIB-SEMs use an ion beam to remove layers, while SBEM uses a diamond knife.

FIB-SEMs offer very high resolution 3D imaging, up to around an isotropic voxel size of 5nm[40][12]. In return though, they have a smaller field of view than SBEM. Milling with an ion beam over a large surface area can lead to artefacts caused by redeposition of material and curtaining (non-planar milling). This means that, in practice, milling with the highest precision is possible over a width of about 20 microns[40][12]. Removing material with an ion beam is also much slower than the SBEM, meaning acquisition times are longer.

The SBEM, in contrast, offers a much larger field of view, but with a lower resolution (maximum around 10 x 10 x 25 nm^3)[40][12]. The resolution is particularly low in the z (cutting) direction as the diamond knife cannot cut as thinly as the FIB's ion beam. It is possible to image very large sample surfaces though, as small image tiles can be acquired and stitched together (around 1mm max). Cutting and imaging are also a lot faster than FIB-SEM meaning acquisition times can be much lower. For example, imaging a 50 x 50 x 50 μm^3 cube with SBEM (at 10 x 10 x 25 nm^3 resolution) would take about 22 hours, while with FIB (at 5 x 5 x 5 nm^3 resolution) this would take 4 months[40].

Therefore, for this chapter, I focus on improving targeted acquisition with the SBEM. The SBEM's large field of view and fast cutting make it a good technique for acquiring large samples, or many smaller samples at high throughput. This makes it a good fit for producing large amounts of EM for the kinds of multimodal atlases I discussed in the aims. It is worth noting that it is possible to produce very large FIB-SEM datasets by splitting resin blocks into thinner pieces with the 'hot-knife' technique, and milling these on multiple FIB-SEMs[22]. This is a difficult procedure though, and is not routine in many EM labs.

3.1.2 The structure of the SBEM

SBEM is a combination of an ultramicrotome and an SEM in one unit[41]. It features a miniaturised ultramicrotome, containing a diamond knife, that cuts thin sections from a sample inside the chamber of the SEM. Cutting with the ultramicrotome, and imaging with the SEM are alternated, so that the surface of the block is gradually removed and imaged in 3D. By necessity, this process is therefore destructive - the sample cannot be re-used after imaging with SBEM.

Samples are fixed to small pins (called stubs) and placed into a sample holder that goes inside the microscope. The sample can then be moved up by a specified distance, and cut with the diamond knife to remove sections. Images are acquired in small patches (called tiles) to cover the desired area of the block surface. These are aligned and stitched together after the acquisition to provide the final images.

While SBEM is faster than most other volume EM methods, it is still a time consuming process. For example, the *Platynereis* atlas I will discuss in chapter 4 was collected by SBEM and consisted of over 11,000 planar images and more than 200,000 individual tiles covering the whole worm[15]. This image stack took 7 weeks of continuous acquisition to acquire. Therefore, we still need to find ways to speed up these acquisitions as much as possible. Again, as with the last chapter, a possible solution is to be more targeted in our acquisitions - i.e. to focus on specific regions of interest. This way, we can cut thicker and faster in parts of the sample we are not interested in, and only image slowly at high resolution in certain areas.

3.1.3 A tale of two microtomes

What are the differences between a standard ultramicrotome, and the miniaturised one in a SBEM? First, due to the size constraints of miniaturising the ultramicrotome, the one inside the SBEM is much simpler in construction. For example, the knife has a fixed position and cannot be rotated to cut at a specific angle. Also, the sample cannot be rotated, and can only be moved up and down in the z direction. This means that it is impossible to target a region of interest at a certain orientation, or to trim from multiple angles.

Another difference is the maximum size of sample. In the SBEM, the sample must be trimmed small enough to fit on the SBEM stub (around 1 mm). In contrast, a standard ultramicrotome has a large clamp allowing very large samples to be placed inside. Also, the maximum cutting thickness is reduced in the SBEM, with 200nm being around the maximum. A normal ultramicrotome can cut much thicker, cutting easily at 2 micron thickness. Finally, the cutting speed is reduced in the SBEM, so trimming away large amounts of resin takes much longer (maximum of around 1 mm per second for the SBEM vs 100 mm per second for a standard ultramicrotome).

All these constraints mean that, in general, samples are trimmed first at a standard ultramicrotome before being placed in the SBEM and trimmed further. Therefore, it is useful to optimise both of these steps to reduce the overall acquisition time. The optimisations discussed in chapter 2 address the first step, while those in this chapter will address the second.

3.1.4 Targeting with external 3D maps

As with the usual ultramicrotome, one strategy to allow targeted acquisition is to use information from external 3D maps (see section 2.1.4 for background on this). For example, let's highlight a few examples that use X-ray in direct connection with volume EM. Bushong et al. [20] use micro-CT of mouse brain slices to guide targeted acquisition of regions in SBEM. They use both finder grids (metal grids with a regular layout of numbered zones) and fiducial markers to find back regions of interest.

Another example is Musser et al.[25]. Here, they use both laboratory micro-CT and X-ray imaging at a synchrotron to target specific neuroid cells in a sponge sample. Both kinds of X-ray were registered together and used to guide trimming at an ultramicrotome, and then a FIB-SEM. Comparison of the high-resolution synchrotron X-ray and the FIB-SEM data show many of the same cellular features allowing very accurate tracking of the region of interest.

Of course, imaging with light microscopy can also provide useful 3D maps for volume EM. For example, Brama et al.[27] that I discussed briefly in the last chapter, also produced a miniaturised light microscope that sits directly in the chamber of the SBEM. This 'miniLM' allowed imaging of fluorescent markers in the resin embedded sample directly after each cut of the ultramicrotome. This allows targeting of specific, fluorescently marked, regions of interest.

To summarise, both X-ray and light microscopy can provide useful 3D data for targeting regions of interest in volume EM (specifically SBEM or FIB-SEM). There are limitations to the ease and accuracy with which these 3D maps can be used though. For example, all the X-ray methods mentioned above rely on manual matching of X-ray and EM images, or separate software to do so. The registration, depth measurement and visualisation are not integrated with the rest of the acquistion software. This means that seeing your EM and X-ray together can take a number of time consuming steps, and are not easy to monitor directly as the microscope runs and collects images. Also, it can be difficult to convert measurements of your region of interest into the same coordinates as the microscope is using for acquisition.

In addition, the cutting thickness of a SBEM or FIB-SEM can be inconsistent depending on the sample quality, charging and a number of other factors[42][43][41]. This can mean that depth estimates made from an X-ray, can be difficult to accurately cut at the microscope. In practice, this means you must continuously monitor your acquisition and compare the X-ray and EM to ensure you reach your region of interest. Again, this is challenging if these steps are not integrated with the main acquisition software, or require many different manual steps.

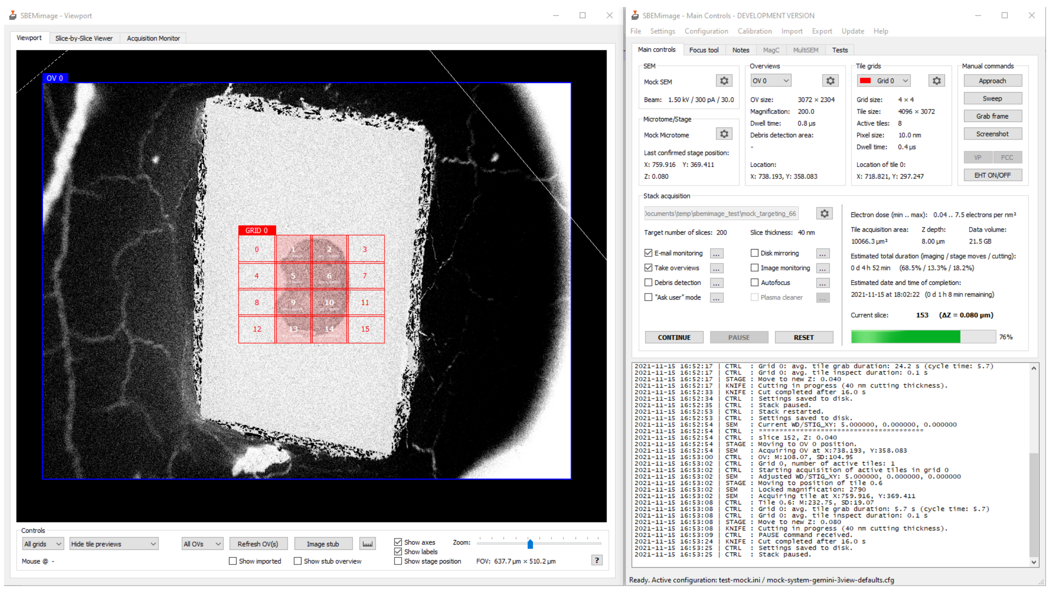
3.1.5 SBEMimage

For our prototyping, we use a Zeiss Gemini SEM with a Gatan 3View microtome. These components are controlled by a combination of two pieces of software: SmartSEM (for the SEM) and GMS3 (for the microtome). As both of these softwares are proprietary, and therefore difficult to extend/adapt to our aims, we run another open-source software on top of these: SBEMimage[44]. SBEMimage is an open-source, python based application developed by Benjamin Titze and others. It is designed to control a variety of SEMs and microtomes, and can be easily extended to add new functionality.

SBEMimage allows a lot of flexibility in how to image with the SBEM. For example, multiple regions can easily be imaged at multiple different resolutions and sizes in the same run. This is achieved through two main imaging methods: overviews and grids. Overviews are represented by blue rectangles in SBEMimage (Figure 3.1) and allow you to select large regions of the sample surface for low resolution imaging. They are used to provide quick snapshots of the sample to guide where to acquire at high resolution. All high resolution imaging is done with grids, which are represented by numbered rectangles of various colours in SBEMimage (Figure 3.1). Each grid is made up of a series of individual numbered tiles, that each produce a high resolution image. This is standard practice for imaging with SBEM - due to the large size of the sample surface, it must be acquired in a series of small patches that are later stitched together to give the full sized image. SBEMimage allows these individual tiles to be turned on/off interactively throughout the run, allowing efficient acquisition of only certain regions of interest.

3.1.6 Aims

My aim was to develop software that integrates closely with SBEMimage to allow targeting of regions of interest with 3D maps. This should allow user-friendly visualisation, image registration and measurement of regions of interest to allow precise targeting. It



Mouse @ -
[✓] Show axes
[✓] Show labels
[ ] Show stage position
[ ] Show imported
[ ] Show stub overview
FOV: 637.7 µm × 510.2 µm

Mock SEM
Beam: 1.50 kV / 300 pA / 30.0
Mock Microtome
Last confirmed stage position:
X: 759.916 Y: 369.411
Z: 0.080

OV 0
OV size: 3072 × 2304
Magnification: 200.0
Dwell time: 0.8 µs
Debris detection area:
-
Location:
X: 738.193, Y: 358.083

Grid 0
Grid size: 4 × 4
Tile size: 4096 × 3072
Active tiles: 8
Pixel size: 10.0 nm
Dwell time: 0.4 µs
Location of tile 0:
X: 718.821, Y: 297.247

)ocuments\temp\sbemimage_test\mock_targeting_66
Target number of slices: 200
Slice thickness: 40 nm
[✓] E-mail monitoring
[✓] Take overviews
[ ] Debris detection
[ ] "Ask user" mode
[ ] Disk mirroring
[ ] Image monitoring
[ ] Autofocus
[ ] Plasma cleaner

Electron dose (min .. max): 0.04 .. 7.5 electrons per nm²
Tile acquisition area: 10066.3 µm²
Z depth: 8.00 µm
Data volume: 21.5 GB
Estimated total duration (imaging / stage moves / cutting):
0 d 4 h 52 min (68.5% / 13.3% / 18.2%)
Estimated date and time of completion:
2021-11-15 at 18:02:22 (0 d 1 h 8 min remaining)
Current slice: **153** **(ΔZ = 0.080 µm)**
76%

```
2021-11-15 16:52:17 | CTRL  : Grid 0: avg. tile grab duration: 24.2 s (cycle time: 5.7)
2021-11-15 16:52:17 | CTRL  : Grid 0: avg. tile inspect duration: 0.1 s
2021-11-15 16:52:17 | STAGE : Move to new Z: 0.040
2021-11-15 16:52:17 | KNIFE : Cutting in progress (40 nm cutting thickness).
2021-11-15 16:52:33 | KNIFE : Cut completed after 16.0 s
2021-11-15 16:52:34 | CTRL  : Settings saved to disk.
2021-11-15 16:52:35 | CTRL  : Stack paused.
2021-11-15 16:52:53 | CTRL  : Stack restarted.
2021-11-15 16:52:53 | CTRL  : Settings saved to disk.
2021-11-15 16:52:54 | SEM   : Current WD/STIG_XY: 5.000000, 0.000000, 0.000000
2021-11-15 16:52:54 | CTRL  : ****************************************
2021-11-15 16:52:54 | CTRL  : slice 152, Z: 0.040
2021-11-15 16:52:54 | STAGE : Moving to OV 0 position.
2021-11-15 16:52:54 | SEM   : Acquiring OV at X:738.193, Y:358.083
2021-11-15 16:53:00 | CTRL  : OV: M:108.07, SD:104.95
2021-11-15 16:53:02 | CTRL  : Grid 0, number of active tiles: 1
2021-11-15 16:53:02 | CTRL  : Starting acquisition of active tiles in grid 0
2021-11-15 16:53:02 | SEM   : Adjusted WD/STIG_XY: 5.000000, 0.000000, 0.000000
2021-11-15 16:53:02 | STAGE : Moving to position of tile 0.6
2021-11-15 16:53:02 | SEM   : Locked magnification: 2790
2021-11-15 16:53:02 | SEM   : Acquiring tile at X:759.916, Y:369.411
2021-11-15 16:53:08 | CTRL  : Tile 0.6: M:232.75, SD:19.07
2021-11-15 16:53:08 | CTRL  : Grid 0: avg. tile grab duration: 5.7 s (cycle time: 5.7)
2021-11-15 16:53:08 | CTRL  : Grid 0: avg. tile inspect duration: 0.1 s
2021-11-15 16:53:08 | STAGE : Move to new Z: 0.080
2021-11-15 16:53:08 | KNIFE : Cutting in progress (40 nm cutting thickness).
2021-11-15 16:53:09 | CTRL  : PAUSE command received.
2021-11-15 16:53:24 | KNIFE : Cut completed after 16.0 s
2021-11-15 16:53:25 | CTRL  : Settings saved to disk.
2021-11-15 16:53:25 | CTRL  : Stack paused.
```

Ready. Active configuration: test-mock.ini / mock-system-gemini-3view-defaults.cfg

Figure 3.1: SBEMimage user interface. The SBEMimage user interface is split into two main windows: left, the viewport where images can be viewed in 2D, and right, the main controls, where all the settings for the acquisition can be controlled. In the viewport, an overview is shown (blue rectangle) of a *Platynereis* embedded in a resin block. A grid is also displayed (red) with 16 tiles - the central tiles are enabled and the rest disabled to acquire only the part of the block containing the *Platynereis*.

should also allow easy monitoring of the acquisition as the microscope runs, allowing EM and X-ray to be continuously compared. I aimed to make this software as general as possible (to accommodate different 3D maps like X-ray, light microscopy, and other modalities), but all initial tests were with X-ray. As X-ray allows many of the same cellular features to be recognised as EM, it provides a great test case with relatively simple image registration. I focus on the highest resolution X-ray (synchrotron) as this provides the best 3D maps, that are closest in resolution to the EM itself.

This software can be used in combination with Crosshair (see chapter 2), or with normal manual trimming at an ultramicrotome. It was designed to be general for targeting any resin-embedded sample that has a good quality 3D map.

Note that all X-ray images shown in this chapter were collected in collaboration with Thomas Schneider's group at EMBL Hamburg, with all imaging and reconstruction by Maxim Polikarpov and Gleb Bourenkov, and all sample preparation by Rachel Templin. See section 7.2.5 of the methods for details.

3.2 Results

3.2.1 Targeting workflow

The targeting problem here is actually a somewhat simpler one than the ultramicrotome in the previous chapter. Here, the sample can only be moved in the z direction (the cutting direction) and cannot be rotated in any way. Therefore, I only need to determine the depth to the target, and no rotation angles.

The region of interest is also defined differently here. For the ultramicrotome, the target was a plane at a certain orientation that I wished to cut to precisely. Here though, the region of interest is instead a 3D volume which I wish to acquire with the SEM. Therefore, I must know the depth to the target, and also its full extent in 3D space i.e. where should a grid be placed to cover this whole region? Which tiles should be active, and at which z depths to acquire it fully in an efficient way? Due to these differences, it needs a different approach to the previous chapter.

When a sample is mounted on a SBEM stub, it is difficult to control its exact orientation. This mounting is done manually with forceps under a binocular microscope, and depending on how you fix the sample to the stub, and how the stub is placed into the holder, its orientation will be slightly different every time. This means it is hard to take measurements made with the X-ray and immediately translate this into coordinates for the SEM. I need some way to reliably determine the exact position and orientation of the sample in the SBEM.

In addition, even if I could know the exact orientation of the sample, assessing the z depth from the initial X-ray alone would still be difficult. Once the sample is placed in the holder, it must be manually raised towards the knife, and cutting started until the surface is reached. Assessing exactly when the sample surface is hit is challenging - the sample must be watched manually down a binocular microscope to determine when this happens. If cutting begins from a corner, only small pieces of resin are removed, making this very difficult to determine accurately. In addition, even if I know exactly when the surface is hit, the cutting depth is not always perfectly reliable introducing more uncertainty. In effect, this means I can't rely on a simple measurement of the depth of the region of interest from the surface in the X-ray. I need a method that can continuously monitor the progress and adjust for any error.

To account for these challenges, the most general solution was to use image registration at regular intervals to track regions of interest. To acquire EM for this registration, I use low-resolution overviews (100nm resolution) acquired with the thickest cutting possible with the SBEM (200nm). This allows the imaging to be done rapidly, while still providing sufficient resolution to register to the X-ray (our X-ray images had an isotropic voxel size of 325nm). The amount required for accurate registration will depend on the sample, but for my tests I use 100 slices at 200nm (20 microns total depth). This can be acquired in less than one hour.

Once this small volume is acquired, it can be used to register the X-ray volume. This allows the exact coordinates of any regions of interest in the X-ray image to be known, and acquired. If the cutting thickness is inconsistent, this process can be repeated, to refine the registration (See Figure 3.2B).

3.2.2 Software stack: SBEMimage and Fiji

To implement this workflow in an accessible way, I combined both SBEMimage and Fiji (Figure 3.2A).

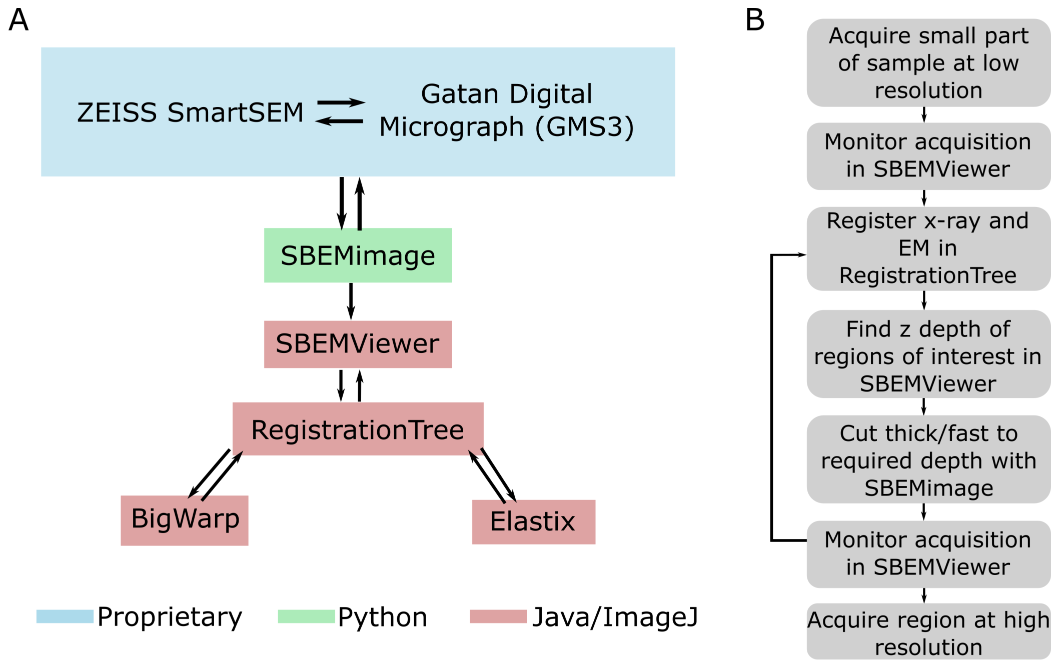

A
ZEISS SmartSEM
Gatan Digital Micrograph (GMS3)
SBEMimage
SBEMViewer
RegistrationTree
BigWarp
Elastix
Proprietary
Python
Java/ImageJ
B
Acquire small part of sample at low resolution
Monitor acquisition in SBEMViewer
Register x-ray and EM in RegistrationTree
Find z depth of regions of interest in SBEMViewer
Cut thick/fast to required depth with SBEMimage
Monitor acquisition in SBEMViewer
Acquire region at high resolution

Figure 3.2: Targeting workflow. ***A*** - Software for workflow. Proprietary software is shown in blue, open-source python in green and open-source Java/ImageJ in red. ***B*** - Summary of overall workflow for targeting at the SBEM.

I first made a number of modifications to SBEMimage. I added a new panel for all targeting related features (that can be expanded in future) with a simple user interface. This allows an overview to be selected, and all images acquired from it to be converted on the fly to the n5 file format along with metadata about their voxel size and position in 3D space. The n5 file format[45] is a chunked, pyramidal file format which allows browsing of very large images (terabytes in size) on a normal laptop. While the overviews are low resolution, for long acquisitions their number and size can be substantial, so the conversion is worthwhile.

Second, I added the option to give a target depth (in microns). This automatically determines the number of slices required at the current section thickness, and stops the run when this is reached.

3.2.3 General registration of very large images

Next, I needed a general way to register X-ray images to the low-resolution SBEM overviews. This needed to handle potentially very large images, and be adaptable to other 3D maps e.g. light microscopy or different resolutions of X-ray imaging. For this, I decided to write a wrapper around BigWarp[46] and Elastix[47][48] that would allow easy passing of images from one software to the other, and smooth the integration of large images with elastix. Both BigWarp and Elastix are registration softwares that allow matching of images based on corresponding points (BigWarp) or intensity based features (Elastix).

This is publicly available as a Fiji plugin called 'RegistrationTree' on github (`https://github.com/K-Meech/RegistrationTree`), and can be easily installed via a fiji update site. It is stand-alone, and can be used separately from the rest of the workflow, making it easily re-useable for other projects that need registration of large images. RegistrationTree, like Crosshair in the previous chapter, uses BigDataViewer[38] to allow browsing of very large images in arbitrary orientations. It also uses BigDataViewer style file formats (hdf5 and n5) to allow these large images to be viewed efficiently on a normal laptop.

RegistrationTree allows users to build arbitrary trees of affine transformations to register their images together (Figure 3.3). Each node on this tree can be viewed on the fly in BigDataViewer, allowing different registrations to be easily compared. Each node in this tree can either be a BigWarp registration, or an elastix registration, allowing easy chaining together of these two softwares. All steps are done without re-writing the original image data, apart from initiating elastix. Elastix requires specific file formats and cannot use hdf5/n5 directly. To make this efficient, RegistrationTree provides interactive ways to specify a cropped region (Figure 3.3B), and a downsampling level to write for elastix. All the required transformations to account for the cropping/downsampling are handled automatically.

Overall, RegistrationTree provides a user-friendly method for affine registration of large images.

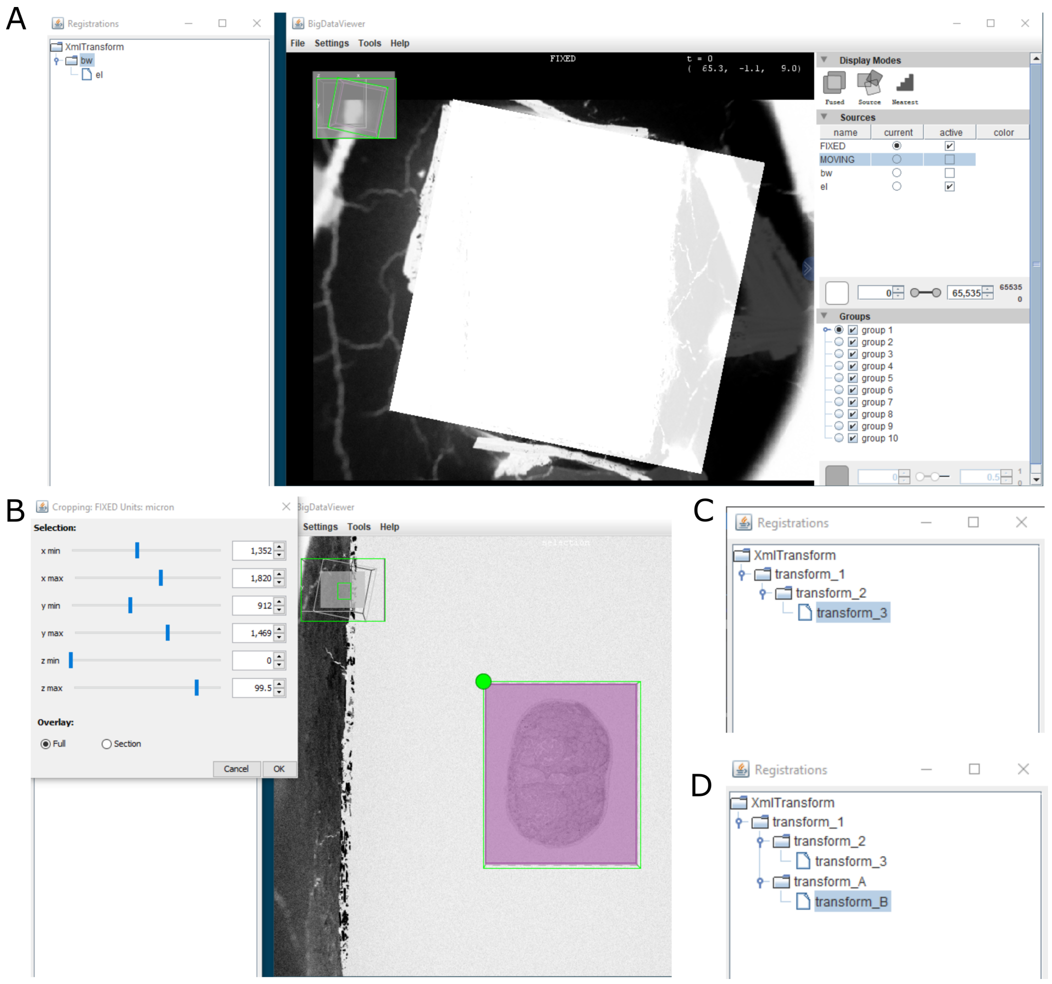
3.2.4 SBEMViewer plugin

Once the modifications to SBEMimage and RegistrationTree were complete, I developed a final fiji plugin to act as a bridge between the two (Figure 3.2A). This plugin, called 'SBEMViewer' allows images from SBEMimage acquisitions to be viewed in real time as the acquisition progresses (Figure 3.4). It also uses BigDataViewer to allow browsing in any orientation. In addition, it allows direct comparison to any X-ray images, all displayed in the same coordinate system used in SBEMimage.

This plugin allows X-ray images to be loaded, viewed and directly sent to RegistrationTree for registration. This allows easy monitoring of EM vs X-ray as the acquisition progresses. Registration can be repeated at any stage if the alignment is seen to deteriorate.

3.2.5 Targeting *Platynereis*

To demonstrate this prototype workflow, I targeted points in a resin-embedded *Platynereis* sample. I imaged 100 slices at 200nm thickness with low resolution overviews



XmlTransform
- bw
  - el

name	current	active	color
FIXED	●	✓	
MOVING	○		
bw	○		
el	○	✓	

Groups: group 1, group 2, group 3, group 4, group 5, group 6, group 7, group 8, group 9, group 10

Selection:

x min	1,352
x max	1,820
y min	912
y max	1,469
z min	0
z max	99.5

Overlay: [✓ Full] Section

XmlTransform
- transform_1
  - transform_2
    - transform_3

XmlTransform
- transform_1
  - transform_2
    - transform_3
  - transform_A
    - transform_B

Figure 3.3: Registration Tree user interface. ***A*** - The RegistrationTree interface is split into two main panels. On the left is a panel where the tree of registrations is displayed. On the right is a BigDataViewer window allowing images to be browsed in 2D slices. ***B*** - example of interactive cropping of an image for elastix. ***C*** - an example registration tree with 3 transforms - transform 1 then 2 then 3. ***D*** - an example registration tree with branching transforms. Here, there are two possible routes: transform 1 then 2 then 3, or transform 1 then A then B. Each node on this tree can be viewed and compared separately in BigDataViewer.

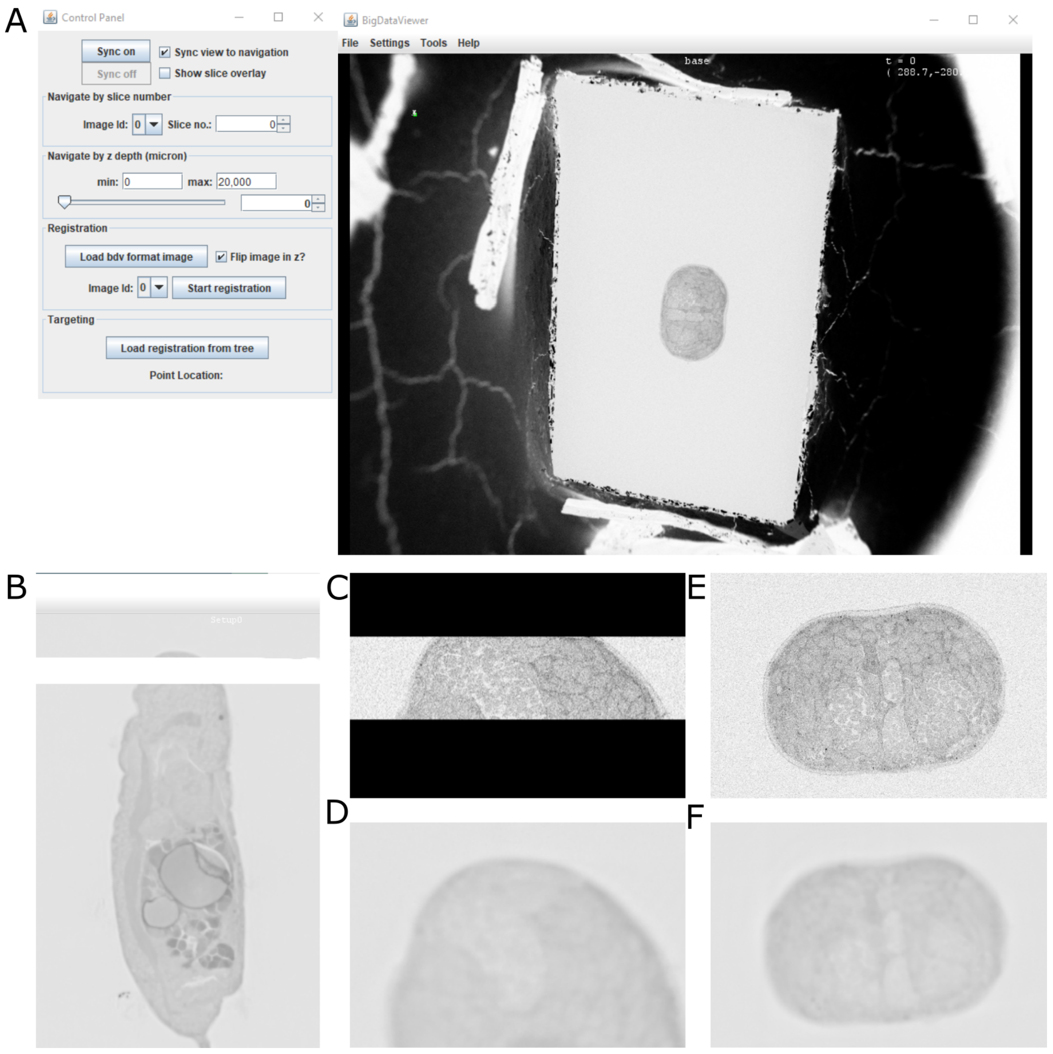


Figure 3.4: SBEMViewer ***A*** - SBEMViewer user interface. The left panel contains all the controls for navigation, and sending images to RegistrationTree for registration. The right panel is a BigDataViewer window for browsing images. ***B*** - Registered EM and X-ray viewed together in BigDataViewer. A cross-section through a *Platynereis* is visible in the X-ray, with the small EM volume as a white stripe in the *Platynereis* head. ***C*** - ***F*** - close ups of registered EM *(C and E)* and X-ray *(D and F)* viewed individually in BigDataViewer. C and D are in the same orientation as B, while E and F are viewed directly in the xy plane.

(100nm in xy) that contained the entire *Platynereis* cross-section. These images were converted on the fly to n5 and monitored in real time in the SBEMViewer plugin.

After collection, this small EM stack was sent to RegistrationTree along with a synchrotron X-ray scan of the same sample (325nm pixel size). They were registered together with a combination of BigWarp and Elastix, and the resulting tree loaded back into SBEMViewer.

I chose an arbitrary point to target, that was between the jaws of the *Platynereis* sample and then set SBEMimage to cut to that depth.

Looking at the results (Figure 3.5), the initial registration worked well, with good correspondence of features between the EM and X-ray in these initial 100 slices. It is clear to see though, that as a greater depth was covered, the quality of the registration degraded (Figures 3.5 and 3.6). This is not unexpected - there are always slight xy shifts in the imaging over time due to a number of factors including imprecision in motor movements of the stage. The most significant contributor though is the cutting depth which can be quite inconsistent depending on the quality of the sample preparation, and factors like the electron dose. In this example, the EM dataset seems compressed in z (Figure 3.6) relative to the X-ray implying that it is, on average, cutting in larger increments than it is should.

In this example, I only did the registration once at the start, so I could assess if drifts in xy and the cutting depth would cause issues. In a real scenario, of course, the registration could be repeated at intervals to correct for this process. Misalignments are easy to see by comparing the X-ray and EM in BigDataViewer, and registration can be triggered again at any stage.

3.3 Discussion

The workflow presented in this chapter provides user-friendly targeting, and visualisation that is integrated closely with the acquisition. The example *Platynereis* demonstrates that monitoring the acquisition, and registering high resolution X-ray can be achieved. The software is a mix of SBEMimage (python) and Fiji plugins, allowing the whole workflow to be used with no programming experience.

This being said, there are clearly a number of areas that can be expanded to improve this prototype. First of all, in this example, I only registered the X-ray once so that any xy shifts or cutting errors would be visible. It would be useful to run another *Platynereis* where the registration is repeated at intervals, and see what kind of accuracy can be achieved. It would also be useful to improve support for repeated registrations. For example, I could integrate an option in SBEMViewer where only a certain subset of slices are sent to RegistrationTree. This would be useful in scenarios where the cutting depth is unreliable, and registration is repeated only on the most recently acquired slices.

Also, it would be useful to add some automation to triggering the registration at certain

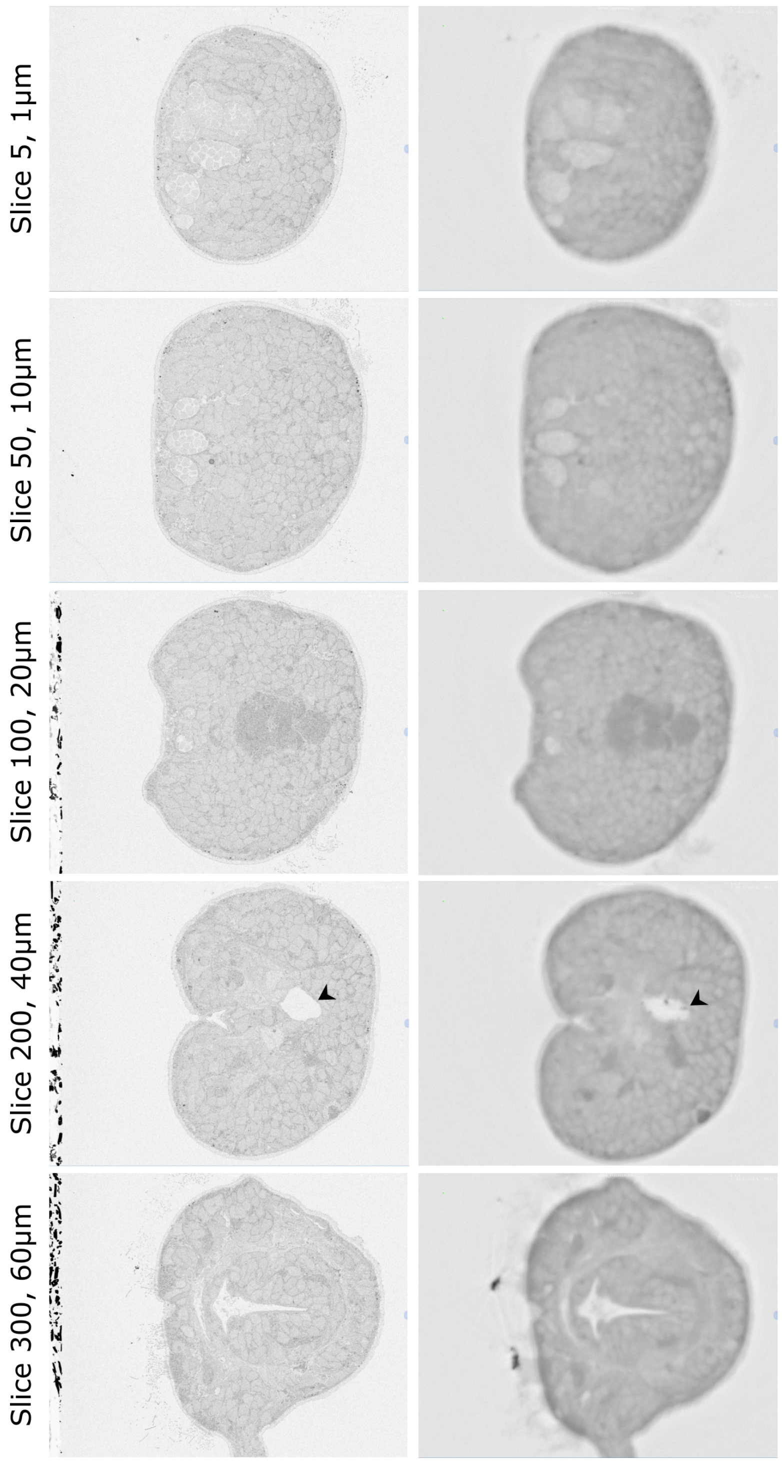


Figure 3.5: XY planes for EM (left column) and X-ray (right column) at various depths in the run. For slice 200, I add some small black arrows indicating an example of misalignment. Here, the white cavity inside the *Platynereis* can be seen to have a different shape in the EM and X-ray. These misalignments are easier to see in the ZX and ZY planes of figure 3.6.

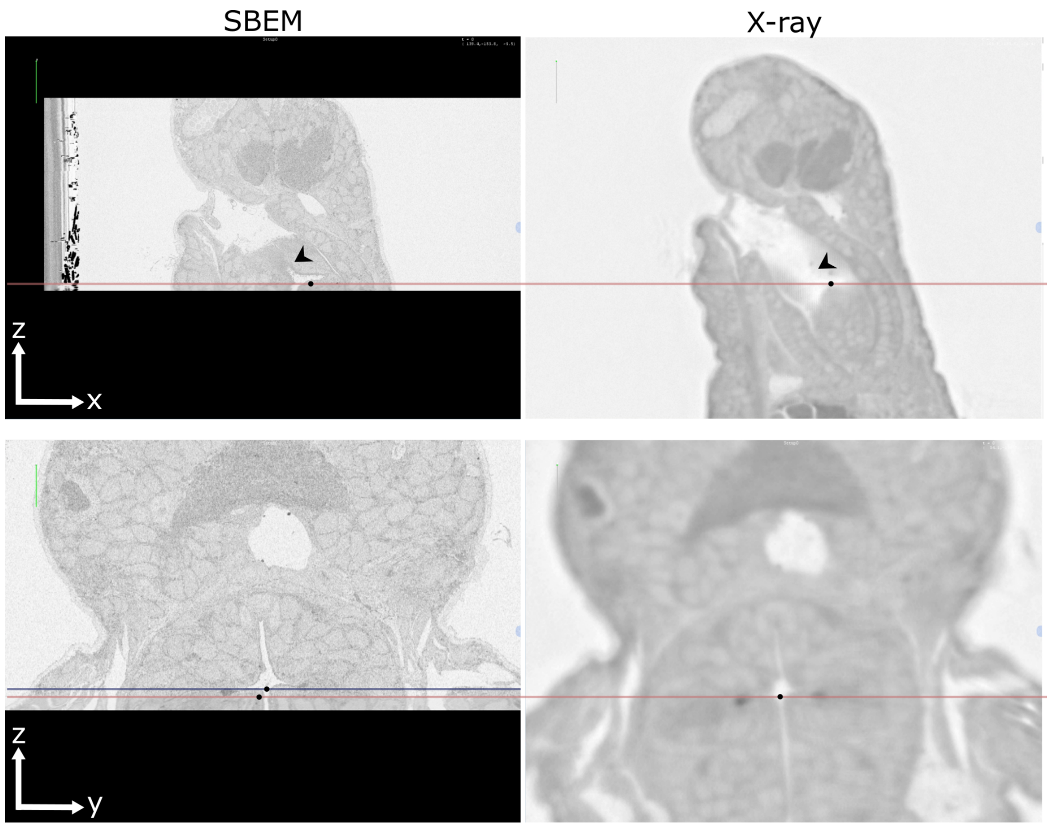


Figure 3.6: EM and X-ray images from the SBEM run for the ZX plane (top row) and ZY plane (bottom row). The target depth (as defined from the X-ray, to lie in the cavity between the two jaws) is indicated as a red line, with the black point indicating the exact position. I also add some markers to highlight areas of misalignment between the EM and X-ray. In the top panels, the two arrows indicate an area where the EM cuts through a tissue boundary that is not seen in the X-ray - this indicates some misalignment in XY. In the bottom left panel, I add an approximate depth and position of the true target in the EM (blue line and point). This indicates some misalignment in both y and z.

intervals. For example, there could be automatic comparison of X-ray and EM images, to determine when a certain threshold of dissimilarity is passed. At the moment, this must be done manually by monitoring the acquisition and figuring out when datasets are visually misaligned.

The most important next step is to allow SBEMViewer to directly communicate back to SBEMimage. In this way, user selected regions of interest could be converted to coordinates to place grids in SBEMimage and acquire them automatically. At the moment, this must be set manually based on the coordinates visible in SBEMViewer.

In addition, it would be beneficial to support interactive placement of regions of interest in SBEMViewer. For example, placement of simple rectangular regions of interest, or painting more complex arbitrary shapes.

Another improvement would be to support deformable registrations in RegistrationTree.

At the moment, only affine is supported as these are easy to chain together in sequence, and perfectly sufficient for X-ray. For other modalities though, like light microscopy, especially if it is done before the sample preparation for EM, more flexible transformations can be required. Both BigWarp and Elastix support deformable registrations, and these can be viewed in BigDataViewer, so it should be possible to implement. The difficultly is how to chain them together in an efficient way and ensure that they are compatible with each other.

Another more long-term improvement would be to possibly swap from Fiji to Napari[49] for these plugins. Napari is an image viewer for python that has undergone a massive increase in popularity over recent years. As SBEMimage is written entirely in python (and so is Napari), this would allow much easier communication back and forth throughout this process. The downside currently is that Napari is still in alpha stage, and so rapidly changing. Also, there are a number of features that are not as well supported yet as in Fiji e.g. slicing in arbitrary orientations, smooth browsing of massive images and some registration features. Napari is rapidly improving though, so in a few years it would make a lot of sense to switch to allow smoother interaction between the plugins and SBEMimage.

Nevertheless, even without these improvements, the prototype already allows much easier targeting with X-ray and SBEM. In recent years, synchrotron X-ray resolution has improved greatly for biological samples. Recent studies can achieve resolutions in the range of hundreds or even tens of nanometres, with samples prepared in the standard way for EM. For example, Bosch et al.[21] image mouse brain samples at around 330nm resolution at a synchrotron, and acquire SBEM data of the same samples. They observe many of the same features between the two datasets, and demonstrate tracing neurons in both. Kuan et al.[19] image a variety of samples including mouse cortex, adult *Drosophila* brain, ventral nerve cord and leg at resolutions up to 100nm at a synchrotron. They also collect EM images of many of the same samples, demonstrating that many of the same features are visible in both datasets. These studies demonstrate how X-ray imaging can provide fast imaging of large volumes, followed by targeted EM acquisition where greater resolution is required e.g. for identifying synapses. As the resolution and accessibility of these methods continues to improve, it will be more important than ever to have robust methods for targeted acquisition. Also, importantly, to have methods that are integrated with the rest of the acquisition pipeline in a user-friendly way.

Chapter 4

Analysis: quantitative morphology from EM

4.1 Background

4.1.1 Quantitative analysis of morphology from EM

Once our regions of interest have been targeted, and acquired with EM, we move to the analysis stage. The exact steps here depend on the particular sample in question, and what other datasets are available in the atlas for comparison. A common goal though is to turn EM images into quantitative descriptions of cell morphology that can be compared to other data in the atlas.

Directly from the microscope, volume EM provides us with a series of 2D grayscale images that represent the structure of the sample. In each of these images, the intensity of each pixel indicates something about the electron density of that particular point, allowing different features to be recognised based on how bright they appear. How do we turn this into a quantitative description of morphology though? Classically, this is done entirely manually - for example, if the researcher is interested in how nucleus diameter varies across a sample, they could manually go through these images and measure each nucleus. Of course, EM data can be extremely large so this quickly becomes unfeasible.

Another manual option is 'stereology'[50][51]. This is a series of standard procedures that allow random sampling of images, and regions within images, to estimate quantitative measures like cell volume or surface area. For example, the volume fraction of a certain organelle could be estimated by randomly placing a series of points over an image, and counting the proportion that fall within a certain organelle. While this increases speed greatly by using only a subset of images, it still requires a lot of manual work. Also, with many samples, or many different morphological features to measure, it will quickly become unfeasible too.

Another option is instead to segment your image first, and then measure morphological features directly from these segmentations. Segmentation is the process of finding which pixels of an image belong to which structure. For example, which areas belong to one cell, and which to another? This task can also be accomplished in many ways. The easiest, but most time consuming solution, is again to do this entirely manually. Image analysis software like ImageJ/Fiji[37], Amira and Napari[49] allow researchers to draw directly on images to specify which pixels belong to which stucture. This process can be very time consuming, and very tedious though - for example, connectomics projects often have many researchers working full time to manually trace neurons in EM data. To give an example, in one serial section TEM dataset of an entire fly brain[52], manually tracing 114 olfactory projection neurons took 1272 hours. This gives a mean reconstruction time of 11.2 hours per cell! Clearly, this approach becomes unfeasible for very large data, or where a large number of researchers can't be hired for this task alone.

In recent years though, machine learning based approaches have revolutionised the possibilities for efficiently segmenting EM data. In these approaches, a small subset of images can be manually segmented and used as training data to segment the rest automatically. This makes it feasible to segment large datasets and analyse their morphologies automatically. A number of user-friendly options exist - for example, ilastik[53] and the Fiji Weka plugin[54]. Deep learning based approaches offer the best performance, but need more training data, and also more expertise and compute power to setup. Pre-trained networks are available though and can be used through user-friendly software like ilastik and deepImageJ[55].

This being said, these approaches still require the creation of manual segmentations for training data. This means some of the same limitations on time, and number of people still exist. Some large scale projects rely on out-sourcing this task to companies like ariadne.ai[56], or crowdsourcing with the general public[57][58].

Regardless of how the images are segmented, there are then many different options for analysing morphologies. In general, researchers tend to choose a series of manually defined features to calculate like volume, surface area and sphericity. For example, Uwizeye et al.[59] analyse the morphological differences between organelles in different species of phytoplankton imaged with FIB-SEM. They compare the volume and surface area of different segmented organelles, as well as the distance between organelles. A much larger scale example is Heinrich et al.[60] that quantify the morphologies of up to 35 different organelles in segmented FIB-SEM cells. They also calculate the volume, surface area and number of each organelle. They also quantify the number and location of organelle-microtubule contacts, as well as the curvature of the ER and the location of planar and tubular regions.

These sorts of approaches allow cells or organelles to be clustered into different groups based on these predefined features. There are also many more specific options for particular cell types. For example, traced neurons have a tree-like, branching structure, which needs its own custom way of defining morphology. NBLAST[61] is an algorithm to compare the similarity of neurons, and can be used to cluster them into different groups automatically. They represent each neuron as a cloud of points with vectors indicating the direction of individual processes. NBLAST then compares them pair-wise, and gives a similarity score based on the disparity in the neuron position and the orientation of individual segments. It has been used for both light and electron microscopy data.

To summarise, for large scale EM datasets, most groups rely on machine learning approaches to segment images automatically. Otherwise, they rely on manual segmentation with large teams, companies, or crowd sourcing - and probably a combination of both as automatic segmentation methods still require manual training data. Once segmented, there are many ways to quantify the morphology, and group cells into different types. For small scale projects, or ones with a particular question in mind, a few pre-defined features may be enough. Otherwise, a much larger number of pre-defined features can be calculated, or custom algorithms used to compare and cluster cell morphologies.

4.1.2 Comparing morphology to gene expression

Once a quantitative description of the cell morphologies is available, we want to compare to other modalities in the atlas. A common goal is to bridge the gap between cell morphology and gene expression. This has been possible for many years with a small number of genes via CLEM approaches (correlative light and electron microscopy) allowing fluorescence imaging to be matched to EM data[62]. For example, Hoffman et al.[63] recently established a pipeline for correlative cryo super-resolution and FIB-SEM. This allowed very precise matching of 3D EM data of cells, and fluorescent images of ER and mitochondria markers. They demonstrate a variety of fluorescent markers but, as with most EM studies, only 1-2 are imaged at the same time.

There are few examples of large scale matching of EM and gene expression i.e. with the expression of many hundreds of genes, rather than a few chosen markers. An exception would be the Virtual Fly Brain[1] and related *Drosophila* connectomics projects, that use algorithms like NBLAST to match traced neurons to large databases of fluorescently labelled neurons[64]. This is possible as the fly brain is highly stereotypical in its development, allowing matching of data between many individual flies. Similarly, the *C. elegans* community has a complete connectome from EM, as well as genetic information for each neuron[65]. Likewise, this relies on the stereotypical development of the worm.

With the rise of spatial transcriptomics techniques[66], it is becoming ever more possible to have the gene expression of many hundreds, or thousands of genes linked to their spatial location. This opens the door for comprehensive comparison of gene expression and cell morphology. For example, multiplexed single-molecule FISH based approaches like MERFISH[67] and seqFISH[68], can now image spatial expression patterns for around 10,000 genes at sub-cellular resolution[69]. These techniques have yet to be applied on a large scale with EM though, and likely much method development is required to make the sample preparation methods compatible. Otherwise, we again must rely on the stereotypical development of certain model systems to allow comparison of EM and spatial transcriptomics on different samples.

Nevertheless, with the rapid development of such techniques, it is important to develop methods and pipelines to analyse and compare morphology and gene expression at a large scale.

4.1.3 *Platyneries* Atlas

The *Platynereis* atlas provides a great test case for this task - combining EM for a whole organism, with gene expression patterns for over 200 genes. The SBEM volume collected at the FMI in Basel covers a complete 6 day old *Platynereis* at a resolution of 10 x 10 x 25 nm^3. All cells and nuclei within the organism have been segmented by Constantin Pape (Kreshuk group, EMBL Heidelberg), along with larger structures like the ventral nerve cord and gut. This makes it the first EM dataset of an entire organism to have all its cells automatically segmented, and therefore a prime candidate for morphological analysis. In addition, it has a large library of in-situ hybridisations registered to this EM dataset. This gene expression atlas, called ProSPR (profiling by signal probability mapping)[36] contains spatial gene expression patterns for 201 genes and 4 EdU proliferation stainings. Each of these patterns represents an average of whole mount in-situ hybridisations of multiple *Platynereis* imaged at 0.55 micron resolution with a confocal microscope. Therefore, it is also one of very few EM datasets to have gene expression data for a large number of genes. All details are in the published paper[15], and the data can be freely browsed within Fiji using the MoBIE plugin (`https://mobie.github.io/`).

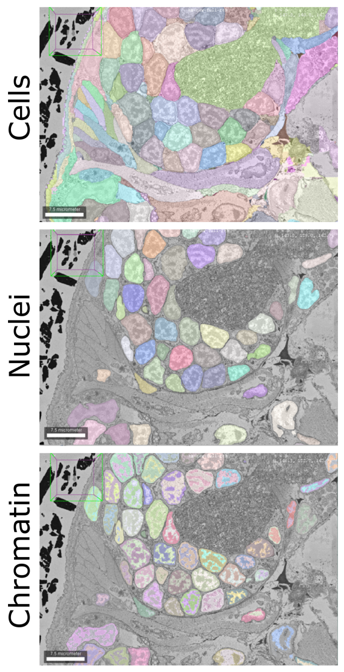


Figure 4.1: Segmentations. Comparison of cell, nucleus and chromatin segmentation for the same region.

4.1.4 Aims

My aim was to establish a pipeline to quantify cell and nuclei morphologies for the entire *Platynereis* atlas. With this, I aimed to automatically cluster all cells in the organism based on their morphology, to identify different subclasses. In addition, I aimed to cluster all cells based on the 201 gene expression patterns in the atlas, and compare this to the cell morphology and tissue boundaries visible with EM.

4.2 Results

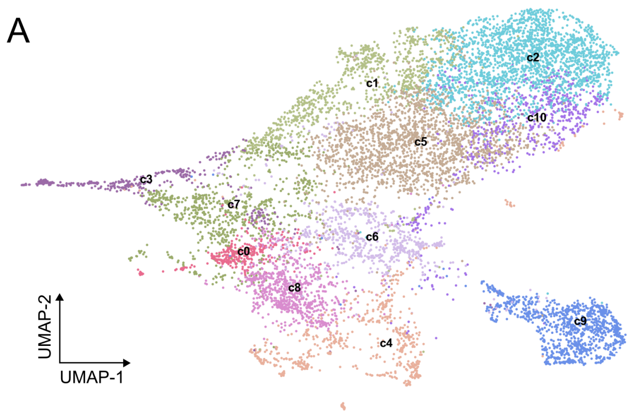
4.2.1 Morphological features

To quantify cell morphologies, I used both the cell and nucleus segmentation created by Constantin Pape (Figure 4.1). As the pattern of dark and light regions in the nucleus also seemed quite distinct, I decided to create a chromatin segmentation for this also. For this, I used Ilastik's[53] pixel classification workflow by manually labelling dark and light regions in a small number of nuclei. The final segmentation separates the dark and light regions, which by classic EM definitions are 'heterochromatin + nucleolus' and 'euchromatin' respectively (Figure 4.1).

To quantify the morphology of all the segmented cells throughout the *Platynereis*, I defined and measured 140 descriptors. This includes features that describe the whole cell, the nucleus of the cell and the distribution of chromatin within the nucleus (see table in Appendix C for details). Features can be split into three broad categories: shape, intensity and texture. For example, shape would include the volume and surface area, intensity would include the mean and standard deviation of intensities, and texture would include the haralick texture features[70].

4.2.2 Morphological clustering

Once features were calculated for all the segmentations, I proceeded to cluster all cells using a graph-based approach. I constructed a K-nearest neighbour graph using the euclidean distance between the feature vectors. This was followed by community detection (with the Louvain method[71]) to give 11 final clusters. All clusters were then visualised with UMAP (Uniform Manifold Approximation and Projection for Dimension Reduction)[72] (Figure 4.2A).



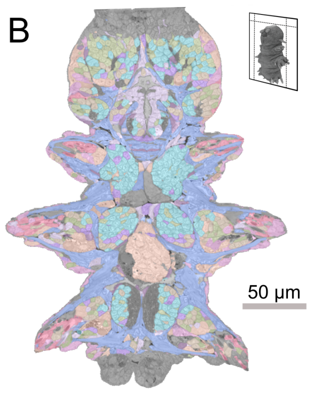


Figure 4.2: Morphological clusters. ***A*** - UMAP of all cells based on morphological features, coloured by morphological clusters (c0-c10) ***B*** - Morphological clusters mapped onto an EM section.

Mapping these same clusters back onto the EM, showed that clusters mostly correspond to broad cell types like neurons, muscles etc. (Figure 4.2B). To analyse this further, Valentyna Zinchenko, Rachel Templin and myself manually labelled 978 cells as belonging to the following cell classes: muscle, epithelial, secretory, digestive, ciliated, neuron and dark cells (Figure 4.3). Secretory cells were any cells that had secretory vesicles / inclusions or large amounts of ER and golgi. Ciliated cells were multiciliated cells on the *Platynereis* surface that contained large numbers of mitochondria. Dark cells were cells with extremely dark / featureless nuclei. Finally, digestive cells were large cells found in the *Platynereis* midgut.

Mapping these manually classified cells back onto the UMAP, showed that they mostly occupy distinct regions (Figure 4.3). This demonstrates that the morphological descriptors can identify these cell classes well. The only main exception is the secretory cells, that are more widely spread in the UMAP. This is not very surprising though, as this class was very broadly defined as anything with secretory structures or plentiful ER/golgi. In all likelihood, this class is actually a number of different morphological classes mixed together.

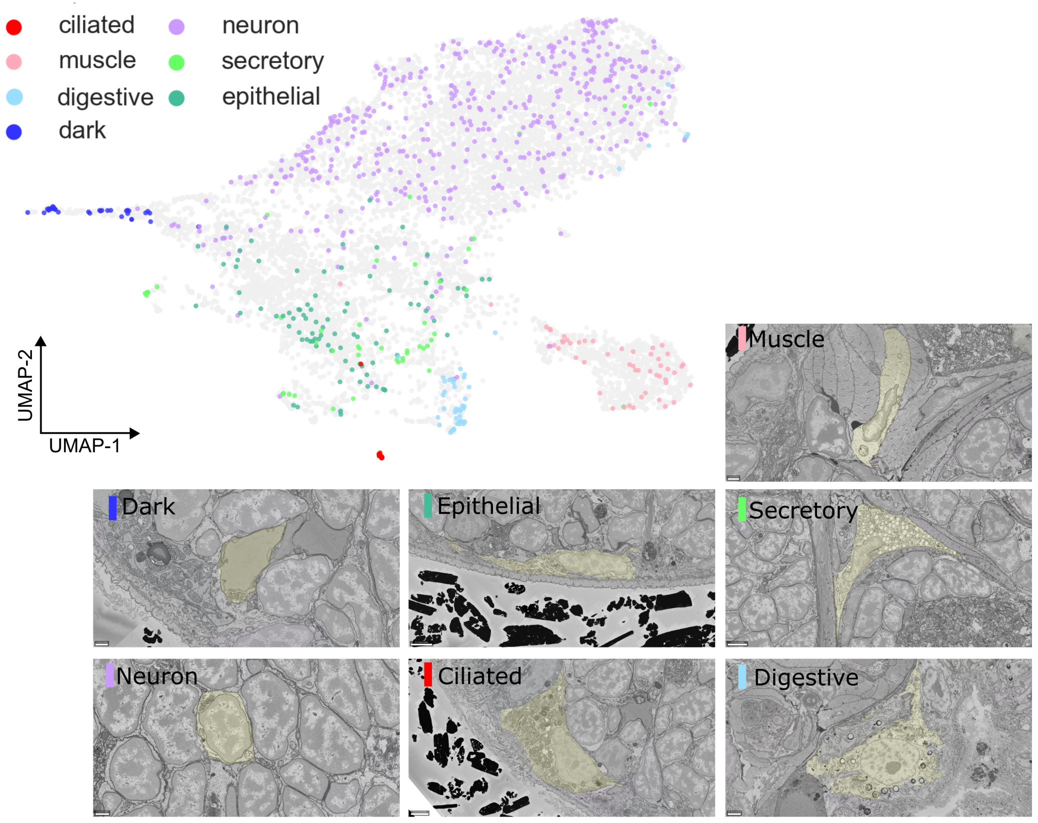


Figure 4.3: Manually labelled cell classes. ***A*** - 978 manually identified cells mapped onto the same UMAP as Figure 4.2A. At the bottom are small examples of each of the labelled classes, with the cell of interest highlighted in yellow.

Once these cell classes were defined, I wanted to understand how the morphological features varied between them. For this, I plotted various morphological features as violin plots separated by cell class (Figure 4.4A). For example, it could be seen that the ciliated and digestive cells have larger nuclei than all other cell classes, and also a larger heterochromatin+nucleolus surface area. Interestingly, I saw that the total heterochromatin+nucleolus volume stays quite constant across all cell classes, while the surface area does not. This implies that the organisation of the dark and light chromatin regions is different between the cell classes in a consistent way i.e. in ciliated and digestive cells the heterochromatin+nucleolus is more spread out giving it a larger exposed surface.

Moving away from our manually labelled subset of cells, general trends for these features can also be seen across all nuclei (Figure 4.4B-C). For example, nucleus volume is positively correlated to the surface area to volume ratio of heterochromatin+nucleolus. This implies some relation between the size of a nucleus and the chromatin organisation in *Platynereis*.

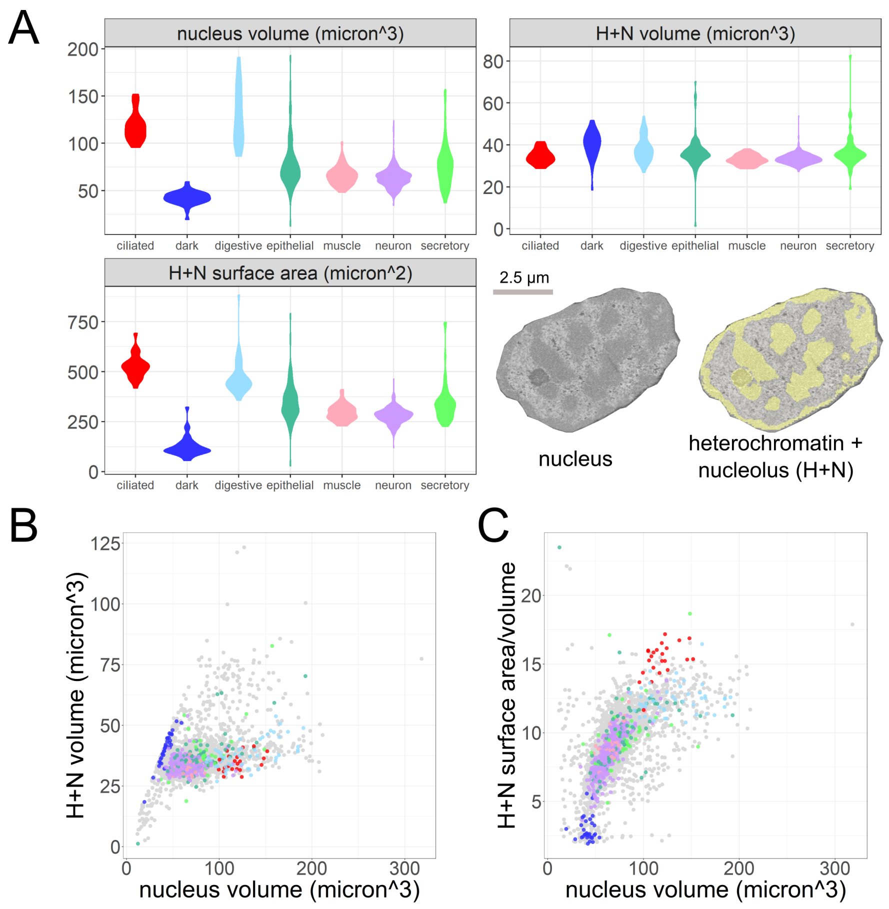


Figure 4.4: Characteristic morphological features. ***A*** - Violin plots comparing 3 morphological features between the manually identified cell classes. Bottom right: example nucleus with chromatin segmentation. ***B-C*** - Scatterplots relating morphological features between all cells, with manually identified cells shown in the same colours as panel A and Figure 4.3.

Next, I wanted to analyse whether the morphological features could be used to recognise bilateral pairs of cells in *Platynereis*. *Platynereis* is a bilaterally symmetric organism, with many cells being easily identified visually as pairs across the midline (Figure 4.5A). I established a set of criteria to identify a cell as a potential bilateral partner - this relied on the mirror image position of the cell across the midline (see section 7.3.4 of the methods for details). I then took every cell and ranked all other cells by distance in morphology space from nearest neighbour, to most distant neighbour. Then, for each cell I found the closest neighbour in the ranking that also met the criteria for being a potential bilateral partner (Figure 4.5B). Overall, this provides a proxy for assessing how well morphological features can find bilateral pairs i.e. if bilateral pairs are fewer neighbours apart, then the morphological features are better at finding them. This analysis showed that with all morphological features used, 14% of cells found a potential bilateral partner within the 5 nearest neighbours (20% within 10 and 28% within 20 nearest neighbours). This shows the morphological features can identify at least some bilateral pairs - and is in fact quite high given that there are more than 10,000 possible partners for each cell, and many cells (like the neurons that only have cell bodies segmented) are morphologically very similar.

To investigate which morphological features perform the best for finding bilateral pairs, I repeated this analysis with different feature subsets. For example with only the cell features, only the nuclear features (shape, intensity, texture and chromatin distribution) or chromatin features alone (intensity and texture). This showed that nuclear features performed the best, followed by chromatin and then cell features. Interestingly, intensity and texture of the chromatin alone could still find many bilateral pairs e.g. 7% of cells found a potential partner within 5 nearest neighbours, 12% within 10 and 18% within 20. This implies that nuclear features, and chromatin properties, can be quite consistent/characteristic between bilateral pairs of cells, and therefore potentially between different cell types.

4.2.3 Gene expression clustering

To compare to the morphological clusters, I also clustered all cells by gene expression. ProSPR gives spatial gene expression patterns that are registered to the same space as the EM volume. Therefore, each segmented cell could be assigned an overlap value for each gene, denoting the fraction of the cell's volume that contains gene expression signal.

I again used a graph-based clustering approach (same as for the morphological clustering), and visualised the results with UMAP (Figure 4.6A). This gave 15 clusters that occupied spatially coherent zones in the *Platynereis*. Mapping these same clusters directly onto the EM showed they largely respect major tissue boundaries (Figure 4.6C). For example, two clusters span the ventral nerve cord, and remain within the tissue boundaries of the ventral nerve cord on the EM. Mapping segmented anatomical regions like the head, midgut, ventral nerve cord etc. onto the same UMAP again demonstrated that clusters separate the main tissues (Figure 4.6B).

To analyse further the gene expression, and how it relates to tissue boundaries, I focused on the head of the *Platynereis*. This region has the highest density of genes in the atlas, and therefore should have the finest subdivision based on gene expression. This high gene density is a result of ProSPr's bias towards nervous system focused genes; ProSPr's genes were hand-selected by members of the Arendt lab whose research centres around the nervous system.

Looking at the EM directly, different sub-compartments can be seen in the head sep-

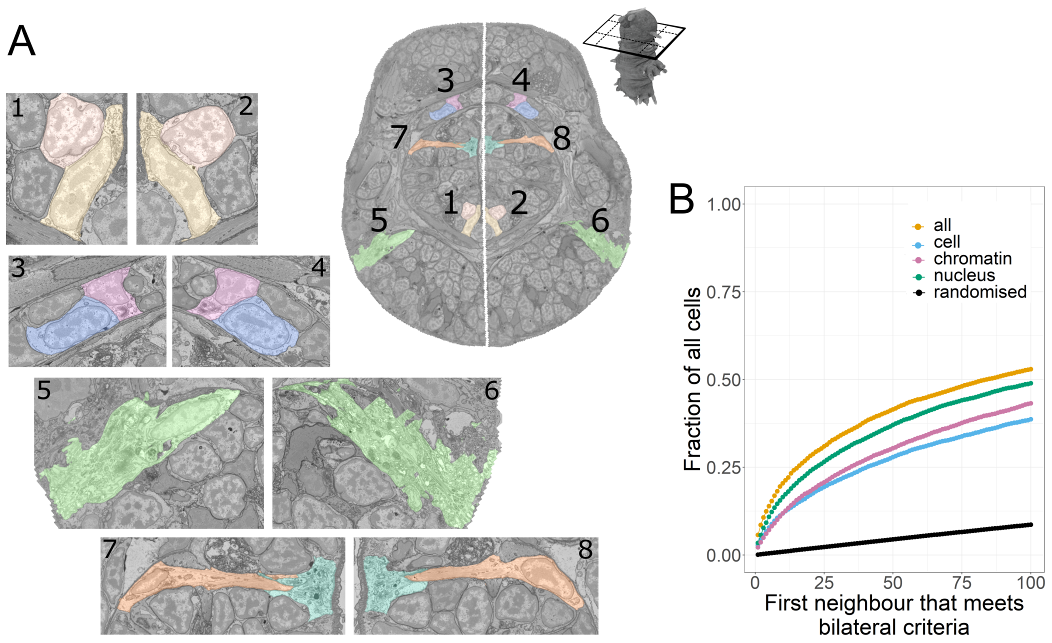


Figure 4.5: Bilateral pairs. ***A*** - Example EM section showing manually identified bilateral pairs (midline shown in white). Zooms are provided of each of the individual cells. ***B*** - Fraction of all cells finding potential bilateral partners within a certain number of morphological neighbours. 'all' is all morphological features, 'cell' only cell features, 'nucleus' all nucleus and chromatin features and chromatin only intensity and texture features of chromatin. The randomised line is the mean of 100 trials of randomly shuffling the ranking of cells in morphology space, and performing the same pairs analysis.

arated by clear tissue boundaries. These 'ganglionic nuclei (GN)' were manually segmented from the volume by Hernando Martinez Vergara (former member of the Arendt Lab at EMBL) based on the EM data alone (Figure 4.7A).

Comparing these morphologically defined GN to the gene expression clustering, showed a clear correspondence between the two (Figure 4.7B). For example, cluster 2 closely matches the extent of the mushroom body GN when viewed on the UMAP and in the EM. Also, cluster 4 closely matches the extent of the palpal GN. To quantify this, I calculated a specificity score for each genetic cluster and gene for every GN (Figures 4.8 and 4.9). This specificity score is a combination of two measures: first, the fraction of expression confined to a given ganglion (A) and second, the fraction of that ganglion covered (B). In analogy to the F1 score, specificity is calculated as: $\frac{2AB}{A+B}$

Analysis of these scores showed that for almost all GN, specificity values for gene clusters were higher than for any individual gene. This implies that a combination of genes are used to define a particular GN, rather than any one clear 'marker gene'.

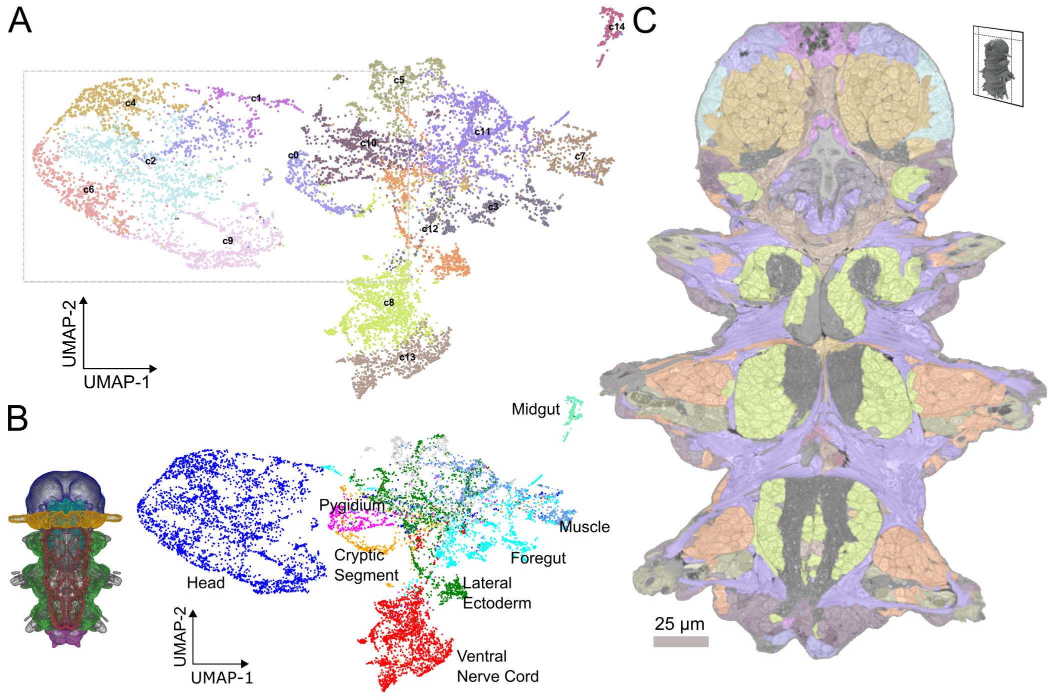


Figure 4.6: Gene clusters. ***A*** - UMAP of all cells based on the expression of 201 genes. Points are coloured by gene expression clusters (c0-c14). The gray rectangle shows the part of the UMAP repeated in Figures 4.7, 4.8 and 4.9. ***B*** - Main body parts in *Platynereis* with matching colours between animal regions and UMAP. ***C*** - Gene clusters mapped onto a section of the EM.

4.2.4 Pipeline

Most steps of this workflow are implemented as Snakemake[73] workflows to make them easy to re-run on new versions of the *Platynereis* atlas. Snakemake is a python based workflow management system that enables easy management of long analysis workflows with many individual steps. All other analysis steps are provided as individual scripts. See methods sections 7.3.2 and 7.3.5 for details.

4.3 Discussion

To summarise, I established a pipeline to quantify cell and nuclei morphologies for the entire *Platynereis* atlas. This allowed automatic clustering of all cells, that could separate a number of cell classes such as muscles, neurons and ciliated cells. Various features such as nucleus volume, and the surface area of heterochromatin+nucleolus were found to be characteristic for particular cell classes. Also, a general relationship between nucleus volume and heterochromatin+nucleolus surface area to volume ratio was observed. In addition, these features could be used to find back some bilateral pairs of cells, including chromatin features.

Next, I turned to the gene expression side of the atlas, and clustered all cells again. This

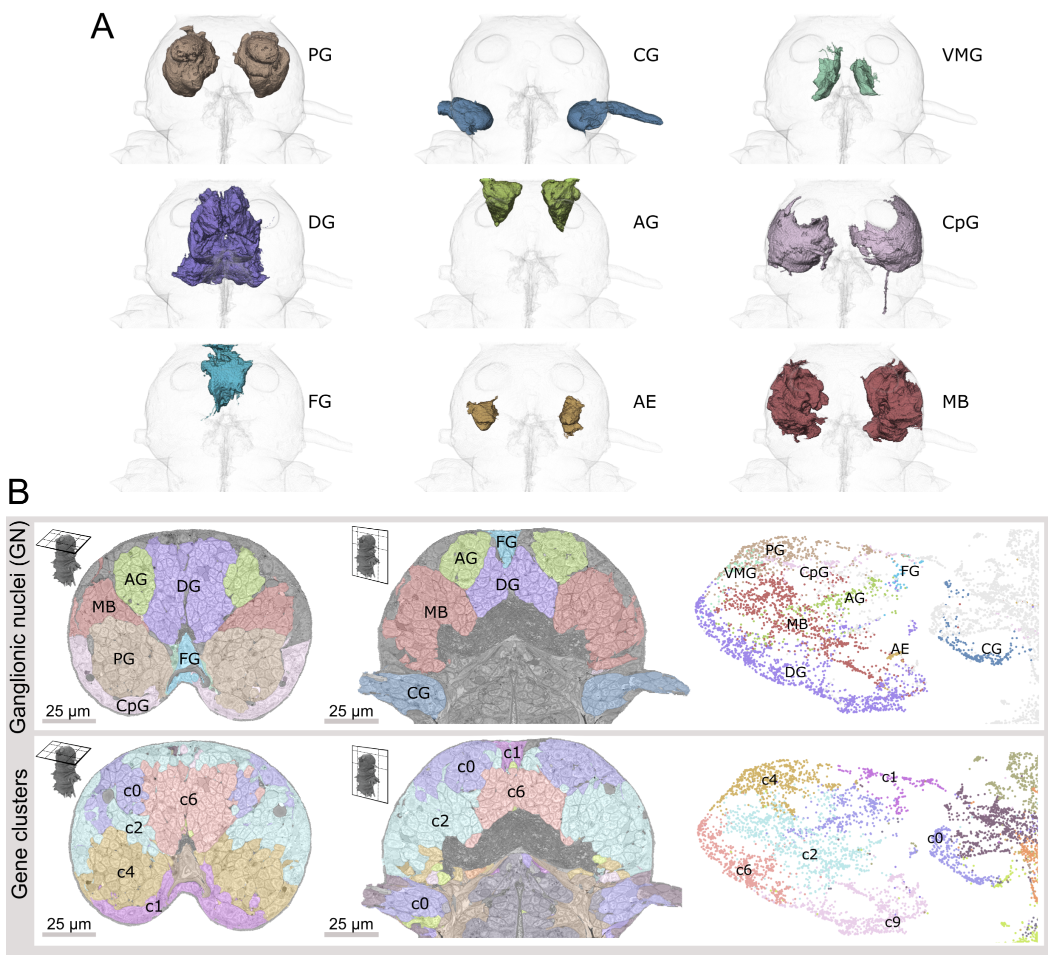


Figure 4.7: Ganglionic Nuclei (GN). ***A*** - 3D images of each GN. ***B*** - Comparison of GN (top) and gene clusters (bottom) mapped on transversal (left) and horizontal (middle) head section and UMAP (right). Abbreviations: AEs, adult eyes; AG, antennal GN; CG, cirral GN; CpG, circumpalpal GN; DG, dorsal GN; FG, frontal GN; MBs, mushroom bodies; PG, palpal GN; VMG, ventro-medial GN.

demonstrated that genetic clusters formed spatially coherent zones in the *Platynereis* and mostly respect tissue boundaries visible in the EM. This relationship was particularly clear in the *Platynereis* head where morphologically defined GNs correlated to gene expression clusters.

Overall, this provides an example framework for comparing morphology from EM and gene expression on a large scale. There are a number of improvements that could be made though. First of all, the snakemake workflows that were produced are quite specific to this *Platynereis* atlas. It would be useful to make this more generalisable, so it could be applied to other atlases in a simple way. This is a rather difficult task though, as datasets with EM and large-scale gene expression are still very rare.

Another goal would be to push the resolution of both the morphology and gene ex-

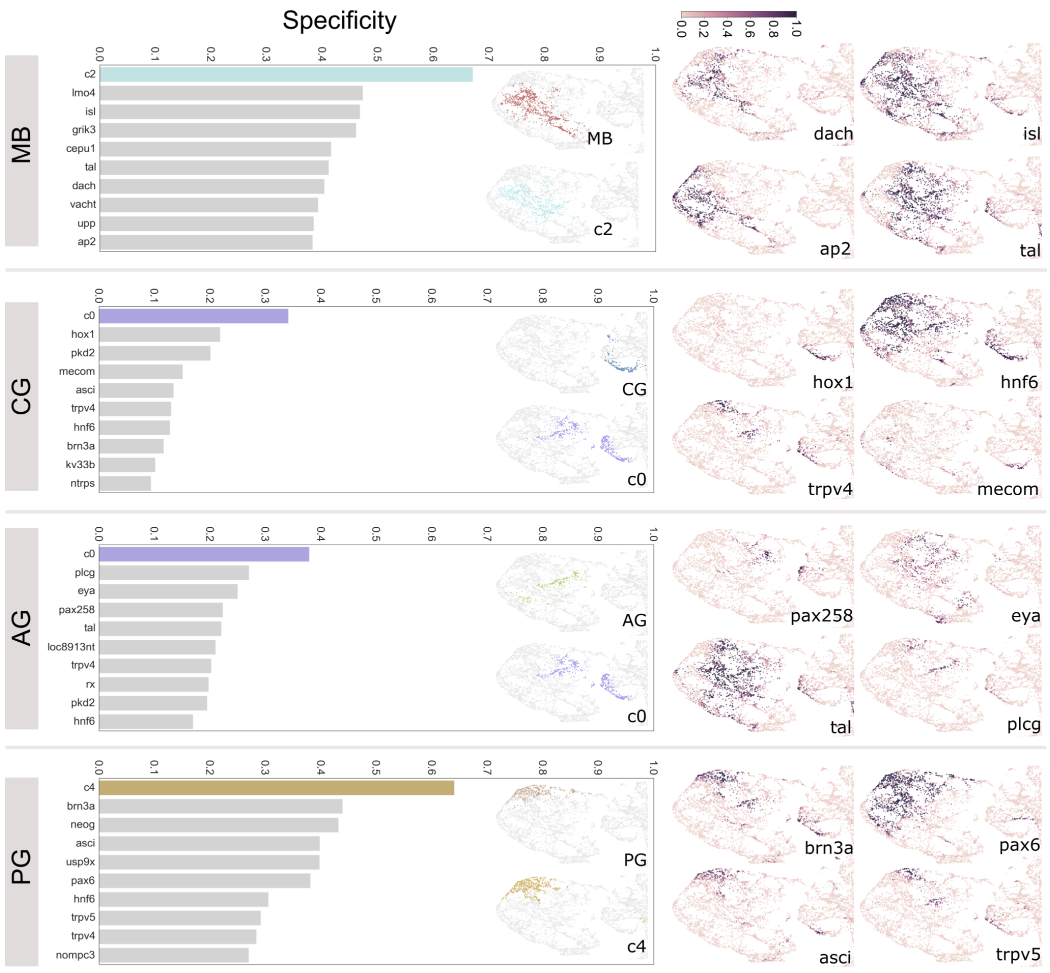


Figure 4.8: Specificity scores. Left column - graphs of top 10 scoring genes (gray bars) or gene clusters (coloured bars) by F1 specificity score. Inset - zooms of the head region of the UMAP from Figure 4.6 coloured by GN (top) and top scoring genetic clusters (bottom). Right column - gene expression overlap value for example genes. Note: here I only show the head region of the UMAPs, for easier comparison, but some genes and gene clusters have expression domains outside of the head that contribute to their lower specificity scores.

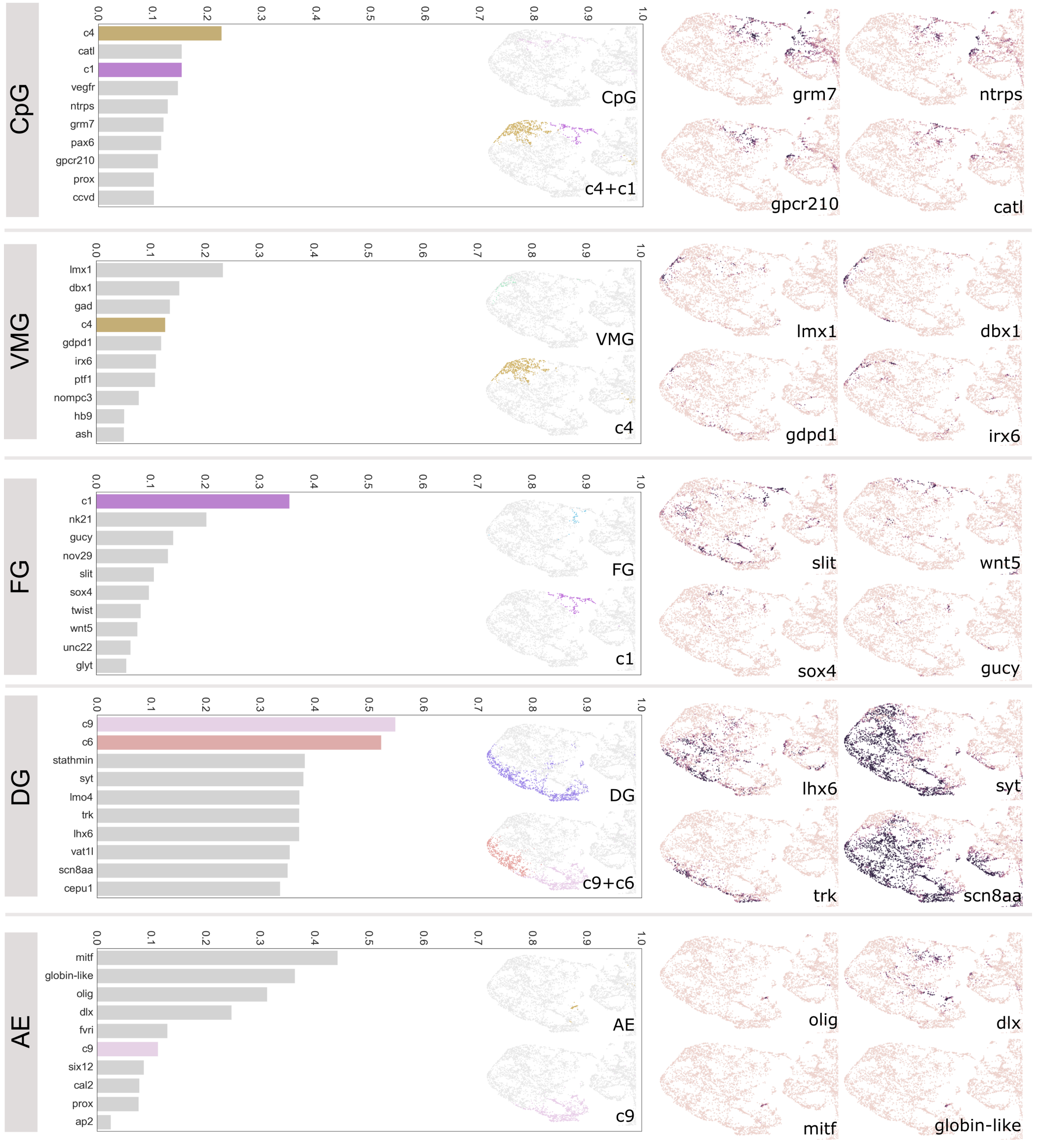


Figure 4.9: Specificity scores. Continuation of figure 4.8

pression that can be compared. For example, here we only look at the scale of cells and nuclei, but EM provides the resolution to look at much finer scale details like the location and orientation of different organelles. It would be interesting to segment various organelles from this dataset, and see how their morphologies vary over the organism. On the gene expression side, it would be interesting to expand the number of genes available in the atlas. Ideally, this ProSPr resource could be used to map single cell sequencing data to its spatial location. This would allow a much more detailed comparison of how morphology varies with gene expression. This kind of mapping has been done previously for a number of organisms[74][75][76], and also the 48 hour old *Platynereis*[77].

In addition, it would be useful to expand the analysis of the relationship between morphology and gene expression. Here, we mostly rely on comparison of gene expression clustering and morphological boundaries in the EM, but it would be interesting to do a more direct comparison of morphological features and individual genes. For example, some of the techniques used in single cell 'multi-omics' for comparing multiple datasets might also be useful for this task[78].

Chapter 5

Resource Sharing: managing massive, multimodal data

5.1 Background

Once the data has been analysed, we move to the final stage of the lifecycle - sharing it as an open resource for the community. This is an extremely challenging step for large multimodal atlases, especially those that contain EM data that can be many terabytes in size. Not only is the size of data challenging, it's diversity is also a problem - how do we display images of different sizes, resolutions and modalities in a useful way? Also, how can we make the measurements and graphs derived from these datasets accessible?

5.1.1 Image Archives

There are a number of solutions to this problem. The first is to deposit these atlases in public databases like BioImage Archive[79], EMPIAR (Electron Microscopy Public Image Archive)[80] and IDR (Image Data Resource)[81]. These archives provide long-lasting open storage of large datasets, with each dataset tied to a unique identifier so they can be cited and re-used easily in future studies. These archives are essential for supporting good scientific practice, and keeping imaging data open to all.

There are limitations to these archives though. Each is accessible through a website that allows browsing of datasets based on their associated metadata e.g. which organism was imaged or what kind of imaging method was used. Usually, you can easily view which files are available, their formats, their size and metadata, but there are limited ways to browse the images themselves without downloading them to your own computer. EMPIAR provides a Volume Browser which allows individual images to be viewed and browsed from 3 orthogonal directions, along with a 3D rendering and segmentation overlay. IDR provides more browsing capabilities, supported by OMERO.iviewer[82] allowing images to be browsed in the z direction, their brightness, contrast and look up tables adjusted, as well as some simple analysis like viewing the histogram of different image channels. Multiple images can also be viewed side by side.

These browsing capabilities allow researchers to investigate the datasets before downloading large amounts of data, and are a great addition to these archives. They are quite limited though - i.e. only one image can be viewed at a time, or only side by side. Different sizes and shapes of registered images cannot be easily viewed in the same space for comparison, and cannot be sliced in arbitrary directions. Also, the analysis that can be done is quite limited i.e. it can be difficult to browse measurements made from the images alongside the images themselves. This is not unexpected - image archives must handle a massive variety of images, with a large variety of file formats and metadata. It does mean that more flexible tools to browse remotely stored images would be desirable.

5.1.2 Tools for browsing remote images

A number of tools already exist for browsing and analysing images stored remotely. Many of these come from the connectomics field, that had an early need for remote access to massive terabyte sized EM datasets. For example, CATMAID[83] allows browsing of massive EM data, along with many analysis tools customised for tracing neurons. It allows many researchers, at various locations worldwide, to trace neurons in the same dataset in a collaborative manner. Its analysis tools are also very developed allowing 3D rendering of neurons, alongside browsable tables of measurements, graphs and network diagrams.

With a similar aim, WebKnossos[84] allows collaborative tracing of neurons in image data stored remotely. As with CATMAID, it also supports 3D rendering of neurons and some analysis.

A more general tool for browsing remote images is Neuroglancer[85], which is also the viewer behind the online OpenOrganelle atlas[60][86]. This allows massive EM data to be browsed alongside segmentations, and any other kind of image data. It also allows viewing at arbitrary orientations, 3D rendering and much more control over the appearance of image layers.

These options still have their limitations though. For example, the analysis tools in CATMAID and webKnossos are very focused on neurons and their morphology. Also, the software is optomised for this very specific task. While Neuroglancer is much more general, it also provides limited browsing of, for example, tables and graphs of measurements alongside the original image data. Also, while most of these options are very user-friendly to use via a web-based interface, they are challenging to setup with your own data, especially for researchers with no programming experience.

5.1.3 Image formats

All the options discussed above rely on chunked, pyramidal file formats to efficiently allow browsing of massive image data. This means that each image is split into a series of smaller chunks that can each be loaded individually. This means that if, for example, you are browsing only one corner of the image, only that part will be accessed rather than loading the entire image. This makes it much more memory efficient for these terabyte sized datasets.

Also, images are stored at a series of different resolution levels, all the way from the massive full resolution data, to tiny very downsampled versions. This means that different levels of resolution can be accessed depending on how closely you are browsing the data. When you are very zoomed out, low resolution data can be shown, with greater resolution being loaded as you zoom further in. This is a similar concept to how google maps allows efficient browsing of its huge data.

A number of different formats exist including HDF5[87], n5[45] and OME-NGFF[88].

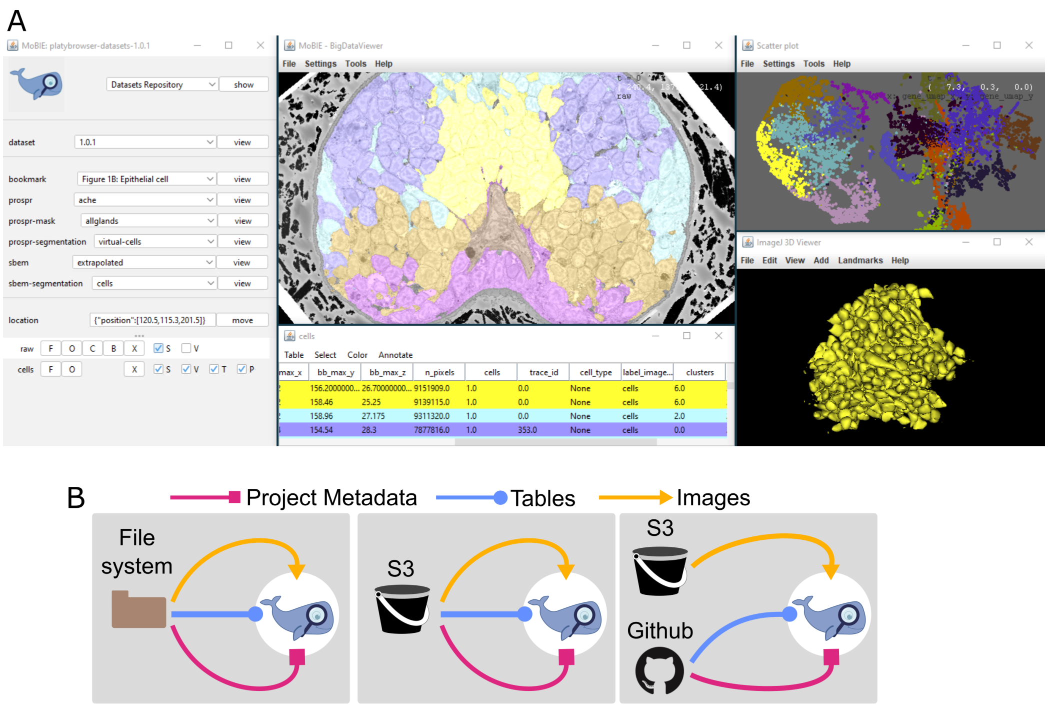
5.1.4 MoBIE

MoBIE (MultiModal Big Image Data Sharing and Exploration) is a Fiji[37] plugin that allows browsing of massive image data stored remotely or locally (`https://mobie.github.io/`). It builds on top of BigDataViewer[38], to allow browsing of 2D slices through images, and the ImageJ 3D viewer for 3D rendering[39]. It was created by Christian Tischer and Constantin Pape for the *Platynereis* atlas that I discussed in the previous chapter[15]. After the completion of this atlas, I became involved in MoBIE's development and I'm now one of the main maintainers, adding new features on a regular basis.

MoBIE allows browsing of images of different sizes, resolutions and modalities in the same coordinate system. It also supports both interactive tables and graphs to allow measurements derived from images to be easily browsed alongside the images themselves. It also supports viewing images and segmentations in 3D, as well as extensive control over how images are displayed.

Perhaps MoBIE's greatest advantage is that it is available within the familiar framework of ImageJ/Fiji. Fiji is used extensively within the biological community for image analysis, and this makes MoBIE easily accessible to researchers with no programming experience. In addition, as many researchers already know how to program Fiji plugins, it is more easily extensible than web-based solutions that are often less familiar to biologists. Also, it can interact more easily with all the image analysis tools already in Fiji, to provide more analysis options.

While using MoBIE is user-friendly for those with no programming experience, creating a MoBIE project with your own data is not. Currently, these are created using a python library created by Constantin Pape (`https://github.com/mobie/mobie-utils-python`). This python library is simple to use for those familiar with the Python programming language, and offers efficient conversion of images to chunked file formats like n5. It can be run on a computer cluster for rapid, parallel conversion of large images.



MoBIE: platybrowser-datasets-1.0.1

	Datasets Repository	show
dataset	1.0.1	view
bookmark	Figure 1B: Epithelial cell	view
prospr	ache	view
prospr-mask	allglands	view
prospr-segmentation	virtual-cells	view
sbem	extrapolated	view
sbem-segmentation	cells	view
location	{"position":[120.5,115.3,201.5]}	move

raw F O C B X ☑ S ☐ V
cells F O X ☑ S ☑ V ☑ T ☑ P

cells
Table Select Color Annotate

max_x	bb_max_y	bb_max_z	n_pixels	cells	trace_id	cell_type	label_image...	clusters
2	156.2000000...	26.70000000...	9151909.0	1.0	0.0	None	cells	6.0
2	158.46	25.25	9139115.0	1.0	0.0	None	cells	6.0
2	158.96	27.175	9311320.0	1.0	0.0	None	cells	2.0
4	154.54	28.3	7877816.0	1.0	353.0	None	cells	0.0

Figure 5.1: MoBIE ***A*** -The MoBIE user interface, displaying the *Platynereis* atlas. Left panel are the controls allowing different images and segmentations to be viewed, and their display settings changed. Top middle is the image viewer, showing a slice through the *Platynereis* head coloured by the gene expression clustering. Bottom middle is the interactive table, coloured by the same clusters. Top right, is the interactive scatter plot showing the gene expression UMAP. Bottom right is the 3D viewer, showing all the cells in gene expression cluster 6. ***B*** - MoBIE projects are made up of three main pieces of data: the project metadata, tables, and images. Here, three of the most common setups are shown - loading data from either the file system, S3 object storage or github.

5.1.5 Aims

My aim was to make creating a MoBIE project accessible to researchers with no programming experience. For this, I would add to the existing Fiji plugin to create a simple user interface that would allow images to be converted to chunked file formats. This would also produce all the required metadata for MoBIE in a user-friendly way. As this must run within ImageJ/Fiji, it was made for projects where individual images would fit within the memory of a normal computer or workstation. It is therefore complementary to the python library mentioned above - making project creation simple for smaller projects, while the python library covers the larger projects that cannot fit within memory.

5.2 Results

5.2.1 Project Creator

A MoBIE project is made up of images, tables and metadata, stored in a standard folder structure. This data is grouped into 'datasets', which are sets of images and tables that can be viewed at the same time within MoBIE. For example, each dataset could be a different experiment, or data from a different type of sample. Each dataset contains a number of 'sources', which are the individual images stored in chunked, pyramidal file formats. It also contains tables stored as tsv (tab separated values) or csv (comma separated values) files. To determine how these images and tables are displayed, there is a variety of metadata stored as JSON files - namely a 'project.json' storing information about the overall project, and individual 'dataset.json' storing information about each dataset.

The display of sources and tables, centres around the idea of 'views' that are stored in the dataset.json. Each view determines a combination of sources and tables to display, along with metadata about how to display them e.g. their brightness, contrast, opacity, and whether to display in 3D. Views are grouped in the MoBIE user interface into different drop-down menus, allowing views to be organised into different categories. For example, in the *Platynereis* atlas, there are groups for ProSPr (i.e. all the gene expression data), and SBEM (i.e. all the EM data) (Figure 5.1A). This structure of files and metadata, is what the project creator needs to produce.

To do so, I added a simple user interface to the Fiji plugin (Figure 5.2A). This shows the different components of a MoBIE project as different rows: dataset (the names of each dataset), source (the names of the images stored in each dataset), group (the names of the MoBIE drop-down menus for each dataset) and view (the names of the views in each dataset). These rows are coupled so that selecting a different dataset, will update the sources shown, and so on. A number of buttons allow datasets and sources to be added and edited.

Starting with the top row, the project creator allows datasets to be created and edited in a user-friendly way. For example, dataset names, and which appears by default in MoBIE can be changed.

The button on the 'source' row allows images to be added to the selected dataset, and converted to a compatible format. For example, images can be opened in Fiji (as you would normally for any image analysis), and then converted to either n5 or OME-NGFF file formats. The plugin automatically suggests good defaults for chunk sizes, and resolution levels, making this process as simple as possible. I also allow these parameters to be entered manually though, if finer control is needed. For images that are already in compatible formats, I also support linking (i.e. leaving the images where they are, and pointing to this location), copying and moving into the project. This prevents images being converted multiple times to compatible formats.

For either of these methods, various metadata can also be specified for the added source. For example, the source name can be specified, along with an arbitrary affine transform. This can be useful to align sources of different shapes and sizes in the viewer. The project creator can also automatically produce a view for a source, so it appears directly in the MoBIE drop-down menus. The group can be specified, to sort views into different categories.

In addition, for segmentations, the project creator automatically produces a table with the location and bounding boxes of each label. This allows the segmentation to be browsed interactively in MoBIE with no extra work.

Via these menus, the entire structure of a MoBIE project can be made locally on your computer - including all necessary images, tables and metadata. This project can be opened, and browsed, directly in MoBIE using the button in the bottom right of the project creator.

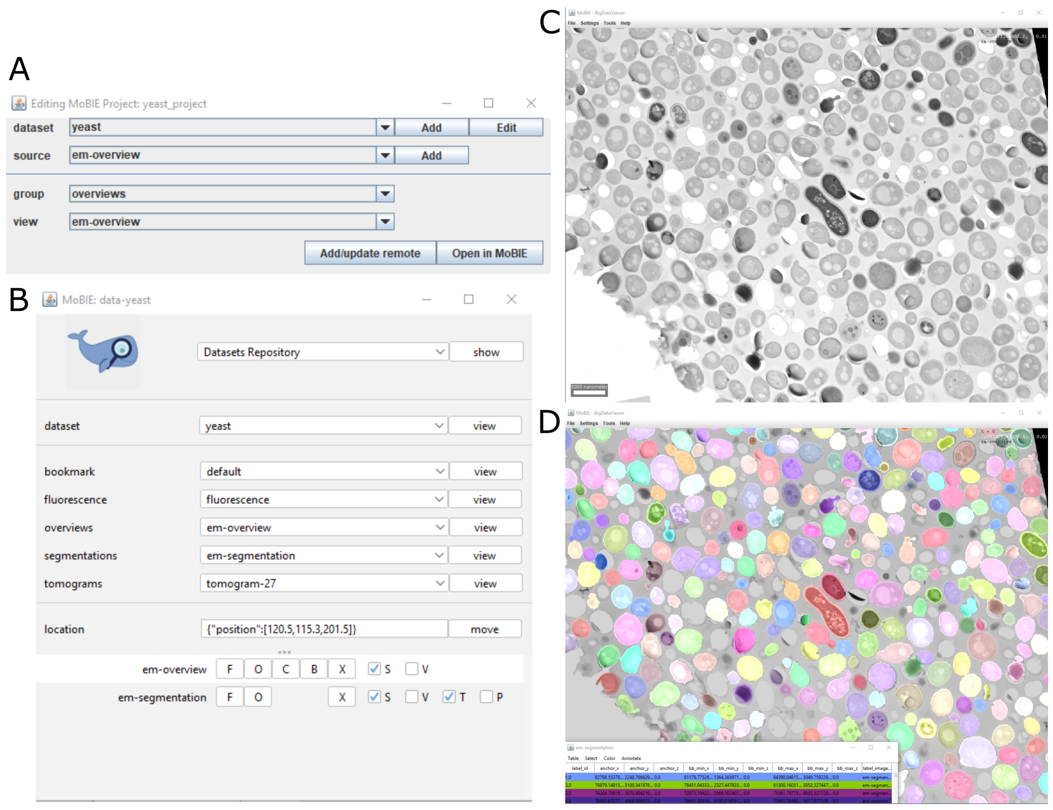
Finally, if this project needs to be accessed remotely, it can be uploaded in its entirety to an S3 object store. The project creator contains options to add the required metadata including e.g. the location of the object store, and the name of the bucket. Once complete, anyone can access the project remotely with only the S3 location and the MoBIE plugin installed.

To demonstrate this workflow, I made a simple example project using the same test data as the tutorials for the python plugin for MoBIE project creation. This data comes from a publication focusing on yeast lipid droplet biogenesis[89]. This small dataset contains a 2D overview image of yeast from EM, along with a yeast segmentation, fluorescence image, and two 3D EM tomograms. It is therefore a nice test case for images of different sizes, dimensions and types. Some snapshots from this project can be seen in Figure 5.2B-D. The project creator allowed all images to be converted, sorted into sensible groups in the MoBIE user interface and interactive tables made for the segmentation. In addition, it allowed affine transforms to be added for both tomograms to display them at the correct position in the EM overview.

5.2.2 Views

Once the project has been made with the project creator, various display settings can be changed within MoBIE and saved to the project. Every entry in the MoBIE menu is a ‘view’ i.e. a collection of one or more images, and the settings determining how they are displayed (e.g. brightness, contrast, which tables to show, and whether to display in 3D). The system for reading and saving views was a joint effort between Christian Tischer, Constantin Pape and myself.

To make it easier for people to make bookmarks, I added an option in MoBIE to save the current settings as a view. This allows people to modify and create new views without manually editing the metadata files, or needing programming knowledge. This is provided as a simple user interface that allows various metadata to be specified such as: the view name, which group it should appear in, and whether the current transform should be included. This allows views to be saved directly to a project (created from



Editing MoBIE Project: yeast_project

dataset: yeast — Add — Edit
source: em-overview — Add
group: overviews
view: em-overview

Add/update remote — Open in MoBIE

MoBIE: data-yeast

Datasets Repository — show
dataset: yeast — view
bookmark: default — view
fluorescence: fluorescence — view
overviews: em-overview — view
segmentations: em-segmentation — view
tomograms: tomogram-27 — view
location: {"position":[120.5,115.3,201.5]} — move

em-overview: F O C B X ✓ S V
em-segmentation: F O X ✓ S V ✓ T P

Figure 5.2: Project Creator ***A*** - Project creator user interface. There are separate rows showing the current datasets, sources (images), groups (i.e. drop-down menus in MoBIE) and views (collections of images and display settings). The add/edit buttons are used to add or edit datasets and images. You can also add metadata for remote access, and open the project directly in MoBIE. ***B*** - MoBIE user interface for an example project made with the project creator. Different groups were made for fluorescence images, em overviews, segmentations and em tomograms. ***C*** - em overview shown in MoBIE. ***D*** - em segmentation shown with automatically generated table in MoBIE.

the project creator, or the python library), as well as to any other location on the file system. This is useful, for example, if you are working on a remote project that you don't have access to - and you want to save particular locations and orientations of interest without having to add them to the main project.

Figure 5.3 provides an example for the yeast project mentioned above. When this project is created by the project creator, all images are displayed with default settings (panel A). Modifying various parameters in MoBIE such as the colour of the fluorescence, and the brightness and contrast, can give a much more informative display (panel B). Using the 'save current settings as view...' dialogs, this new display can be saved as a view directly to the project, so that it is accessible via the MoBIE menus the next time the project is opened.

In this way, complicated views containing different regions of interest, combinations of images, and tables/plots can be created in a user-friendly way.

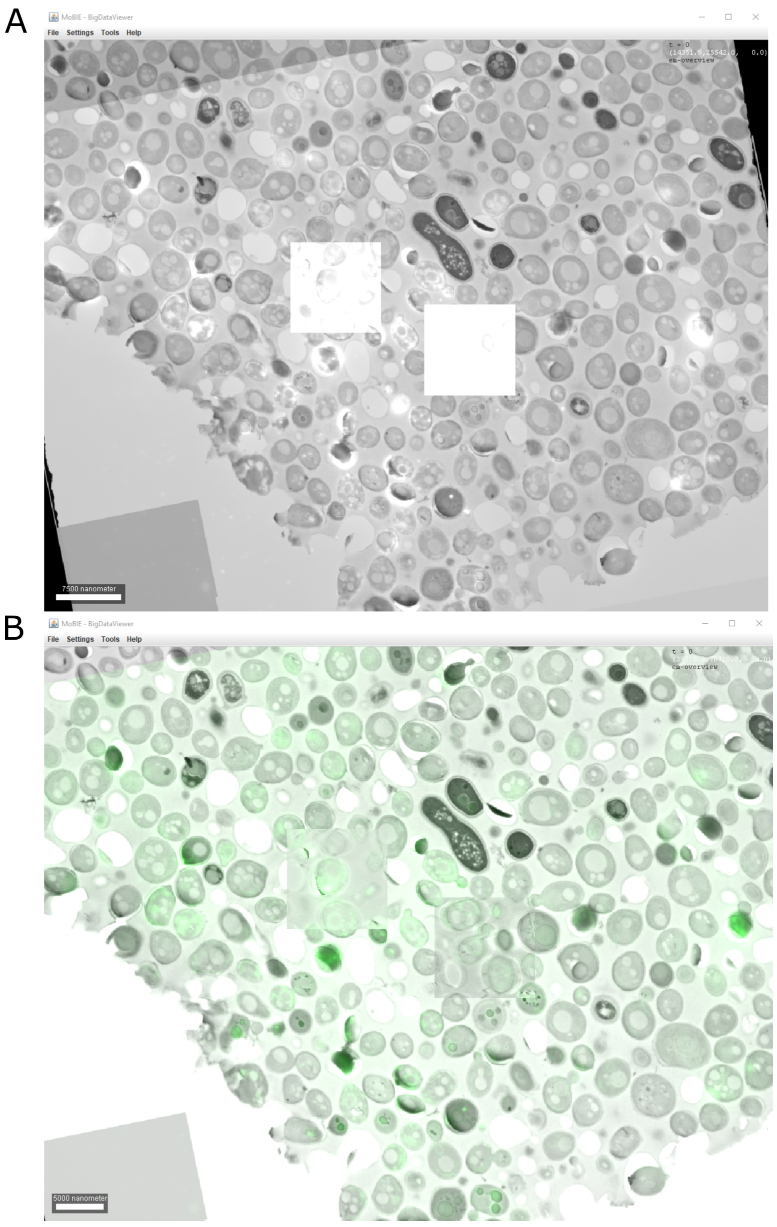


Figure 5.3: Views. ***A*** - Example yeast project open in MoBIE directly from project creator. The EM overview, fluorescence image, and two smaller tomograms are displayed. ***B*** - Same as *A*, but with changed colour of the fluorescence image, and changed brightness and blending mode of the two tomograms. This can be saved as a view, so it is directly accessible in the MoBIE drop-down menus for any user.

5.3 Discussion

To summarise, I added a user-friendly project creator to the MoBIE plugin. This allows creation of projects, and the required metadata, with no programming experience. Projects can be further customised by modifying the display directly in MoBIE, and saving a series of views. These views can also be created in an easy way, via the options I added to the main MoBIE user interface. This project creator focuses on images that can fit in the memory of your computer or workstation, while the python library provides efficient conversion for much larger datasets.

There are a number of improvements that could be made to the project creator. For example, at the moment images must be added one by one. For projects with many images, it would be useful to provide options that allow adding images in batches. Also, it would be useful to provide more options to modify existing projects. For example, moving images between different datasets or groups, or removing certain views or images.

Looking more widely at MoBIE itself, and other tools that allow remote access to images, it would be very useful to see greater integration with image archives in future. At the moment, most of these tools require specific file formats and metadata, which means the data is usually hosted elsewhere remotely e.g. in an S3 object store at their research institute. This often means that images are duplicated, and held in two locations. Ideally, it would be possible to upload to an image archive (with all the benefits of long-term storage, and unique identifiers), and still easily access such data in other tools.

With the recent rise of the OME-NGFF[88] project, this is becoming ever more possible. OME-NGFF is an attempt by the image analysis community to make a standard, openly available, chunked pyramidal file format. If this can see wide adoption, it would make it much easier to make various remote tools work together.

Chapter 6

Discussion

To summarise, my goal was to ease the integration of large-scale volume EM into multimodal atlases at various points within its lifecycle (Figure 1.1). At the acquisition stage, I developed two different methods to enhance the speed and precision of targeted EM acquisition. The first, a semi-automated ultramicrotome and workflow, allows precise targeting of a chosen plane within a sample. It also allows targeting in one step, unlike previous workflows, and allows user-friendly calculation of the required angles and distances.

The second option was an integrated software for targeting specific volumes of interest with SBEM. This integrates closely with the acquisition software SBEMimage allowing continuous comparison of EM data and corresponding X-ray scans.

Moving to the analysis stage, I created an example pipeline to quantify cell and nuclei morphologies from EM and compare to gene expression. This focused on the large *Platynereis* atlas, finding clear correspondence between gene expression and morphological tissue boundaries from the EM.

Finally, I added various features to the software MoBIE that allows remote browsing of massive multimodal atlases. This made creation and modification of MoBIE projects accessible to those with no programming experience.

While this work helps to integrate large EM at various steps, there is much future work to be done to make this a routine procedure. I have already noted specific future directions at the end of each chapter, so here I will consider the broader context within the field. What are the main bottlenecks that remain, and how can they be addressed?

Let's start with the acquisition step. As noted previously, for datasets that focus on specific regions of interest or cell types, the easiest way to collect large numbers of samples in EM is to be targeted in what you acquire. The main bottlenecks here remain the many manual steps that are required during this process. This is a general feature of sample preparation for EM, with many of the steps taking many days and requiring extensive training and expertise. More automation in sample preparation, and in the targeting steps would improve the throughput and ease of this step greatly. While the work in chapter 2 and 3 begins this automation, there is still a long way to go to make this truly user-friendly. Also, it would be useful to find techniques that work well for targeting from both light microscopy and X-ray data, as both can provide informative details for choosing a region of interest.

For questions that cannot rely on acquiring smaller regions, the solution must be faster acquisition of large EM volumes. There has been great progress on this in recent years, mostly driven by the connectomics community, that requires high-resolution imaging of very large EM volumes. Much of the improvements here are on the hardware side, for example, automatic sample exchangers[52] and custom FIB-SEM systems[22]. In recent years, there has also been the development of systems like the multi-beam SEM[90] that allow many electron beams to image the sample in parallel, leading to massively reduced imaging times.

Nevertheless, with all these methods, there are large downsides. Thinly sectioning a sample with an ultramicrotome is time consuming, difficult, and affected by problems with compression of sections as they are collected. Conversely, using methods like FIB-SEM and SBEM are easier, but they destroy the sample meaning no further analysis can be performed. As such, X-ray imaging is starting to become an interesting alternative, especially as its throughput and resolution continue to improve. X-ray imaging is non-destructive, and can be done with large pieces of sample without the need for serial sectioning. It is also much faster, being achievable in a few hours or days, depending on the resolution required. Both Bosch et al.[21] and Kuan et al.[19] demonstrate the use of high resolution synchrotron X-ray imaging for tracing neurons. Of course, X-ray cannot reach the same resolution as EM, so a hybrid approach is likely to become standard. X-ray can image large volumes and provide morphological context, where targeted EM can then provide high resolution detail where necessary.

Moving on to the integration step in the lifecycle, what are the bottlenecks for registration and integration of different modalities into an atlas? I think the most important future step is to increase the availability of open-source, user-friendly software for image registration. A number already exist e.g. BigWarp[46] and EC-CLEM[91], but these tend to focus on point-based image registration. It would be useful to develop more user-friendly intensity-based registration softwares, that require less human intervention (i.e. no manual placement of points). Elastix[47][48] is one option for this, but is not very accessible to those who cannot program - there is a command line interface, as well as wrappers in various programming languages like python. Some user-friendly wrappers exist for Fiji[92] and napari[93], but they could still be improved to simplify this process. In addition, many existing softwares cannot handle the extremely large images that EM produces, making them difficult to use. BigWarp is a clear exception to this, making use of chunked, pyramidal file formats to allow massive image data to be aligned. Even so, it would be useful if more software natively supported such formats to avoid re-writing existing data, and to increase their usability for EM.

Of course, intensity-based registration relies on having images with reasonably similar features visible (e.g. X-ray vs EM). For more disparate data (e.g. fluorescence microscopy vs EM), it is more difficult to find fully automatic solutions. We must find ways to make these two data types more similar. One option is to to use fluorescent markers that highlight similar features to EM - for example, it is possible to induce uranyl acetate (a common electron-dense stain for EM) to fluoresce brightly at cryogenic temperatures[94]. This is not always possible though depending on which structures you need to fluorescently label, and the type of EM you wish to do. Another option is to segment features from the EM, and use this for registration to fluorescence data. This is how the *Platynereis* atlas registration was completed -matching fluorescently stained nuclei to segmented EM nuclei with elastix[15]. Again though, this process is far from trivial, as EM segmentation is also challenging (as discussed below). Another option is to use deep learning approaches to predict fluorescent signal from EM images[95][96] to allow automatic registration. This also has its own challenges though, requiring large amounts of training data of paired fluorescence and EM images which are difficult to generate on a large scale. Finding fully automatic approaches, with minimal human intervention, is therefore still an area that needs much development.

Coming to the analysis step, the bottlenecks here really depend on the details of the atlas in question. Even so, a common goal for EM data is to automatically produce quantitative descriptions of cell morphologies. One of the big bottlenecks here is segmentation of large EM datasets. Machine learning and deep learning based approaches are improving rapidly, and more and more EM datasets can be segmented in an automatic or semi-automatic way. Still, these approaches require large amounts of training data which is often produced entirely manually by painting different pixels as belonging to different structures. There are a number of options for speeding up this process e.g. producing training data can be out-sourced to companies or crowd-sourced[57][58]. Also, a number of approaches can accept sparse training data where only small parts of an image are annotated, massively decreasing the amount of time required. Ilastik's workflows[53] are a good example of this. In addition, there are many pre-trained networks that are available through projects like the BioImage Model Zoo[97]. If your data are of a similar type, these can be applied directly, or used as a starting point for making a custom solution. Nevertheless, applying many of these options still takes a lot of expertise - we are still far away from fully automatic, and user-friendly solutions, especially for very large data.

Once a dataset is segmented, there are also many different options for quantifying cellular morphologies. Many approaches currently rely on a set of manually chosen features to tackle this problem e.g. volume, surface area and sphericity. This can lead to some bias in the interpretation though. Different sets of features will give different results, and it can be hard to know which features are useful to distinguish different cell types. It would be useful for more unbiased solutions to become available, in a somewhat user-friendly manner. For example, Hartmann et al.[98] use a point cloud based approach to describe morphological differences between cells in an unbiased way.

Another common goal, as spatial transcriptomics techniques become ever more popular, is to compare morphology from EM to gene expression. The main bottleneck here is with techniques to correlate large scale gene expression data to EM. One method is to match single cell RNA sequencing data to atlases of in-situ hybridisations of key genes. For example, this is the next goal of the *Platynereis* atlas discussed in chapter 4. This is only possible for organisms that have a very stereotypical development though, allowing matching of EM and gene expression from many individual samples. It would be very interesting to see if existing spatial transcriptomics methods (such as multiplexed single molecule FISH approaches like MERFISH[67] and seqFISH[68]) could somehow be made compatible with EM sample preparation methods. Being able to correlate thousands of genes with high resolution EM in the same sample would allow extremely high quality comparisons to be made.

We would also need better approaches for comparing large scale morphology and gene expression data. Here, techniques from single cell 'multi-omics' could be very useful, where there are already methods for comparing large datasets from a number of disparate sources. Regardless, for all these analysis steps, we still need to make these more accessible to individuals with no programming knowledge. For example, providing user-friendly wrappers for morphological analysis in software like napari or Fiji.

Finally, we come to the resource sharing stage - what are the main bottlenecks here? A lot of the problems here stem from the very large size of atlases containing EM data (often many terabytes in size). This means there is an ever increasing need for reliable, and efficient remote access to these datasets. There have been great improvements over recent years in this area, with a number of tools now available that support this. In future, it would be useful to see greater interoperability between these different options, and also image archives themselves. This will be greatly enhanced by standardising an open-source chunked, pyramidal file format, as the OME-NGFF[88] project aims to do. If images can be shared and analysed in the same format across many tools, then remote image analysis becomes much easier. Another shift that is starting now, but still has a long way to go, is to allow full image analysis tasks from remote data. At the moment, most tools focus on allowing the images to be browsed, and viewed conveniently, but only offer limited options for actual analysis. Again, this will improve as these softwares and formats become more standardised and widely accepted.

So, there are many points for future work to make large-scale integration of EM data into multimodal atlases standard practice. Perhaps a good point to end on, is that we must always be careful to not lose sight of the final goal when creating an atlas. Maps exist to integrate data in a spatial way, and allow connections to be seen that would otherwise remain hidden. A map where every inch is covered by lines and symbols is useless; they must always be a curated simplification of the real system. I'm reminded of a comparison that was made in one of the Human Cell Atlas papers[10] to a one paragraph story by Jorge Luis Borges termed 'On Exactitude in Science'. Here, an empire obsessed with cartography makes ever more detailed maps before finally creating the perfect map, of the same 1:1 size as the empire itself. Of course, such a map becomes useless, as it is just as complex as the real system. We must bear the same concepts in mind when we create biological atlases - what level of detail is necessary, and useful, when balanced against the cost? Which data will be most useful, to the most researchers?

Biological atlases have the potential to be invaluable resources for the biological community, and should therefore be made in close collaboration with the researchers who will benefit from it. After all, a map with no-one to read it is also useless, no matter how beautiful, or impressive its construction is.

Chapter 7

Methods

7.1 Acquisition: faster targeting with an ultramicrotome

7.1.1 Semi-automated microtome

The semi-automated microtome system was designed and made by: Alfons Riedinger, Vera Stankova, Helmuth Schaar and Arthur Milberger (of the electronic and mechanical workshops at EMBL), with guidance from Yannick Schwab and myself.

It was designed as an add-on to a standard Leica UC7 Ultramicrotome, with only one small modification to the ultramicrotome itself. A hole was drilled into the cutting arm at a precisely vertical position (see Figure 2.3C). This fits a small pin that was added to the back of the arc piece (Figure 2.3D) to keep the arc piece precisely vertical.

A brand new sample holder/arc and knife holder were purchased from Leica, and modified with motors for each axis. These motors are controlled by a small Raspberry Pi computer contained in the box to the right of the microtome. This has a small touch-screen interface that allows the motor positions to be controlled and monitored. The user interface is made with Node-RED (`https://nodered.org/`)

7.1.2 Crosshair

Crosshair is a Fiji[37] plugin written in Java, and is freely available from the github repository: `https://github.com/K-Meech/crosshair`. It can also be easily installed via a Fiji update site (instructions in the github repository).

It builds on top of BigDataViewer[38] for viewing images in 2D, and the ImageJ 3D Viewer[39] for viewing images in 3D. It also builds on a number of functions from Christian Tischer's imagej-utils repository (`https://github.com/embl-cba/imagej-utils`).

7.1.3 Platynereis sample preparation

All *Platynereis* sample preparation and resin-embedding was done by Rachel Templin (former member of the Schwab lab, now at Monash University in Australia).

Platynereis were prepared with an adaptation of the rOTO protocol aided by microwave processing (see details in table 7.1 below). *Platynereis* were flat embedded in thin pieces of resin.

7.1.4 Accuracy tests

For the accuracy tests, samples were prepared by Rachel Templin as specified in section 7.1.3. Samples were then trimmed and X-ray imaged by Rosa Pipitone (former member of the Schwab lab/EMCF) as follows: samples were cut out in small pieces of resin with a razor blade. These pieces were then glued to larger resin blocks to make them easier to handle in the ultramicrotome. Samples were trimmed close to the *Platynereis* with a diamond knife, so that the block surface was less than one feed length (200 microns) from the target. A flat surface was trimmed with a rectangular shape as required for Crosshair. These blocks were then X-ray imaged with a Bruker SkyScan 1272, with an isotropic voxel size of 1 micron.

I then processed these X-ray images with Crosshair. I flipped the X-ray stacks vertically in Fiji to ensure they weren't mirrored with respect to the actual sample. I also cropped them to remove empty areas in the X-ray scan outside the resin block. Then, I set the block face plane in Crosshair by fitting to a series of points placed on the surface. The target plane was set manually to cut through both anterior dorsal cirrus that protrude form either side of the *Platynereis* head. I then completed the remaining targeting steps of the workflow as laid out in the protocol in appendix B.

After I targeted the 5 samples, they were X-rayed again with the Bruker SkyScan 1272 at 1 micron resolution by Inés Romero Brey (member of the Schwab lab).

I then proceeded to calculate various accuracy measures. I first registered the before and after X-ray scans using my RegistrationTree plugin (see method's section 7.2.2). For this, I registered the before scan (moving image) to the after scan (fixed image). I used a rough point-based registration with BigWarp[46] and manually placed points (limited to rotation and translation only). I then used an intensity-based registration from elastix[47][48] (again limited to rotation and translation only). Once this was complete, I exported it from RegistrationTree as a BigDataViewer[38] xml file containing the required transform in the 'MOVING' space i.e. so that it transformed the after scan onto the before scan.

To calculate the measures, I used the 'Measure Targeting Accuracy' functions included within the Crosshair plugin. This takes the before X-ray scan, the registered after X-ray scan and the Crosshair settings and solution files and allows easy calculation of accuracy measures. I fitted a plane to the after scan's block surface by manually placing a series of points on its surface (same workflow as fitting the planes in usual Crosshair

Step No.	Step	Time	Temperature	Microwave Watt	Vacuum
1	2% paraformaldehyde, 2.5% glutaraldehyde in seawater	2 minutes	Room Temp	100	On
2	2% paraformaldehyde, 2.5% glutaraldehyde in 0.1M cacodylate	2x 14 minutes (2 min on/off cycles)	Room Temp	100	On
3	0.1M cacodylate	1 immediate. Then 2x40 seconds.	Room Temp	100	On
4	2% OsO4 in 0.1M cacodylate	2x 14 minutes (2 min on/off cycles)	Room Temp	100	On
5	2.5% $K_4[Fe(CN)_6] \cdot 3H_2O$ in 0.1M cacodylate	2x 14 minutes (2 min on/off cycles)	Room Temp	100	On
6	Water wash	1 immediate. Then 2x40 seconds.	Room Temp	100	On
7	1% TCH unbuffered	2x 14 minutes (2 min on/off cycles)	40 degrees C	100	On
8	Water wash	1 immediate. Then 2x40 seconds.	40 degrees C	100	On
9	2% OsO4 aqueous	2x 14 minutes (2 min on/off cycles)	Room Temp	100	On
10	Water wash	1 immediate. Then 2x40 seconds.	Room Temp	100	On
11	Dehydration series in ethanol (25%, 50%, 75%, 3x100%)	40 seconds each	10 degrees C	250	Off
12	Infiltration series in Durcupan (25%, 50%, 75%, 90%, 3x100%)	3 minutes each.	Room Temp	150	On
13	100% Durcupan	overnight	Room Temp	N/A	N/A
14	Polymerisation in oven	48 hours	60 degrees C	N/A	N/A

Table 7.1: Table of *Platynereis* preparation steps

usage). I then used the ‘Save Measures’ option to save a simple json file containing the measures.

This includes two measures - one for the angle error, and one for the distance error. The angle error is calculated as the absolute angle between the normals of the target plane, and the after scan surface. The distance error is more complex to calculate, as any planes that are not exactly parallel will intersect at some location (i.e. distance = 0). I instead try and calculate the actual distance cut (given the final position of the surface), and the difference between this and the solution’s distance. It is therefore a measure of how accurately you managed to cut the solution’s distance, rather than a direct measure of how close you are to the target plane. This is calculated by determining the distance perpendicular to the after scan surface to the predicted first touch point on the original block surface. Note that I assume that cutting started from the predicted point - if the angle was so far off that cutting started from another point this measure would be erroneous. This is something that is checked at the start of each run though, by looking down the microtome binocular, and determining that the correct vertex is touched first.

Once the perpendicular distance is known, this is converted to the NS cutting distance of the microtome by:

$$D_{NS} = \frac{D_P}{cos(\theta_K)}$$

where D_{NS} is the NS distance, D_P is the perpendicular distance between the after surface plane and the first touch point, and θ_K is the knife angle from the chosen solution.

The final distance error is then:

$$Error = D_{NS} - D_{Sol}$$

where D_{NS} is the NS distance, and D_{Sol} is the distance from the chosen solution. A positive error indicates that you cut too far, while a negative error indicates you didn’t cut enough.

Note this assumes that the knife angle used was the same as set in the solution. If there is any error in the knife angle setting, then this will also affect this accuracy measure.

See here for the exact calculation scripts: `https://github.com/K-Meech/crosshair/blob/master/src/main/java/de/embl/schwab/crosshair/targetingaccuracy/AccuracyCalculator.java`

7.2 Acquisition: faster targeting with SBEM

7.2.1 SBEMimage

SBEMimage[44] is an open-source program developed by Benjamin Titze and others (`https://github.com/SBEMimage/SBEMimage`). The n5 conversion I added uses Constantin Pape's pyBDV python package (`https://github.com/constantinpape/pybdv`).

7.2.2 RegistrationTree

RegistrationTree is available open-source on github - `https://github.com/K-Meech/RegistrationTree` - along with a short user-guide. It can easily be installed via a Fiji[37] update site. RegistrationTree uses BigDataViewer[38] for 2D visualisation, and integrates BigWarp[46] and elastix[47][48] for registration. It builds on top of Christian Tischer's elastixWrapper[92](`https://github.com/embl-cba/elastixWrapper`) and imagej-utils repository (`https://github.com/embl-cba/imagej-utils`). Also, the open-source repositories for image-transform-converters (`https://github.com/image-transform-converters/image-transform-converters`) and bigdataviewer-playground (`https://github.com/bigdataviewer/bigdataviewer-playground`).

7.2.3 SBEMViewer

SBEMViewer is a fiji[37] plugin, that also uses BigDataViewer[38] for 2D visualisation. It integrates directly with RegistrationTree for registration. It also uses functions from MoBIE[15] (`https://mobie.github.io/`) and Christian Tischer's imagej-utils repository (`https://github.com/embl-cba/imagej-utils`).

7.2.4 Platynereis sample preparation

All samples were prepared by Rachel Templin, using the same protocol as in section 7.1.3.

7.2.5 Synchrotron X-ray acquisition

Samples were mounted on pins by Rachel Templin as follows: samples were cut out in small pieces of resin with a razor blade. These small pieces were glued to SPINE sample holders[99] which were originally developed for macromolecular crystallography. This holder was then inserted into a plastic vial which was then mounted into a protecting metal sample basket for shipment to the synchrotron in Hamburg.

All X-ray imaging was done in collaboration with Thomas Schneider's group at EMBL Hamburg. Imaging and reconstruction were done by Maxim Polikarpov and Gleb Bourenkov as follows: samples were imaged on the EMBL beamline P14 on the PETRA III storage ring (c/o DESY, Hamburg, Germany) using the propagation-based phase-contrast imaging setup described in Polikarpov et al.[100] at an X-ray energy of 18 keV. X-ray images were recorded using an Optique Peter (Lyon, France) X-ray microscope consisting of an LSO:Tb scintillator with $8\mu m$ active layer; an Olympus UPlanFL 20-fold objective (Olympus, Tokio, Japan), numerical aperture 0.5; a 45° mirror; a 180 mm tube lens and a PCO.edge 4.2 sCMOS camera with 2048x2048 pixels, pixels ($6.5\mu m$ pixel size). Therefore, the effective pixel size was $0.325\mu m$ with a field of view 666 x

$666\mu m^2$. This setup typically delivers a resolution of about 0.5-0.7μm, as determined from the analysis of projection images from a Siemens star (Ta on SiN; XRESO-50HC, NTT-AT, Japan).

For these resin embedded *Platynereis* samples, projection images were acquired at four camera distances: 136, 141, 146 and 151 mm as selected according to Zabler et al.[101]. At each distance, 30 flat-field images and 3600 projections covering 360° of continuous rotation were recorded with an exposure time of 10 ms per frame. Data collection (including robotic sample transfer from a storage vessel to the rotation axis and sample centring via an on-axis optical microscope) was completed in 5 minutes.

Flat-field corrections were applied by dividing each projection image by the most similar flat-field image according to the SSIM criterion[102]. For lateral shift compensation at the four camera positions, images recorded at each projection angle were registered using Fourier-space correlation with a sub-pixel interpolation. Registered images were further processed by a multi-distance non-iterative holographic reconstruction[101][103], using a complex refraction index decrement ratio $\beta/\delta = 0.15$ and a zero compensation of 0.001. Tomographic reconstructions were performed using the TOMOPY package[104], employing the built-in Gridrec algorithm and Shepp-Logan filtering with default settings. All steps of the XIMG data processing were combined into a python-based custom software pipeline, available at: `https://pypi.org/project/maximus48/`.

7.2.6 EM acquisition

After returning from the synchrotron, samples were cut off of the SPINE sample holders and glued to SBEM pins by Rachel Templin. They were then surrounded by silver epoxy resin.

All EM was acquired with a Zeiss GeminiSEM 450 with a Gatan 3View 2XP. Zeiss SmartSEM and Gatan Digital Micrograph (GMS3) controlled the SEM and the microtome respectively, with SBEMimage running on top.

For the example run, I acquired 100 slices at 200nm thickness with overview images in SBEMimage and converted them on the fly to n5. I monitored this continuously with SBEMViewer, before registration with RegistrationTree. I registered the X-ray (moving image) to the EM (fixed image). The registration consisted of flipping the stack in z, followed by a rough point-based alignment with BigWarp (limited to rotation and translation only). Then, I cropped both images tightly to the *Platynereis* and registered with Elastix (also limited to rotation and translation only).

This registration was then loaded back into SBEMViewer so the EM and X-ray could be compared. A target point at position (53.973, -153.966, 76.548) in (x, y, z) micron coordinates aligned to the SEM coordinate space of SBEMimage was set. A further 282 slices were cut at 200nm thickness making a total of 382 slices and a z depth of 76.4 microns. After this point, the slice thickness was reduced to 40nm and a further 72 slices were acquired to a total depth of 79.28 microns.

All overviews were acquired with a pixel size of 100nm, at 1.5kV and 300pA current.

7.3 Analysis: quantitative morphology from EM

All methods are available in detail from the published paper for the *Platynereis* atlas[15]. Here, I summarise the methods for the parts I directly contributed to. All results use version 1.0.1 of the *Platynereis* atlas data unless noted otherwise. All scripts and data are available in the github respository: `https://github.com/mobie/platybrowser-project`

7.3.1 Chromatin segmentation

I segmented chromatin into dark (heterochromatin + nucleolus) and light (euchromatin) regions using ilastik's[53] pixel classification workflow. I manually labelled the nuclei of 50 cells with diverse nuclear morphologies into these two classes, on images downsampled to 20 x 20 x 25 nm^3 voxel size. Constantin Pape's nuclei segmentation was used as a mask, to limit the segmentation to only nuclear regions.

7.3.2 Morphology clustering

Using the cell, nucleus and chromatin segmentations, I calculated 140 different features to describe their morphology, intensity and texture (see table C.1 in appendix C for details of features). These were calculated on images downsampled to 80 x 80 x 100 nm^3 resolution. Cells (and their associated nuclei and chromatin) were removed that had no assigned nucleus, were in certain regions (yolk, neuropil, cuticle, cavities), or outside a reasonable size range, as these were most likely to be affected by segmentation errors. This resulted in 11348 remaining cells.

Cells (and their associated nuclei and chromatin) were further filtered to remove those within the region for extrapolated intensity correction (see methods of Vergara et al.[15] for details). This was to avoid any possible artefacts in features that rely on the raw intensity data. This left a total of 10344 cells.

The final table was then 140 features by 10344 cells. At this stage, all features were standardised by centring to a mean of 0 and scaling to unit variance. A K-nearest neighbour graph was then constructed (k = 10) using Euclidean distance between the feature vectors. Finally, community detection was used (with the Louvain method[71], and a resolution parameter of 1.2), to give 11 final clusters. These clusters were then displayed on a UMAP[72] (Uniform Manifold Approximation and Projection for Dimension Reduction) with parameters: n_neighbors = 10 and min_dist = 0.1.

The morphological features script is available here: `https://github.com/mobie/platybrowser-project/blob/main/mmpb/extension/attributes/morphology_impl.py` and the Snakemake workflow here: `https://github.com/mobie/platybrowser-project/tree/main/analysis/morphology_clustering`. The final clustering is here: `https://github.com/mobie/platybrowser-project/blob/main/data/1.0.1/tables/`

sbem-6dpf-1-whole-segmented-cells/morphology_clusters.tsv and the final UMAP here: https://github.com/mobie/platybrowser-project/blob/main/data/1.0.1/tables/sbem-6dpf-1-whole-segmented-cells/morphology_umap.tsv.

All morphological clustering analysis used python and the following packages: scikit-image[105], vigra(http://ukoethe.github.io/vigra/), scipy[106], mahotas[107], scikit-learn[108], networkx[109], python-louvain(https://github.com/taynaud/python-louvain), umap-learn[72], pandas[110], numpy[111] and snakemake[73].

7.3.3 Cell type annotation

All of the 11,402 segmented cells were split into mini-blocks of 10 consecutive cells. These were then randomly shuffled and merged together into larger blocks of 100 cells each, to ensure uniform sampling across the *Platynereis*. 10 of these final 115 blocks were then manually annotated by Valentyna Zinchenko, Rachel Templin and myself. Each cell was assigned to one of seven cell categories (neurons, dark cells, epithelial, muscle, digestive midgut cells, secretory and ciliated cells) or as 'unsure' or 'error'. This resulted in annotations of 65% neurons, 16% epithelial, 11% muscle, 3.5% dark, 2.38% digestive midgut, 1.80% secretory and 0.35% ciliated. Finally, more cells were manually annotated to target the lower percentage cell classes of ciliated, secretory, midgut and dark cells. This gave final numbers of annotated cells as: 571 neuron, 141 epithelial, 53 muscle, 41 dark, 25 ciliated, 51 secretory and 55 digestive midgut cells. The annotation results are available here: https://github.com/mobie/platybrowser-project/blob/main/data/1.0.1/tables/sbem-6dpf-1-whole-segmented-cells/default.tsv (column 'cell_type').

7.3.4 Bilateral pair analysis

I first needed a way to determine if one cell was a potential bilateral partner of another (i.e. if they were symmetric about the midline of the *Platynereis*). To do this, I established a set of criteria based on the position of the cells' nuclei. As the *Platynereis* is bent in the EM volume, I cannot directly use the xyz position of each nucleus to determine if they are potential partners. Instead, I first calculated a midline surface for the *Platynereis* by fitting a second order polynomial of two variables to a set of manually chosen points that lie on the midline. These points were chosen by David Puga (a former member of the Arendt lab at EMBL). I then calculated the absolute distance from this surface for all nuclei.

While this can be used to determine if cells are a similar distance from the midline, it does not account for them having different positions along the anterior-posterior (AP) or dorsal-ventral (DV) axis of the *Platynereis*. To tackle this, I transformed the xyz position of all nuclei from EM space back to the original ProSPr space of the gene expression atlas. This was done with Elastix[47][48] with the same parameters as used for the main registration (see Vergara et al.[15] for details). In ProSPr the AP and DV axes are nicely aligned to y and z, so we can use these coordinates to estimate if cells have a similar position. The final criteria are then: similar absolute distance from the midline, on opposite sides of the midline, similar y position in ProSPr space and similar z position in ProSPr space.

The range to consider 'similar' was calculated from a set of manually chosen bilateral pairs (202 in total). The acceptable difference was set to the mean over these pairs plus two standard deviations.

Given this criteria to determine if two cells are potential bilateral partners, I then worked to calculate how far must be travelled in morphology space from each cell to its first potential partner. This was calculated by taking different subsets of the morphology features (see Morphology clustering methods above) and standardising them by centring each feature's mean to 0 and scaling to unit variance. Then, for all cells a ranking was formed from very closest neighbour, to most distant neighbour based on Euclidean distance in morphology space. I then found the closest neighbour in this ranking that met the bilateral criteria described above.

To compare to random assignment of bilateral pairs, the ranking of cells was randomly shuffled, and the analysis repeated. This was done 100 times, and the mean of all trials taken for the final randomised results figure 4.5B.

The analysis scripts are available here: `https://github.com/mobie/platybrowser-project/tree/main/analysis/bilateral_pairs` and here: `https://github.com/mobie/platybrowser-project/tree/main/analysis/midline`. The table of manually curated pairs is available here: `https://github.com/mobie/platybrowser-project/blob/main/data/1.0.1/tables/sbem-6dpf-1-whole-segmented-cells/symmetric_cells.tsv`. The midline fit and final graphs were calculated in R, making use of the tidyverse[112] and rgl (`https://cran.r-project.org/web/packages/rgl/index.html`) packages. All other analysis was in Python with pandas[110], numpy[111], scikit-learn[108] and scipy[106].

7.3.5 Gene expression clustering

Each cell in the *Platynereis* atlas has a gene expression value for 201 genes determined by 'overlap assignment' i.e. by the fraction of the volume of the cell overlapping with the registered gene volume. This gives each cell a vector of length 201 with values ranging from 0 (no overlap) to 1 (complete overlap).

I filtered all segmented cells to remove those most likely to be affected by segmentation errors. Cells were removed that had no assigned nucleus, were in certain regions (yolk, neuropil, cuticle, cavities), or outside a reasonable size range. Also, any cells that expressed no genes were removed. This gave 11,366 final cells.

In a very similar manner to the morphological clustering above, a K-nearest neighbour graph was contructed (k=20) using the Euclidean distance between the gene overlap vectors. Community detection was used (with the Louvain method[71] and a resolution parameter of 1.2) to give 15 final clusters. Clusters were visualised with UMAP[72] with parameters: n_neighbors=20 and min_dist=0.1.

This section used version 1.0.0 of the data. The snakemake workflow is here: `https://github.com/mobie/platybrowser-project/tree/main/analysis/gene_clustering`. The final clustering and UMAP are available here: `https://github.com/mobie/platybrowser-project/blob/main/data/0.6.5/tables/sbem-6dpf-1-whole-segmented-cells/gene_clusters.tsv` and here: `https://github.com/mobie/platybrowser-project/blob/main/data/1.0.0/tables/sbem-6dpf-1-whole-segmented-cells/gene_umap.tsv`.

All gene expression clustering analysis was done with Python and the following packages: scikit-learn[108], networkx[109], python-louvain (`https://github.com/taynaud/python-louvain`), umap-learn[72], pandas[110], numpy[111] and snakemake[73].

7.3.6 Gene specificity analysis

A specificity score was calculated for every gene cluster and individual gene for every ganglionic nuclei (GN). This score is a combination of two measures: first, the fraction of expression confined to a given ganglion (A) and second, the fraction of that ganglion covered (B). In analogy with the F1 score, specificity is calculated as: $\frac{2AB}{A+B}$

A threshold of 0.5 gene overlap was used to label a cell as expressing a particular gene for this analysis. The specificity calculations are part of the same workflow as described in the gene expression clustering section: `https://github.com/mobie/platybrowser-project/tree/main/analysis/gene_clustering`, specific script here: `https://github.com/mobie/platybrowser-project/blob/main/analysis/gene_clustering/scripts/ganglia_specificity.py`, and also used version 1.0.0 of the data.

7.4 Resource Sharing: managing massive, multimodal data

7.4.1 MoBIE

MoBIE is a Fiji plugin that allows browsing of large image data stored remotely or locally. It was created/maintained jointly by Christian Tischer, Constantin Pape and myself. All the code is available in the github repository: `https://github.com/mobie/mobie-viewer-fiji`. It was originally developed for the *Platynereis* atlas - see Vergara et al.[15] for details.

N5 conversion in the project creator builds on the existing BigDataViewer plugin for n5 export (`https://github.com/bigdataviewer/bigdataviewer_fiji`). OME-ZARR (OME-NGFF) export builds on the Saalfeld lab's Zarr exporters (`https://github.com/saalfeldlab/n5-zarr`). All these conversion functions are made available in the following repository: `https://github.com/mobie/mobie-io`.

References

[1] Nestor Milyaev et al. "The virtual fly brain browser and query interface". In: *Bioinformatics* 28.3 (2012), pp. 411–415. ISSN: 13674803. DOI: 10.1093/bioinformatics/btr677.

[2] *Virtual Fly Brain Website*. URL: https://v2.virtualflybrain.org/.

[3] Arnim Jenett et al. "A GAL4-Driver Line Resource for Drosophila Neurobiology". In: *Cell Reports* 2.4 (2012), pp. 991–1001. ISSN: 22111247. DOI: 10.1016/j.celrep.2012.09.011.

[4] C Shan Xu et al. "A Connectome of the Adult Drosophila Central Brain". In: *bioRxiv* (Jan. 2020), p. 2020.01.21.911859. DOI: 10.1101/2020.01.21.911859.

[5] Zhihao Zheng et al. "Structured sampling of olfactory input by the fly mushroom body". In: *bioRxiv* (Jan. 2020), p. 2020.04.17.047167. DOI: 10.1101/2020.04.17.047167.

[6] *Allen Brain Atlas Website*. URL: https://portal.brain-map.org/.

[7] Ed S. Lein et al. "Genome-wide atlas of gene expression in the adult mouse brain". In: *Nature* 445.7124 (2007), pp. 168–176. ISSN: 00280836. DOI: 10.1038/nature05453.

[8] Yin Cai et al. "Experimental and computational framework for a dynamic protein atlas of human cell division". In: *Nature* 561.7723 (Sept. 2018), pp. 411–415. ISSN: 0028-0836. DOI: 10.1038/s41586-018-0518-z.

[9] *Mitocheck Website*. URL: https://www.mitocheck.org/mitotic_cell_atlas.

[10] Aviv Regev et al. "Science Forum: The Human Cell Atlas". In: *eLife* (2017), pp. 1–30. ISSN: 2050-084X.

[11] Jennifer E. Rood et al. "Toward a Common Coordinate Framework for the Human Body". In: *Cell* 179.7 (2019), pp. 1455–1467. ISSN: 10974172. DOI: 10.1016/j.cell.2019.11.019.

[12] Christopher J. Peddie and Lucy M. Collinson. "Exploring the third dimension: Volume electron microscopy comes of age". In: *Micron* 61 (2014), pp. 9–19. ISSN: 09684328. DOI: 10.1016/j.micron.2014.01.009.

[13] Kevin L Briggman and Davi D Bock. "Volume electron microscopy for neuronal circuit reconstruction". In: *Current Opinion in Neurobiology* 22.1 (Feb. 2012), pp. 154–161. ISSN: 0959-4388. DOI: 10.1016/J.CONB.2011.10.022.

[14] *EMofCellsTissuesOrganisms Website*. URL: https://twitter.com/EMofCTO.

[15] Hernando M. Vergara et al. "Whole-body integration of gene expression and single-cell morphology". In: *Cell* 184.18 (2021), 4819–4837.e22. ISSN: 00928674. DOI: 10.1016/j.cell.2021.07.017.

[16] Philip J Withers et al. "X-ray computed tomography". In: *Nature Reviews Methods Primers* (). ISSN: 2662-84492662-8449. DOI: 10.1038/s43586-021-00015-4.

[17] Shelley D. Rawson et al. "X-ray computed tomography in life sciences". In: *BMC Biology* 18.1 (2020), pp. 1–15. ISSN: 17417007. DOI: 10.1186/s12915-020-0753-2.

[18] Matthia A Karreman et al. "Find your way with X-Ray: using microCT to correlate in vivo imaging with 3D electron microscopy". In: (2017). DOI: 10.1016/bs.mcb.2017.03.006.

[19] Aaron T. Kuan et al. "Dense neuronal reconstruction through X-ray holographic nano-tomography". In: *Nature Neuroscience* 23.12 (2020), pp. 1637–1643. ISSN: 15461726. DOI: 10.1038/s41593-020-0704-9.

[20] Eric A. Bushong et al. "X-Ray Microscopy as an Approach to Increasing Accuracy and Efficiency of Serial Block-Face Imaging for Correlated Light and Electron Microscopy of Biological Specimens". In: *Microscopy and Microanalysis* 21.1 (2015), pp. 231–238. ISSN: 14358115. DOI: 10.1017/S1431927614013579.

[21] Carles Bosch et al. "Functional and multiscale 3D structural investigation of brain tissue through correlative in vivo physiology, synchrotron micro-tomography and volume electron microscopy". In: *bioRxiv* (Jan. 2021), p. 2021.01.13.426503. DOI: 10.1101/2021.01.13.426503.

[22] Louis K Scheffer et al. "A connectome and analysis of the adult Drosophila central brain". In: *eLife* 9 (Sept. 2020). Ed. by Eve Marder et al., e57443. ISSN: 2050-084X. DOI: 10.7554/eLife.57443.

[23] Matthia A Karreman et al. "Fast and precise targeting of single tumor cells in vivo by multimodal correlative microscopy." In: *Journal of cell science* 129.2 (Jan. 2016), pp. 444–56. ISSN: 1477-9137. DOI: 10.1242/jcs.181842.

[24] C. Shan Xu et al. "An open-access volume electron microscopy atlas of whole cells and tissues". In: *Nature* November 2020 (2021). ISSN: 0028-0836. DOI: 10.1038/s41586-021-03992-4.

[25] Jacob M Musser et al. "Profiling cellular diversity in sponges informs animal cell type and nervous system evolution". In: *Science* 374.6568 (2021), pp. 717–723. DOI: 10.1126/science.abj2949.

[26] Paolo Ronchi et al. "High-precision targeting workflow for volume electron microscopy". In: *The Journal of cell biology* 220.9 (2021). ISSN: 15408140. DOI: 10.1083/jcb.202104069.

[27] Elisabeth Brama et al. "ultraLM and miniLM : Locator tools for smart tracking of fluorescent cells in correlative light and electron microscopy [version 1 ; peer review : 3 approved , 1 approved with reservations]". In: *Wellcome Open Res* (2016), pp. 1–26.

[28] Valentina Baena et al. *Serial-section electron microscopy using automated tape-collecting ultramicrotome (ATUM).* 1st ed. Vol. 152. Elsevier Inc., 2019, pp. 41–67. ISBN: 9780128170182. DOI: 10.1016/bs.mcb.2019.04.004.

[29] Thomas Templier. "MagC, magnetic collection of ultrathin sections for volumetric correlative light and electron microscopy". In: *eLife* 8 (2019), pp. 1–18. DOI: 10.7554/elife.45696.

[30] Timothy J. Lee et al. "Large-scale neuroanatomy using LASSO: Loop-based Automated Serial Sectioning Operation". In: *PLOS ONE* 13.10 (Oct. 2018). Ed. by Konradin Metze, e0206172. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0206172.

[31] B. Duygu Özpolat et al. "The Nereid on the rise: Platynereis as a model system". In: *EvoDevo* 12.1 (2021), pp. 1–22. ISSN: 2041-9139. DOI: 10.1186/s13227-021-00180-3.

[32] Albrecht Fischer and Adriaan Dorresteijn. "The polychaete Platynereis dumerilii (Annelida): A laboratory animal with spiralian cleavage, lifelong segment proliferation and a mixed benthic/pelagic life cycle". In: *BioEssays* 26.3 (2004), pp. 314–325. ISSN: 02659247. DOI: 10.1002/bies.10409.

[33] Elizabeth A Williams and Gáspár Jékely. "Towards a systems-level understanding of development in the marine annelid Platynereis dumerilii". In: *Current Opinion in Genetics & Development* 39 (Aug. 2016), pp. 175–181. ISSN: 0959-437X. DOI: 10.1016/J.GDE.2016.07.005.

[34] Pavel Vopalensky et al. "From spiral cleavage to bilateral symmetry: The developmental cell lineage of the annelid brain". In: *BMC Biology* 17.1 (2019), pp. 1–19. ISSN: 17417007. DOI: 10.1186/s12915-019-0705-x.

[35] Nadine Randel et al. "Inter-individual stereotypy of the Platynereis larval visual connectome". In: *eLife* 4 (June 2015), e08069. ISSN: 2050-084X. DOI: 10.7554/eLife.08069.

[36] Hernando Martínez Vergara et al. "Whole-organism cellular gene-expression atlas reveals conserved cell types in the ventral nerve cord of Platynereis dumerilii." In: *Proceedings of the National Academy of Sciences of the United States of America* 114.23 (June 2017), pp. 5878–5885. ISSN: 1091-6490. DOI: 10.1073/pnas.1610602114.

[37] Johannes Schindelin et al. "Fiji: An open-source platform for biological-image analysis". In: *Nature Methods* 9.7 (2012), pp. 676–682. ISSN: 15487091. DOI: 10.1038/nmeth.2019.

[38] Tobias Pietzsch et al. "BigDataViewer: Visualization and processing for large image data sets". In: *Nature Methods* 12.6 (2015), pp. 481–483. ISSN: 15487105. DOI: 10.1038/nmeth.3392.

[39] Benjamin Schmid et al. "A high-level 3D visualization API for Java and ImageJ". In: *BMC Bioinformatics* 11.1 (2010), p. 274. ISSN: 1471-2105. DOI: 10.1186/1471-2105-11-274.

[40] Benjamin Titze and Christel Genoud. “Volume scanning electron microscopy for imaging biological ultrastructure”. In: *Biology of the Cell* 108.11 (2016), pp. 307–323. ISSN: 1768322X. DOI: 10.1111/boc.201600024.

[41] Winfried Denk and Heinz Horstmann. “Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure”. In: *PLoS Biology* 2.11 (2004). ISSN: 15449173. DOI: 10.1371/journal.pbio.0020329.

[42] Philipp Hanslovsky, John A. Bogovic, and Stephan Saalfeld. “Image-based correction of continuous and discontinuous non-planar axial distortion in Serial section microscopy”. In: *Bioinformatics* 33.9 (2017), pp. 1379–1386. ISSN: 14602059. DOI: 10.1093/bioinformatics/btw794. arXiv: 1511.01161.

[43] K. M. Boergens and W. Denk. “Controlling FIB-SBEM slice thickness by monitoring the transmitted ion beam”. In: *Journal of Microscopy* 252.3 (2013), pp. 258–262. ISSN: 00222720. DOI: 10.1111/jmi.12086.

[44] Benjamin Titze, Christel Genoud, and Rainer W. Friedrich. “SBEMimage: Versatile Acquisition Control Software for Serial Block-Face Electron Microscopy”. In: *Frontiers in Neural Circuits* 12.July (2018), pp. 1–9. ISSN: 16625110. DOI: 10.3389/fncir.2018.00054.

[45] *N5 File Format*. URL: https://github.com/saalfeldlab/n5.

[46] John A Bogovic et al. “Robust registration of calcium images by learned contrast synthesis”. In: *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*. 2016, pp. 1123–1126. DOI: 10.1109/ISBI.2016.7493463.

[47] Stefan Klein et al. “elastix: A Toolbox for Intensity-Based Medical Image Registration”. In: *IEEE Transactions on Medical Imaging* 29.1 (2010), pp. 196–205. DOI: 10.1109/TMI.2009.2035616.

[48] Denis P. Shamonin et al. “Fast parallel image registration on CPU and GPU for diagnostic classification of Alzheimer’s disease”. In: *Frontiers in Neuroinformatics* 7.JAN (2014), pp. 1–15. ISSN: 16625196. DOI: 10.3389/fninf.2013.00050.

[49] Nicholas Sofroniew et al. *napari/napari: 0.4.12rc2*. Oct. 2021. DOI: 10.5281/zenodo.5587893. URL: https://doi.org/10.5281/zenodo.5587893.

[50] John M. Lucocq et al. “Systems biology in 3D space - enter the morphome”. In: *Trends in Cell Biology* 25.2 (2015), pp. 59–64. ISSN: 18793088. DOI: 10.1016/j.tcb.2014.09.008.

[51] John Lucocq. “Unbiased 3-D quantitation of ultrastructure in cell biology”. In: *Trends in Cell Biology* 3.10 (1993), pp. 354–358. ISSN: 09628924. DOI: 10.1016/0962-8924(93)90106-B.

[52] Zhihao Zheng et al. “A Complete Electron Microscopy Volume of the Brain of Adult Drosophila melanogaster.” In: *Cell* 0.0 (July 2018). ISSN: 1097-4172. DOI: 10.1016/j.cell.2018.06.019.

[53] Stuart Berg et al. “Ilastik: Interactive Machine Learning for (Bio)Image Analysis”. In: *Nature Methods* 16.12 (2019), pp. 1226–1232. ISSN: 15487105. DOI: 10.1038/s41592-019-0582-9.

[54] Ignacio Arganda-Carreras et al. “Trainable Weka Segmentation: A machine learning tool for microscopy pixel classification”. In: *Bioinformatics* 33.15 (2017), pp. 2424–2426. ISSN: 14602059. DOI: 10.1093/bioinformatics/btx180.

[55] Estibaliz Gómez-de-Mariscal et al. “DeepImageJ: A user-friendly environment to run deep learning models in ImageJ”. In: *Nature Methods* 18.10 (2021), pp. 1192–1195. ISSN: 15487105. DOI: 10.1038/s41592-021-01262-9.

[56] *Ariadne.ai website*. URL: https://ariadne.ai/.

[57] Helen Spiers et al. “Deep learning for automatic segmentation of the nuclear envelope in electron microscopy data, trained with volunteer segmentations”. In: *Traffic* 22.7 (2021), pp. 240–253. ISSN: 16000854. DOI: 10.1111/tra.12789.

[58] Moritz Helmstaedter et al. “Connectomic reconstruction of the inner plexiform layer in the mouse retina”. In: *Nature* 500.7461 (2013), pp. 168–174. ISSN: 14764687. DOI: 10.1038/nature12346.

[59] Clarisse Uwizeye et al. “Morphological bases of phytoplankton energy management and physiological responses unveiled by 3D subcellular imaging”. In: *Nature Communications* 12.1 (2021), pp. 1–12. ISSN: 20411723. DOI: 10.1038/s41467-021-21314-0.

[60] Larissa Heinrich et al. “Whole-cell organelle segmentation in volume electron microscopy”. In: *Nature* November 2020 (2021). ISSN: 0028-0836. DOI: 10.1038/s41586-021-03977-3.

[61] Marta Costa et al. “NBLAST: Rapid, Sensitive Comparison of Neuronal Structure and Construction of Neuron Family Databases”. In: *Neuron* 91.2 (2016), pp. 293–311. ISSN: 08966273. DOI: 10.1016/j.neuron.2016.06.012.

[62] Pascal De Boer, Jacob P. Hoogenboom, and Ben N.G. Giepmans. “Correlated light and electron microscopy: Ultrastructure lights up!” In: *Nature Methods* 12.6 (2015), pp. 503–513. ISSN: 15487105. DOI: 10.1038/nmeth.3400.

[63] David P. Hoffman et al. “Correlative three-dimensional super-resolution and block-face electron microscopy of whole vitreously frozen cells”. In: *Science* 367.6475 (2020). ISSN: 10959203. DOI: 10.1126/science.aaz5357.

[64] Alexander Shakeel Bates et al. “Neuronal cell types in the fly: single-cell anatomy meets single-cell genomics”. In: *Current Opinion in Neurobiology* 56 (2019), pp. 125–134. ISSN: 18736882. DOI: 10.1016/j.conb.2018.12.012.

[65] *WormAtlas Website*. URL: https://www.wormatlas.org/.

[66] Anjali Rao et al. “Exploring tissue architecture using spatial transcriptomics”. In: *Nature* 596.7871 (2021), pp. 211–220. ISSN: 14764687. DOI: 10.1038/s41586-021-03634-9.

[67] Jeffrey R. Moffitt et al. “Molecular, spatial, and functional single-cell profiling of the hypothalamic preoptic region”. In: *Science* 362.6416 (2018). ISSN: 10959203. DOI: 10.1126/science.aau5324.

[68] Sheel Shah et al. “In Situ Transcription Profiling of Single Cells Reveals Spatial Organization of Cells in the Mouse Hippocampus”. In: *Neuron* 92.2 (2016), pp. 342–357. ISSN: 10974199. DOI: 10.1016/j.neuron.2016.10.001.

[69] Chee Huat Linus Eng et al. "Transcriptome-scale super-resolved imaging in tissues by RNA seqFISH+". In: *Nature* (2019). ISSN: 14764687. DOI: 10.1038/s41586-019-1049-y.

[70] Robert M Haralick, K Shanmugam, and Its'Hak Dinstein. "Textural Features for Image Classification". In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-3.6 (1973), pp. 610–621. DOI: 10.1109/TSMC.1973.4309314.

[71] Vincent D Blondel et al. "Fast unfolding of communities in large networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (Oct. 2008), P10008. ISSN: 1742-5468. DOI: 10.1088/1742-5468/2008/10/p10008.

[72] Leland McInnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction.* 2020. arXiv: 1802.03426 [stat.ML].

[73] F Mölder et al. "Sustainable data analysis with Snakemake [version 2; peer review: 2 approved]". In: *F1000Research* 10.33 (2021). DOI: 10.12688/f1000research.29032.2.

[74] Rahul Satija et al. "Spatial reconstruction of single-cell gene expression data". In: *Nature Biotechnology* 33.5 (2015), pp. 495–502. ISSN: 15461696. DOI: 10.1038/nbt.3192. arXiv: arXiv:1011.1669v3.

[75] Nikos Karaiskos et al. "The Drosophila embryo at single-cell transcriptome resolution." In: *Science (New York, N.Y.)* 358.6360 (Aug. 2017), pp. 194–199. ISSN: 1095-9203. DOI: 10.1126/science.aan3235.

[76] Keren Bahar Halpern et al. "Single-cell spatial reconstruction reveals global division of labour in the mammalian liver". In: *Nature* 542.7641 (Feb. 2017), pp. 352–356. ISSN: 0028-0836. DOI: 10.1038/nature21065.

[77] Kaia Achim et al. "Whole-Body Single-Cell Sequencing Reveals Transcriptional Domains in the Annelid Larval Body". In: *Molecular Biology and Evolution* 35.5 (May 2018). Ed. by Gunter Wagner, pp. 1047–1062. ISSN: 0737-4038. DOI: 10.1093/molbev/msx336.

[78] Ricard Argelaguet et al. "Computational principles and challenges in single-cell data integration". In: *Nature Biotechnology* 39.10 (2021), pp. 1202–1215. ISSN: 15461696. DOI: 10.1038/s41587-021-00895-7.

[79] Jan Ellenberg et al. "A call for public archives for biological image data". In: *Nature Methods* 15.11 (2018), pp. 849–854. ISSN: 15487105. DOI: 10.1038/s41592-018-0195-8.

[80] Andrii Iudin et al. "EMPIAR: A public archive for raw electron microscopy image data". In: *Nature Methods* 13.5 (2016), pp. 387–388. ISSN: 15487105. DOI: 10.1038/nmeth.3806.

[81] Eleanor Williams et al. "Image Data Resource: A bioimage data integration and publication platform". In: *Nature Methods* 14.8 (2017), pp. 775–781. ISSN: 15487105. DOI: 10.1038/nmeth.4326.

[82] Chris Allan et al. "OMERO: Flexible, model-driven data management for experimental biology". In: *Nature Methods* 9.3 (2012), pp. 245–253. ISSN: 15487091. DOI: 10.1038/nmeth.1896.

[83] Stephan Saalfeld et al. "CATMAID: Collaborative annotation toolkit for massive amounts of image data". In: *Bioinformatics* 25.15 (2009), pp. 1984–1986. ISSN: 13674803. DOI: `10.1093/bioinformatics/btp266`.

[84] Kevin M. Boergens et al. "WebKnossos: Efficient online 3D data annotation for connectomics". In: *Nature Methods* 14.7 (2017), pp. 691–694. ISSN: 15487105. DOI: `10.1038/nmeth.4331`.

[85] *Neuroglancer*. URL: `https://github.com/google/neuroglancer`.

[86] C. Shan Xu et al. "An open-access volume electron microscopy atlas of whole cells and tissues". In: *Nature* 599.7883 (2021), pp. 147–151. ISSN: 14764687. DOI: `10.1038/s41586-021-03992-4`.

[87] *The HDF5 Library and File Format*. URL: `https://www.hdfgroup.org/solutions/hdf5/`.

[88] Josh Moore et al. "OME-NGFF: a next-generation file format for expanding bioimaging data-access strategies". In: *Nature Methods* 18.12 (2021), pp. 1496–1498. ISSN: 15487105. DOI: `10.1038/s41592-021-01326-w`.

[89] Vineet Choudhary et al. "Seipin and Nem1 establish discrete ER subdomains to initiate yeast lipid droplet biogenesis". In: *Journal of Cell Biology* 219.7 (2020). ISSN: 15408140. DOI: `10.1083/JCB.201910177`.

[90] Anna Lena Eberle and Dirk Zeidler. "Multi-beam scanning electron microscopy for high-throughput imaging in connectomics research". In: *Frontiers in Neuroanatomy* 12.December (2018), pp. 1–7. ISSN: 16625129. DOI: `10.3389/fnana.2018.00112`.

[91] Perrine Paul-Gilloteaux et al. "EC-CLEM: Flexible multidimensional registration software for correlative microscopies". In: *Nature Methods* 14.2 (2017), pp. 102–103. ISSN: 15487105. DOI: `10.1038/nmeth.4170`.

[92] Christian Tischer. *ElastixWrapper: Fiji plugin for 3D image registration with elastix*. Mar. 2019. DOI: `10.5281/zenodo.2602549`. URL: `https://doi.org/10.5281/zenodo.2602549`.

[93] *elastix-napari*. URL: `https://www.napari-hub.org/plugins/elastix-napari`.

[94] Maarten W. Tuijtel et al. "Inducing fluorescence of uranyl acetate as a dual-purpose contrast agent for correlative light-electron microscopy with nanometre precision". In: *Scientific Reports* 7.1 (2017), pp. 1–12. ISSN: 20452322. DOI: `10.1038/s41598-017-10905-x`.

[95] Rick Seifert et al. "DeepCLEM: automated registration for correlative light and electron microscopy using deep learning [version 1; peer review: 2 approved with reservations]". In: *F1000Research* 9 (2021), pp. 1–10. ISSN: 1759796X. DOI: `10.12688/F1000RESEARCH.27158.1`.

[96] Eric M. Christiansen et al. "In Silico Labeling: Predicting Fluorescent Labels in Unlabeled Images". In: *Cell* 173.3 (2018), 792–803.e19. ISSN: 10974172. DOI: `10.1016/j.cell.2018.03.040`.

[97] *BioImage Model Zoo*. URL: `https://bioimage.io`.

[98] Jonas Hartmann et al. “An image-based data-driven analysis of cellular architecture in a developing tissue”. In: *eLife* 9 (2020), pp. 1–33. ISSN: 2050084X. DOI: 10.7554/eLife.55913.

[99] F. Cipriani et al. “Automation of sample mounting for macromolecular crystallography”. In: *Acta Crystallographica Section D: Biological Crystallography* 62.10 (2006), pp. 1251–1259. ISSN: 09074449. DOI: 10.1107/S0907444906030587.

[100] Maxim Polikarpov et al. “Visualization of protein crystals by high-energy phase-contrast X-ray imaging”. In: *Acta Crystallographica Section D: Structural Biology* 75 (2019), pp. 947–958. ISSN: 20597983. DOI: 10.1107/S2059798319011379.

[101] S. Zabler et al. “Optimization of phase contrast imaging using hard x rays”. In: *Review of Scientific Instruments* 76.7 (2005). ISSN: 00346748. DOI: 10.1063/1.1960797.

[102] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[103] P. Cloetens et al. “Holotomography: Quantitative phase tomography with micrometer resolution using hard synchrotron radiation x rays”. In: *Applied Physics Letters* 75.19 (1999), pp. 2912–2914. ISSN: 00036951. DOI: 10.1063/1.125225.

[104] Doğa Gürsoy et al. “TomoPy: A framework for the analysis of synchrotron tomographic data”. In: *Journal of Synchrotron Radiation* 21.5 (2014), pp. 1188–1193. ISSN: 16005775. DOI: 10.1107/S1600577514013939.

[105] Stéfan Van Der Walt et al. “Scikit-image: Image processing in python”. In: *PeerJ* 2014.1 (2014), pp. 1–18. ISSN: 21678359. DOI: 10.7717/peerj.453.

[106] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[107] Luis Pedro Coelho. “Mahotas: Open source software for scriptable computer vision”. In: *Journal of Open Research Software* 1.1 (2013), e3. ISSN: 2049-9647. DOI: 10.5334/jors.ac. arXiv: 1211.4907.

[108] F Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[109] A A Hagberg, D A Schult, and P J Swart. “Exploring network structure, dynamics, and function using NetworkX”. In: *7th Python in Science Conference (SciPy 2008)* SciPy (2008), pp. 11–15.

[110] Wes McKinney. “Data Structures for Statistical Computing in Python”. In: *Proceedings of the 9th Python in Science Conference* 1.Scipy (2010), pp. 56–61. DOI: 10.25080/majora-92bf1922-00a.

[111] Charles R. Harris et al. “Array programming with NumPy”. In: *Nature* 585.7825 (2020), pp. 357–362. ISSN: 14764687. DOI: 10.1038/s41586-020-2649-2. arXiv: 2006.10256.

[112] Hadley Wickham et al. “Welcome to the Tidyverse”. In: *Journal of Open Source Software* 4.43 (2019), p. 1686. ISSN: 2475-9066. DOI: 10.21105/joss.01686.

[113] Aaron Meurer et al. “SymPy: Symbolic computing in python”. In: *PeerJ Computer Science* 2017.1 (2017), pp. 1–27. ISSN: 23765992. DOI: `10.7717/peerj-cs.103`.

Appendix A

Ultramicrotome solutions

A.1 Solving for sample tilt

Variables:

Variable	Explanation
$\boldsymbol{R_x}$	Rotation matrix about the x axis
$\boldsymbol{R_y}$	Rotation matrix about the y axis
$\boldsymbol{R_z}$	Rotation matrix about the z axis
θ_T	Tilt angle of sample
θ_R	Rotation angle of sample
θ_K	Tilt angle of knife
θ_{IT}	Initial tilt of sample (on alignment)
θ_{IK}	Initial tilt of knife (on alignment)
θ_{to}	Target offset angle i.e. block face to target plane z axis rotation (measured from X-ray)
θ_{tr}	Target rotation angle i.e. block face to target plane x axis rotation (measured from X-ray)

Constants:

Constant	Full expression
A	$cos(\theta_{IK} + \theta_{to})$
B	$sin(\theta_{tr})sin(\theta_{IK} + \theta_{to})$
C	$sin(\theta_{IT})sin(\theta_{IK} + \theta_{to})$
D	$cos(\theta_{IT})sin(\theta_{IK} + \theta_{to})$
E	$cos(\theta_{tr})sin(\theta_{IK} + \theta_{to})$
F	$sin(\theta_{IT})cos(\theta_{tr})$
G	$sin(\theta_{tr})cos(\theta_{IT})$
H	$sin(\theta_{IT})sin(\theta_{tr})$
I	$cos(\theta_{IT})cos(\theta_{tr})$

Forward kinematics equation, describing the rotation from the World coordinate frame to the target plane:

$$\boldsymbol{F} = \boldsymbol{R_x}(\theta_T)\boldsymbol{R_y}(\theta_R)\boldsymbol{R_x}(-\theta_{IT})\boldsymbol{R_z}(\theta_{IK})\boldsymbol{R_z}(\theta_{to})\boldsymbol{R_x}(\theta_{tr}) \tag{A.1}$$

Equation for positions where target plane is vertical, and therefore reachable by the knife:

$$(\boldsymbol{Fy}) \cdot \boldsymbol{z} = 0$$

where $\boldsymbol{y}$ is the y vector $\left(\begin{smallmatrix}0\\1\\0\end{smallmatrix}\right)$ and $\boldsymbol{z}$ is the z vector $\left(\begin{smallmatrix}0\\0\\1\end{smallmatrix}\right)$.

Using sympy[113] to help with rearrangement this gives:

$$\begin{aligned}\sin(\theta_T)\Big(\sin(\theta_{tr})\sin(\theta_{IT}) + \cos(\theta_{IT})\cos(\theta_{tr})\cos(\theta_{IK}+\theta_{to})\Big)\\ + \cos(\theta_R)\cos(\theta_T)\Big(-\sin(\theta_{IT})\cos(\theta_{tr})\cos(\theta_{IK}+\theta_{to}) + \sin(\theta_{tr})\cos(\theta_{IT})\Big)\\ + \sin(\theta_R)\cos(\theta_T)\Big(\sin(\theta_{IK}+\theta_{to})\cos(\theta_{tr})\Big) = 0\end{aligned}$$

As θ_{IT}, θ_{IK}, θ_{to} and θ_{tr} are constants for a particular targeting run, we can replace these with the A-I constants defined above to make the expression easier to work with.

$$(AI + H)\sin(\theta_T) + (-AF + G)\cos(\theta_R)cos(\theta_T) + E\sin(\theta_R)\cos(\theta_T) = 0$$

$$(-AF + G)\cos(\theta_R)cos(\theta_T) + E\sin(\theta_R)\cos(\theta_T) = -(AI + H)\sin(\theta_T)$$

$$(-AF + G)\cos(\theta_R) + E\sin(\theta_R) = \frac{-(AI + H)\sin(\theta_T)}{\cos(\theta_T)}$$

$$\tan(\theta_T) = \frac{-AF + G}{-AI - H}\cos(\theta_R) + \frac{E}{-AI - H}\sin(\theta_R)$$

This means the final solution for the sample tilt is:

$$\theta_T = \arctan\left(\frac{-AF + G}{-AI - H}\cos(\theta_R) + \frac{E}{-AI - H}\sin(\theta_R)\right) \tag{A.2}$$

This solution can of course be simplified further by combining these constants together to give something of the form:

$$\theta_T = \arctan\Big(C_1\cos(\theta_R) + C_2\sin(\theta_R)\Big)$$

In practice, we use the form with constants A-I, as this allows the same constants to be used for this and the knife solution below.

The script for the sympy parts is available in the Crosshair github repository: `https://github.com/K-Meech/crosshair/blob/master/python_scripts/multiple_solutions.py`

A.2 Solving for knife tilt

Now we must find the knife angle that corresponds to a particular sample rotation. For this, we need the signed angle between global y (i.e. microtome NS) and local y (the target plane's normal) in the z plane (i.e. the plane with the global z axis as its normal). This is because the knife can only rotate about the z axis, so we only need the angle within this plane.

This is equal to:

$$\theta_K = \arctan\left(\frac{(\boldsymbol{y} \times (\boldsymbol{F}\boldsymbol{y})) \cdot \boldsymbol{z}}{\boldsymbol{y} \cdot (\boldsymbol{F}\boldsymbol{y})}\right)$$

where $\boldsymbol{y}$ is the y vector $\left(\begin{smallmatrix}0\\1\\0\end{smallmatrix}\right)$, $\boldsymbol{z}$ is the z vector $\left(\begin{smallmatrix}0\\0\\1\end{smallmatrix}\right)$ and $\boldsymbol{F}$ is the matrix from the forward kinematics equation defined above in equation A.1.

Using sympy[113] to help with rearrangement, and substitution of the same constants A-I from above, this becomes:

$$\theta_K = \arctan\left(\frac{E\cos(\theta_R) + (AF - G)\sin(\theta_R)}{-E\sin(\theta_R)\sin(\theta_T) + (AF - G)\sin(\theta_T)\cos(\theta_R) + (AI + H)\cos(\theta_T)}\right)$$

This equation gives the corresponding signed knife angle for all values of θ_T and θ_R, but we only want the values that are valid given the solution calculated above for the sample tilt. I.e. we only want knife angles where the target plane is vertical, and therefore reachable by the knife. Therefore, we substitute in the solution from above (equation A.2) for θ_T.

This gives (with some rearrangement / simplification with sympy):

$$\theta_K = \arctan\left(\frac{\Big(AI + H\Big)\Big(E\cos(\theta_R) + (AF - G)\sin(\theta_R)\Big)}{\left(\sqrt[2]{\Big(AI + H\Big)^2 + \Big(E\sin(\theta_R) + (-AF + G)\cos(\theta_R)\Big)^2}\right)|AI + H|}\right) \tag{A.3}$$

This solution can of course be simplified further by combining these constants together to give something of the form:

$$\theta_K = \arctan\left(\frac{C_1\Big(C_2\cos(\theta_R) + C_3\sin(\theta_R)\Big)}{\left(\sqrt[2]{{C_1}^2 + \Big(C_2\sin(\theta_R) - C_3\cos(\theta_R)\Big)^2}\right)|C_1|}\right)$$

In practice, we use the form with constants A-I in the Crosshair code.

Therefore, with equation A.2 and equation A.3 we can find the corresponding sample tilt and knife tilt for any given sample rotation.

The script for the sympy parts is available in the Crosshair github repository: `https://github.com/K-Meech/crosshair/blob/master/python_scripts/multiple_solutions.py`

Appendix B

Targeting workflow

Detailed step by step of the targeting workflow outlined in figure 2.12.

1. Trim Block
Trim the block at the ultramicrotome with a diamond knife. A flat surface must be made that has 4 corners and straight lines in between. Make sure you trim this block face deep enough, so that the knife will always touch part of that face first and not other parts of the block.

2. X-ray
X-ray your block at the highest resolution possible (the resolution here will determine how accurate measures made from the X-ray can be). Ensure the sample and the entirety of the block surface are visible.

After the X-ray, compare your image stack and your sample. If it appears as a mirror image of your sample, then flip your image stack. This can be done e.g. in Fiji with Image > Transform > Flip Vertically.

3. Define block + target plane
Open your X-ray in Crosshair and set your block and target planes. The target plane can be set using the 'TRACK' button and navigating in the 2D viewer. The block face can be set by placing a series of points on the block surface and fitting a plane to them. This step is described in the Crosshair wiki on github.

4. Define block vertices + orientation
Place one point on each of the 4 corners of your block face. Then, decide on the orientation the block will have in the microtome i.e. which side of the block will face up? Mark the points accordingly as 'Top Left', 'Top Right', 'Bottom Left' and 'Bottom Right'. This step is described in the Crosshair wiki on github.

Once this is complete, it is good practice to 'Save Settings' in Crosshair to make a .json file as a record of your planes and points.

5. Set sample tilt 0
Set the sample tilt to zero by eye (just by looking at the scale on the arc piece and trying to get it as close to 0 as possible). Press 'Set Zero' on the semi-automated microtome's touchscreen.

5. Set knife tilt 0
For the knife zero, there is a slightly more complicated procedure than for the sample tilt. We assumed that since the knife is removed and replaced every time, there may be more variation in its position and therefore an extra step may be required to accurately find its proper zero.

Put a blank block into the microtome sample holder (i.e. just resin with no sample). Set the knife tilt to zero by eye (just by looking at the scale on the knife holder and trying to get it as close to zero as possible). Cut into the blank block at this orientation to make a flat face. Back the knife away, and rotate the sample holder exactly 90 degrees (using the touchscreen controls). Align the knife to this face again (only adjusting the knife tilt!). Press 'Set Zero' on the touchscreen.

6. Align knife and block face
Replace the blank block with your resin-embedded sample. Make sure you put it the same way up as you decided in Crosshair.

Disable the motors and manually align the knife and block face, ensuring the bottom edge of the block is also aligned along the cutting edge of the knife. (You can freely change any axis for this, just as you would normally).

7. Set sample rotation 0
Set the sample rotation to zero by pressing 'Set Zero' on the touchscreen.

8. Enter θ_{IK} and θ_{IT}
Enter the current knife angle and sample tilt angle into Crosshair.

9. Choose solution
Use the Crosshair 'Microtome Mode' to cycle through all the possible solutions, and choose the one you want to use. This step is described in the Crosshair wiki on github.

Once this is complete, it is good practice to 'Save Solution' in Crosshair to make a .json file as a record of your solution.

9. Set microtome to solution angles
Back the knife away from the block, to ensure there is no risk of hitting it. Enable the motors, and move each axis to the angles stated in the solution from Crosshair.

10. Approach block surface + cut solution distance
Manually approach the block and get as close as possible to its surface. Any error in this distance (i.e. starting before touching the block, or once you've already cut into it) will contribute to your final error. A good solution is to get as close as possible with the NS movement, then set the cutting thickness to some small value (e.g. 70-100nm).

Then you can start cutting slowly, carefully watching the knife edge, and stop as soon as you see any debris on the knife / any pieces being cut from the block. This will ensure you are as close as possible to the true surface.

Once complete, cut the distance specified in the Crosshair solution.

Appendix C

Morphological features

Feature id	Segmentations	Description
Shape_volume _in_microns	Cell, nucleus, H+N, euchromatin	Volume in μm^3
shape_extent	Cell, nucleus, H+N, euchromatin	Ratio of pixels in the object to pixels in the total bounding box
shape_equiv_diameter	Cell, nucleus, H+N, euchromatin	Equivalent diameter - the diameter of a sphere with the same volume as the object
shape_major_axis	Cell, nucleus, H+N, euchromatin	Length of the major axis of the fitted ellipsoid
shape_minor_axis	Cell, nucleus, H+N, euchromatin	Length of the minor axis of the fitted ellipsoid
shape_surface_area	Cell, nucleus, H+N, euchromatin	Surface area of object mesh (mesh calculated by the Lewiner marching cubes algorithm)
shape_sphericity	Cell, nucleus, H+N, euchromatin	Measure of how spherical an object is (0-1 scale, with 1 being a perfect sphere) calculated as $36V^2/S^3$ where V is the volume of the object, and S is its surface area.
shape_max_radius	Cell, nucleus, H+N, euchromatin	Maximum distance from a pixel within the object to the outside (Euclidean distance)
intensity_mean	Cell (excluding nucleus), nucleus, H+N, euchromatin	Mean intensity of the segmented object

intensity_st_dev	Cell (excluding nucleus), nucleus, H+N, euchromatin	Standard deviation of intensity of the segmented object
intensity_median	Cell (excluding nucleus), nucleus, H+N, euchromatin	Median intensity of the segmented object
intensity_iqr	Cell (excluding nucleus), nucleus, H+N, euchromatin	Interquartile range (iqr) of intensity of the segmented object
intensity_total	Cell (excluding nucleus), nucleus, H+N, euchromatin	Sum of intensity values of the segmented object
Intensity_mean_25 Intensity_mean_50 Intensity_mean_75 Intensity_mean_100	Nucleus	Mean intensity of different radial zones of the nucleus.
Intensity_st_dev_25 Intensity_st_dev_50 Intensity_st_dev_75 Intensity_st_dev_100	Nucleus	Standard deviation of intensity of different radial zones of the nucleus.
Intensity_median_25 Intensity_median_50 Intensity_median_75 Intensity_median_100	Nucleus	Median intensity of different radial zones of the nucleus.
Intensity_iqr_25 Intensity_iqr_50 Intensity_iqr_75 Intensity_iqr_100	Nucleus	Interquartile range (iqr) of intensity of different radial zones of the nucleus.
Intensity_total_25 Intensity_total_50 Intensity_total_75 Intensity_total_100	Nucleus	Sum of intensity values of different radial zones of the nucleus.

Texture_hara1 Texture_hara2 Texture_hara3 Texture_hara4 Texture_hara5 Texture_hara6 Texture_hara7 Texture_hara8 Texture_hara9 Texture_hara10 Texture_hara11 Texture_hara12 Texture_hara13	Cell (excluding nucleus), nucleus, H+N, euchromatin	Haralick texture features of the segmented object. Haralick texture features 1-13 are commonly used texture descriptors in image analysis - each is a statistic derived from the grey level co-occurrence matrix of an image[70].
Shape_edt_mean _(het/eu)_nucleus	H+N, euchromatin	Mean of values in normalised euclidean distance transform (edt) of whole nucleus covered by the current segmented object (either euchromatin or heterochromatin + nucleolus segmentation). This is a measure of the distribution of chromatin within the nucleus (low values indicate a distribution mostly towards the edge of the nucleus, while higher values indicate a distribution closer to the centre). The euclidean distance transform is calculated for the whole nucleus segmentation, and normalised to run from 0 to 1 (1 being the point furthest from the edge).
Shape_edt_stdev _(het/eu)_nucleus	H+N, euchromatin	Standard deviation of values in normalised euclidean distance transform (edt) of whole nucleus covered by the current segmented object (either euchromatin or heterochromatin + nucleolus segmentation). This is a measure of the distribution of chromatin within the nucleus (high values indicate a varied distribution in the nucleus i.e. some towards the outside of the nucleus, as well as some towards the centre). The euclidean distance transform is calculated for the whole nucleus segmentation, and normalised to run from 0 to 1 (1 being the point furthest from the edge).
Shape_edt_median _(het/eu)_nucleus	H+N, euchromatin	Same as shape_edt_mean_(het/eu)_nucleus, but using median rather than the mean
Shape_edt_iqr _(het/eu)_nucleus	H+N, euchromatin	Same as shape_edt_stdev_(het/eu)_nucleus, but using interquartile range (iqr) rather than standard deviation.
Shape_percent_25 _(het/eu)_nucleus Shape_percent_50 _(het/eu)_nucleus Shape_percent_75 _(het/eu)_nucleus Shape_percent_100 _(het/eu)_nucleus	H+N, euchromatin	Percent of different radial zones in the nucleus that are filled by either the heterochromatin+nucleolus or euchromatin segmentation - this is a measure of the distribution of chromatin within the nucleus.

Table C.1: Description of morphological, intensity and texture features used in clustering analysis. Note: some features are calculated for a number of different segmentations (as indicated by the second column), so e.g. Shape_volume_in_microns is calculated for 4 segmentations, and results in 4 features in the final table. Het / eu in feature names are short for 'heterochromatin + nucleolus' (H+N) and 'euchromatin' respectively. Some intensity and texture features are calculated for 'Cell (excluding nucleus)' i.e. the nucleus segmentation is used to mask out that part of the cell, and allow the measurement to be made only for the cytoplasm and organelles. Some features are measured within 'radial zones' of the nucleus - these zones are calculated from a euclidean distance transform of the nucleus, that is normalised to run from 0 to 1 (1 being the point in the nucleus furthest from the edge). Zone 25 is then the outermost 25% (values 0-0.25), zone 50 from 25% to 50% (values 0.25-0.5), zone 75 from 50% to 75% (values 0.5-0.75) and zone 100 from 75% to 100% (0.75-1.0). The full table of calculated features for cells is available here: `https://github.com/mobie/platybrowser-project/blob/main/data/1.0.1/tables/sbem-6dpf-1-whole-segmented-cells/morphology.tsv` and for nuclei (+ chromatin) here: `https://github.com/mobie/platybrowser-project/blob/main/data/1.0.0/tables/sbem-6dpf-1-whole-segmented-nuclei/morphology.tsv`. The clustering derived from both these sets of features combined is available here: `https://github.com/mobie/platybrowser-project/blob/main/data/1.0.1/tables/sbem-6dpf-1-whole-segmented-cells/morphology_clusters.tsv`