FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Visualisation and Interaction Techniques for the Exploration of the Fruit Fly's Neural Structure

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Nicolas Swoboda
Matrikelnummer 0425828

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Ao.Univ.-Prof. Univ.-Doz. Dipl.-Ing. Dr.techn. Eduard Gröller
Mitwirkung: Dipl.-Math. Dr. Katja Bühler

Wien, 27.02.2020

(Unterschrift Verfasser) (Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Visualisation and Interaction Techniques for the Exploration of the Fruit Fly's Neural Structure

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Nicolas Swoboda
Registration Number 0425828

to the Faculty of Informatics
at the TU Wien

Advisor: Ao.Univ.-Prof. Univ.-Doz. Dipl.-Ing. Dr.techn. Eduard Gröller
Assistance: Dipl.-Math. Dr. Katja Bühler

Vienna, 27.02.2020

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Nicolas Swoboda
Schloßgasse 2/1/9, 1050 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

Acknowledgements

This thesis was made possible by the VRVis Zentrum für Virtual Reality und Visualisierung Forschungs-GmbH. The research company is funded by BMVIT, BMWFW, Styria, SFG and Vienna Business Agency in the scope of COMET - Competence Centers for Excellent Technologies (854174) which is managed by FFG.

My thanks go to Katja Bühler and Meister Eduard Gröller for their helpful advice and patience during the course of my thesis, and for reviewing the work. I want to thank Florian Schulze and everyone else at the VRVis who helped with all my problems. Stefan Bruckner, thank you for your support and thank you especially for waking my interest in the topic. I thank Judith Moosburner who developed the beautiful and innovative design as her Diploma Thesis at the University of the Arts in Zürich. Thank you to all interviewees for helping with the evaluation and the constructive feedback.

I thank my family for their unending patience and support. I thank all my friends who encouraged me to continue work on this project when my focus was elsewhere. Finally, an honest thank you to the Republic of Austria for the free access to higher education.

Abstract

In their studies of the brain of the common fruit fly *Drosophila melanogaster*, neurobiologists investigate neural connectivity with the goal to discover how complex behaviour is generated. Gaining information on *potential connectivity* between neurons is an essential step in their workflow. This thesis presents a way to compute and visualise such potential connectivity information from segmented neurons. It shortens the tedious month-long search for potential connectivity down to a few minutes.

Overlaps of arborisations of two or more neurons indicate a potential anatomical connection, and thus a potential functional connection. The computation of this data starts from neuron meshes. The meshes—segmented from confocal light-microscopy images of the fruit fly—are intersected to find overlapping areas, i.e. areas of potential anatomical connectivity. This information can then help to discover actual functional connectivity in a neural circuit.

Analysing higher order overlaps, i.e. intersections of more than two arborisations of segmented neuron data in the same location, poses new challenges. The visualisation in 2D sections or 3D is impeded by visual clutter and occlusion. Computation of relevant volumetric information becomes difficult for higher order overlaps, because the number of possible overlaps increases exponentially with the number of arborisations. This makes the pre-computation for all possible combinations infeasible. Previous tools have thus been restricted to the quantification and visualisation of pairwise overlaps.

The thesis presents a novel solution addressing these issues for higher order overlaps. A novel abstracting design is coupled with a modern approach for on-demand GPU computations. Our tool calculates for the first time volumetric information of higher order overlaps on the GPU using A-buffers. The thesis addresses the visual complexity of the data with the implementation of an innovative novel design created by the graphics designer Judith Moosburner. We realised this design using non-photorealistic rendering techniques and perspicuous user interfaces, including interactive glyphs and linked views on quantitative overlap information. To complement the neuroscientists' workflows the resulting interactive tool has been integrated into BrainGazer, a software tool for advanced visualisation and exploration of neural images and circuit data. Qualitative evaluation with neuroscientists and non-expert users demonstrated the utility and usability of the tool.

Kurzfassung

Biologen untersuchen neuronale Verbindungen im Gehirn der Fruchtfliege *Drosophila melanogaster* um die Entstehung komplexen Verhaltens zu verstehen. Informationen über die *potentielle Konnektivität* zwischen Neuronen ist dafür essentiell. Diese Diplomarbeit beschreibt eine Lösung für die Berechnung und Visualisierung dieser potentiellen Konnektivität von segmentierten Neuronen. Der bisher mehr-monatige Aufwand einer Suche nach potentieller Konnektivität reduziert sich auf einige Minuten.

Eine Überlappung zweier oder mehrerer Arborisierungen (Verästelungen von Neuronen) impliziert eine potentielle anatomische Verbindung, was wiederum eine tatsächliche funktionale Verbindung der Neuronen nahe legt. Berechnet werden diese Schnitte von Meshes der Neuronen. Die Meshes, segmentiert von Lichtmikroskopiebildern, werden miteinander geschnitten um überlappende Regionen zu finden. Diese Regionen potentieller anatomischer Konnektivität helfen bei der Suche nach funktionaler Konnektivität in einem neuronalen Netz.

Die Analyse von Schnitten höherer Ordnung (Überlappungen von mehr als zwei Arborisierungen) stellt uns vor neue Aufgaben. Die Visualisierung mehrerer Meshes in 2D oder 3D erzeugt Verdeckungen. Die Berechnung der volumetrischen Informationen wird schwierig, denn die Anzahl der möglichen Schnitte steigt exponentiell mit der Anzahl der Meshes. Eine Vorberechnung ist daher nicht umsetzbar. Bisherige Werkzeuge beschränken die Quantifizierung und Visualisierung auf paarweise Schnitte.

Die Diplomarbeit präsentiert eine Implementierung, die sich dieser Probleme widmet. Sie verbindet ein neuartiges abstrahierendes Design mit einem modernen Ansatz für on-demand GPU Berechnungen. Zum ersten Mal werden A-Buffer auf der GPU für volumetrische Berechnungen genutzt. Das neue innovative Design von Grafikdesignerin Judith Moosburner behandelt die komplexen visuellen Ansprüche der neuronalen Daten. Mittels nicht-fotorealistischer Rendering Techniken, interaktiver Glyphen in 2D und 3D, verständlicher Benutzeroberflächen und verlinkter Darstellungen der quantitativen Daten haben wir das Design verwirklicht. Diese Implementierung wurde in BrainGazer integriert, um die Arbeitsschritte der Neurowissenschafter zu unterstützen. Eine qualitative Evaluierung mit Neurobiologen und Nichtfachleuten belegt den Nutzen des neuen Werkzeugs und zeigt, dass es intuitiv verwendbar ist.

Contents

CHAPTER 1

Introduction

The fruit fly *Drosophila melanogaster* has long been a popular model organism in the neuroscience community. More specifically in the field of *circuit neuroscience* it is studied to link complex behaviour with neuronal circuits in the brain. In cooperation with our partners at the Institute of Molecular Pathology (IMP) in Vienna we develop tools to improve the scientists' workflows or create novel views on the fly's data. The scientists acquire this data via confocal microscopy to create 3D images of *Drosophila*'s brain. The images of a few thousand flies are stored, along with meta-information, in a relational database.

In the course of this thesis I have published and presented many parts of this research at the 2014 Eurographics Workshop on Visual Computing for Biology and Medicine (VCBM) [68] and in the Eurographics Computer Graphics Forum (CGF) [69].

1.1 Problem Statement

Neuroscientists investigate relations between genes, neurons, and behaviour to gain insights into how the brain works. They form hypotheses linking behaviour with complex neuronal circuits. To arrive at such hypotheses they need to investigate their data for *potential connectivity* among neurons.

Via confocal light-microscopy the scientists acquire data about shape and location of single neurons inside the brain of Drosophila. From carefully crafted single-cell images (i.e. images where a single neuron is genetically marked and fluoresces because of a special protein) scientists semi-automatically segment neurons into neuron meshes. Often the source of the segmentation is instead an average of approximately five single-cell images of the same cell to account for biological variability.

Tedious pair-wise comparisons of neuron meshes lead the biologists to potential-connectivity information from which they are able to form hypotheses about connectivity. A novel hypothesis on connectivity then requires further biological experiments to confirm or reject it.

With the number of segmented neurons rising and a new interest in intersections of three and more neuron meshes (i.e. higher order intersections), the number of possible intersections explodes exponentially. Using current workflows it is not realistically possible to investigate higher order intersections.

1.2 Requirements

The neurobiologists require an interactive system to visualise and quantify neuron mesh intersections in an efficient way. Their data is stored as meshes of so-called *arborisations*, the terminal branching structures of the neurons where synapses communicate with other neurons. The system must make it possible to investigate intersections/overlaps of arborisations. This must include *higher order arborisation overlaps*, i.e. overlaps of three or more meshes.

Investigating arborisations for potential connectivity requires first and foremost an uncluttered visualisation in 3D. This must be complemented by a quantification of the overlaps, both in absolute and relative sizes of overlap volumes. The interactive tool must be intuitively integrated with other tools (neuroMap [65], heat maps, etc.) and thus with the neurobiologists' current workflows.

1.3 Thesis Overview

Chapter 2 gives a brief introduction to the scientific field of neurobiology and issues relevant to the thesis. For more information on the motivation of the thesis we refer to Sections 2.3 and 2.4. There we give details on the data we work on and pose *core questions* which our system aims to solve.

Chapter 3 ties this thesis to related works on similar topics over recent years. Chapter 4 discusses design choices in general - more specific interaction techniques follow in Chapter 7. Technical implementation challenges and their solutions are discussed in Chapters 5 and 6. Finally we discuss the user experience in Chapter 8 and offer conclusions in Chapter 9.

CHAPTER 2

Introduction to Neural Circuitry

This chapter gives a brief introduction to the field of neural circuitry and the scientists' workflows. Building on this, we present three *core questions*. The goal of this thesis is to support the scientists at the IMP [29] in Vienna in answering those questions.

2.1 Circuit Neuroscience

To articulate the contribution of this thesis it is first necessary to present the challenges in the field of circuit neuroscience. Neuroscience is the study of the nervous system. More specifically, circuit neuroscience tries to understand the computational function of a neural circuit, as well as linking this function with the circuit micro-structure [77]. Over a century ago scientists identified the neuron as the functional unit of the brain. The scientific consensus at the moment is to represent the brain as a complex network with functionally connected units at several hierarchical levels of organisation (columns, areas) [67]. Circuit neuroscience investigates biological circuits by reverse-engineering them, to comprehend their structure and logic, and ultimately their computational algorithms [77]. Olsen and Wilson [53] define this *'circuit-cracking'* with the following five points: *'To completely solve a neural circuit would require*

1. *describing a behaviour whose neural circuit mechanisms we seek to understand,*
2. *identifying which neurons are involved,*
3. *determining what drives activity in each type of neuron and how these signals are transformed through the circuit,*
4. *discovering the cellular, synaptic, and circuit mechanisms underlying these neural transformations, and*

5. *understanding why these neural transformations are useful intermediates in producing this behaviour.'* [53]

As part of this, a *'fundamental challenge is to decipher the connectivity diagram of neural circuits, one of the holy grails of neuroscience'* [77].

Documenting the *connectome*, the complete set of neural connections, in the human brain would be the ultimate goal. Understanding every neural circuit in every species would be impossible and pointless, however a comparison of several circuits across different species should help with this goal [53]. Therefore, brain networks are analysed on different levels with different imaging techniques. Whereas the tiny brain of the worm *Caenorhabditis elegans* (around 300 neurons) was completely deciphered on the synaptic level using electron microscopy (EM), this is technically not feasible for larger brains [34]. Recently EM data has been produced for a larval and an adult brain of *Drosophila melanogaster*, but the process is slow and costly [59]. The data thus exist only for single individuals and may help to confirm findings from higher levels of resolution. For larger brains, e.g., mice, optical techniques can provide a resolution on a cellular level but not on a synaptic level [42]. Imaging techniques used on the human brain, such as magnetic resonance imaging (MRI), help study connectivity on even higher levels, i.e. between brain regions [67].

Scientists use model organisms—such as *Caenorhabditis elegans*, fruit flies, mice, or macaques—as research objects to eventually gain insight into neural connectivity within the human brain. Model organisms can be quickly and cheaply bred in large numbers and studying them raises only limited ethical concerns. They must provide some behavioural characteristics that can be linked to neural circuits.

Caenorhabditis elegans is the first species to have its connectome completely deciphered. In the 1970s and 1980s researchers imaged extremely thin slices using serial electron microscopy. This made it possible to virtually reconstruct every neuron and find every synaptic connection between neurons [61]. With its 302 neurons, however, it provides only limited behaviour.

2.2 Drosophila Melanogaster

The fruit fly is the main research subject of the team of neuroscientists at the IMP. Its brain's complexity lies between that of the simple *C. elegans* and the huge complexity of a mouse brain. The fly is only 2.5 mm in length and features a brain of about 100,000 neurons with two halves and a ventral nerve cord (VNC) (Figure 2.1).

There are two major reasons why *Drosophila* is often chosen as model organism. First of all the fly breeds and matures quickly. Females lay up to 400 eggs every 16 days and the development from egg to adult takes only seven days under controlled conditions. Keeping and growing a population is very cheap. Second the genetic toolbox is

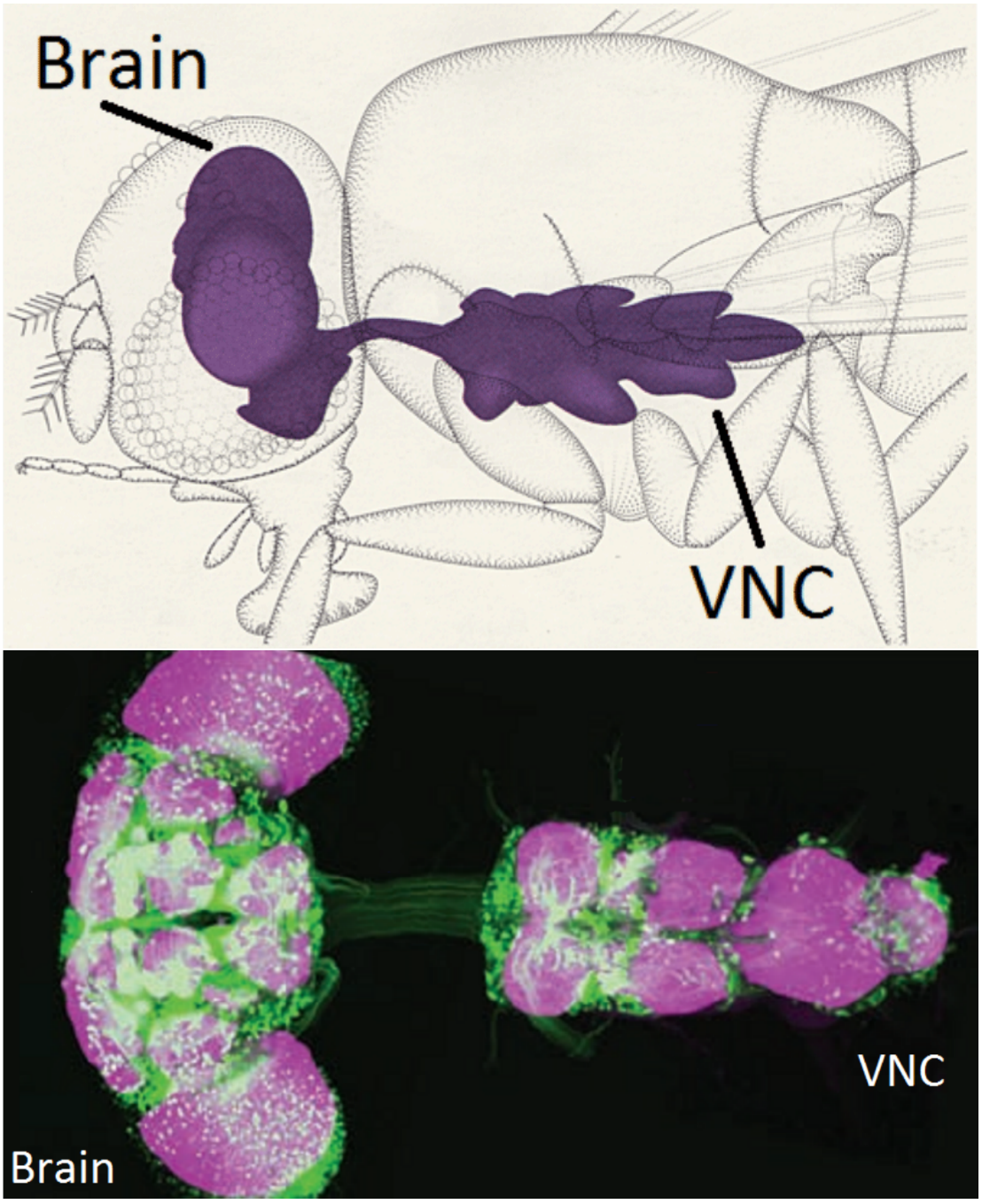


Figure 2.1: *Drosophila*'s brain and ventral nerve cord (VNC). top: anatomical drawing [71], bottom: a male fly's confocal microscopy image using the GAL4/UAS system; nc82 staining reveals brain tissue (magenta), green fluorescent protein (GFP) highlights groups of neurons [20].

extensive, although it is rivalled by that of the mouse. The genome has been sequenced and well annotated. Modifying and mapping chromosomes is fairly easy. Along with these two major reasons comes the advantage that experimentation with the fruit fly does not raise ethical concerns.

There are many more reasons to choose this rewarding model organism when conducting circuit neuroscience. Very relevant is its numerical simplicity. In comparison the mouse brain has about 1000 times more neurons. Another feature are stereotyped neurons, which can be located (in theory) in every fly [53]. The fruit fly is the smallest model animal with a brain. The brain shows rudimentary consciousness, and high abilities such as learning and memory. It perceives sound, vision, and smell and features complex traits, some of which have clear human homologs, e.g., circadian rhythm, sleep, drug responses, locomotion, aggressive behaviour, and longevity [63].

The fly's similarities with humans go even further making it a viable model organism in genetics. Evolutionary conservation of key genes and cellular mechanisms often enables extrapolation of observations from flies to humans. There is a direct homology between *Drosophila* genes and genes that affect human diseases. Of all genes known to affect human diseases, more than 60% have *Drosophila* orthologs. More than 50% of *Drosophila* protein sequences are similar to mammalian sequences. Because of this evolutionary proximity, the fly can serve as a model organism to study human disorders like alcoholism, sleep disorders, and neurodegenerative diseases. The latter includes Alzheimer's disease, Parkinson's disease, and Huntington's disease. In each instance genomic approaches can be used to explain disease mechanisms at the genetic level. The large populations needed for such experiments are readily available when using *Drosophila* [47].

Cracking courtship behaviour

Using the imaging technique on *Drosophila* as described in the next section (Section 2.3), scientists have identified circuits related to, among others, courtship behaviour [75, 76], the olfactory system [28], visual information processing [44], and walking direction [6]. To explain the *circuit cracking process* in more detail we present work by Barry Dickson [20]. Dickson and his team at the IMP have studied neural functionality of *Drosophila*'s courtship behaviour.

The analysed neural mechanisms that govern courtship behaviour are adaptive and robust, they are accessible to genetic and physiological investigation at the level of single identifiable neurons. A fly selects specific actions on the basis of sensory input, internal physiological states, and individual experience. Upon encountering another fly, a male decides whether or not to court. The courtship ritual may take only a few minutes. The male fly is able to discriminate females from males, discriminating receptive from unreceptive females is partly learned behaviour. Depending on detected pheromones, the fly decides to initiate courtship.

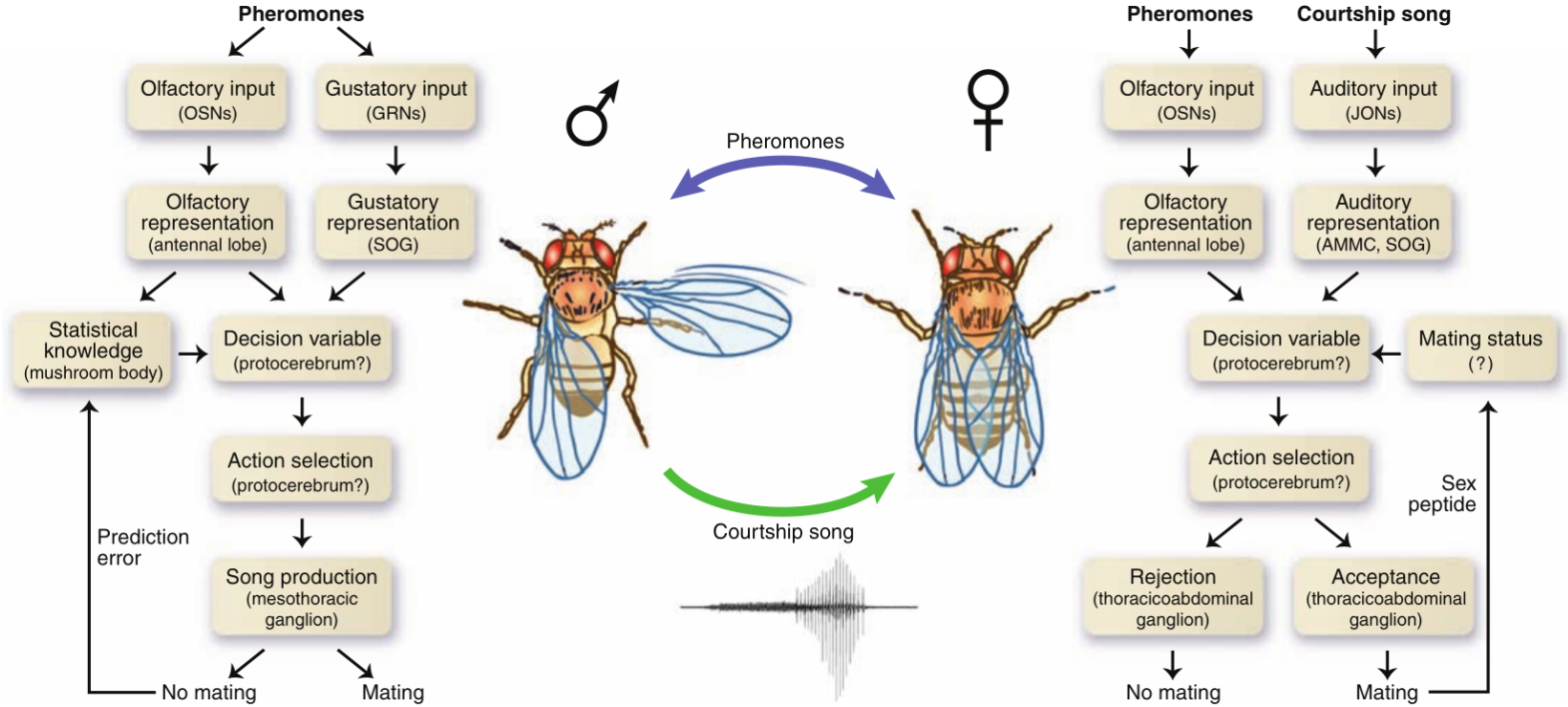


Figure 2.2: Mating decisions of *Drosophila* as presented by Dickson [20]. The male decides whether or not to initiate courtship depending on perceived female pheromones and past experience. The female then decides whether to accept or reject the male based on (mostly) the courtship song, male pheromones and current mating status. Terms in parentheses indicate the relevant neurons or brain regions.

The eligible female assesses her suitor's quality and her own readiness to mate. She judges the male mostly based on his courtship song and partially also on detected pheromones. Circuits in the female brain integrate sensory input from the olfactory system, auditory system, and reproductive tract to make her decision. The mating decision as shown in Figure 2.2 is a relatively simple and genetically tractable system. It provides an excellent model to study the brain's decision making. The neurobiologists investigate how neural circuits process and store all this information and hereby guide behaviour. As depicted they trace the neural pathways that mediate this complex behaviour from sensory input to motor output. To define the neural circuits that govern male courtship behaviour they employ genetic approaches. These make it possible to identify, characterize, and manipulate individual circuit elements. The process culminates in established relationships linking cellular biochemistry, circuit function, and behaviour [20].

The Michael Jackson Fly

Another success story of this kind of research is the work by Bidaye et al. [6] on *Drosophila* walking direction. They investigated the brain's decision making process—specifically in the locomotor circuit—when animals choose to change walking direction. Using the powerful genetic toolkit, Bidaye et al. created a phenotype dubbed *moonwalker* that walks backward instead of forward. With subsequent experiments they identified a pair of neurons responsible for this behaviour. Promising results come from a follow up paper that supports these findings with data from functional imaging [60].

As Mann [48] puts it, Bidaye et al.'s *'findings provide the first glimpse into how flies control walking direction.* [...] *we need a better understanding of the circuitry that controls forward walking. Answers will no doubt come from a wide variety of approaches, including ones similar to those used by Bidaye et al., to provide cellular resolution to complex motor outputs such as walking.'*

Researching decision making in *Drosophila*, be it on courtship behaviour, motor functions, or other systems, has the potential to completely solve a neural circuit. It may reveal fundamental mechanisms of action selection [20].

Neuroanatomy

The central nervous system of *Drosophila* consists of the brain and the ventral nerve cord (VNC), see Figure 2.1 as mentioned above. All images considered for this thesis are from the brain, the VNC is out of scope. The brain is partitioned into 43 *neuropils*. These are functional brain regions *'that synergistically* [sic] *cooperate to achieve computational tasks'* [30], i.e. they are associated with performing or being relevant for specific tasks. This may make it useful to determine the exact location of an anatomical connection between neurons. Each neuron is made up of a single cell body and one

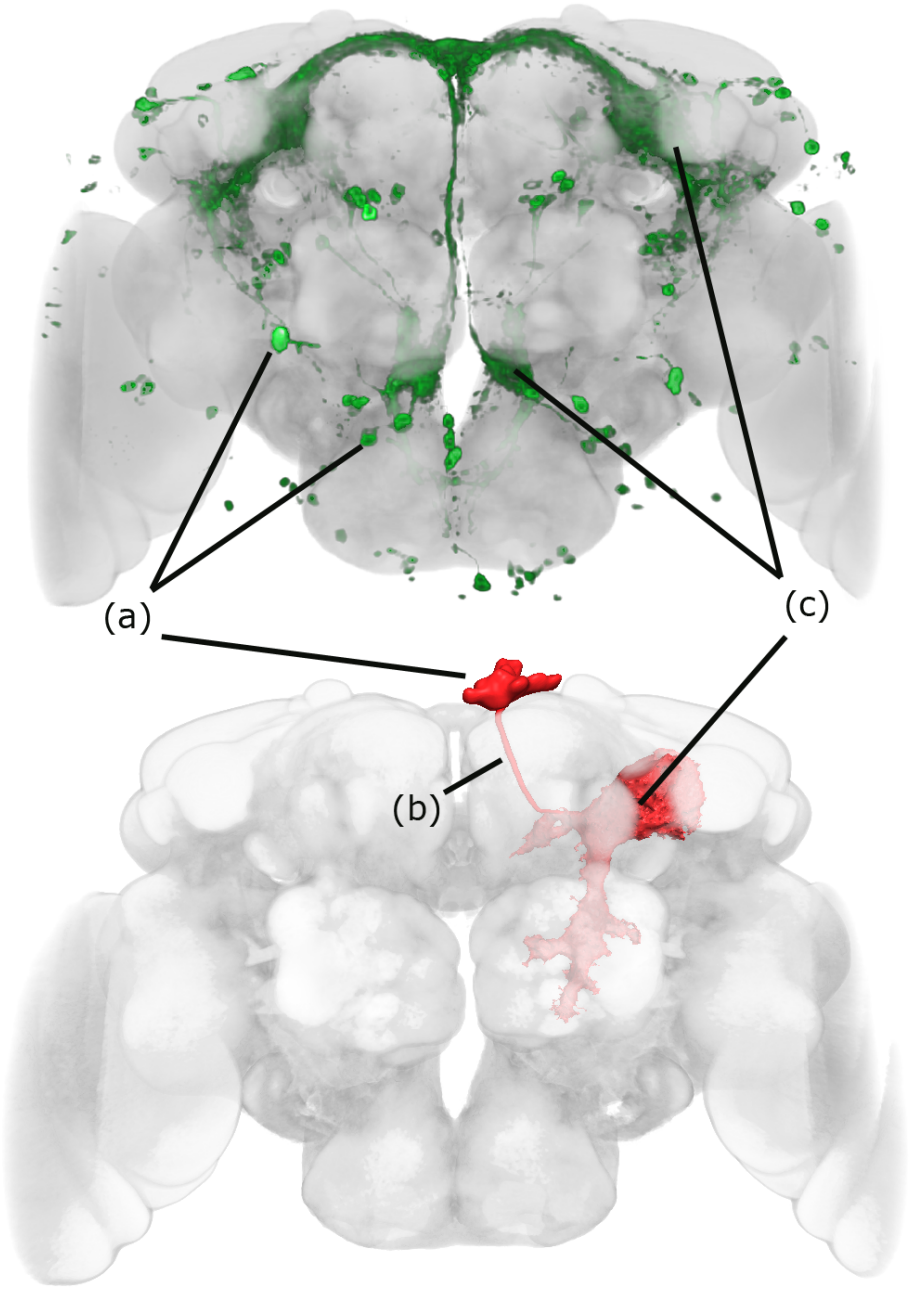


Figure 2.3: Two renderings of the standard brain in grey. Top: A confocal microscopy image using the GAL4 system, the green fluorescent protein highlights specific neurons. Bottom: A single segmented neuron in red. *Cell bodies (a)* are clearly visible as bright green dots (top), and as a large red structure (bottom), representing the average over multiple images of the same neuron. Synapses are located at the *arborisations (c)*. The neuron is completed by the *projection (b)* that connects all its arborisations to the cell body. These thin nerve fibres are barely visible under the microscope. Both images were created with BrainGazer [11].

or more arborisations, connected by a projection (compare Figure 2.3). The *cell body* contains the cell's nucleus. Synapses are located at *arborisations*, the terminal branchings of nerve fibres where communication with other neurons may occur. In *Drosophila* and most other invertebrates the cell body is located at the outer regions of the brain. The arborisations inside it are linked to their cell bodies via thin nerve fibres, the *projections*. Synapses, i.e. connections between two overlapping arborisations, can only exists between pre- and post-synaptic terminals [7]. Pre-synaptic terminals (also called axon terminals, synaptic boutons, or terminal boutons) are the transmitting/sending part of a synapse. The opposite part belonging to a different neuron can receive a signal, this is the post-synaptic terminal (also called post-synaptic receptor). Our data however does not discriminate pre- and post-synaptic arbours.

Potential Connectivity and Peters' Rule

Cracking neural circuits requires us to make assumptions about connectivity. Functional connectivity between neurons requires overlap of axons and dendrites, the area where synapses are located. Shape and location of these arborisations are an important source of specificity. For excitatory neurons the consensus states that where axons and dendrites are sufficiently close, synaptic connections occur [23, 62, 72].

In case it is infeasible to investigate neurons down to the synaptic level due to technological and/or cost constraints, neurobiologists may employ *Peters' Rule* [12,54]. The rule makes an assertion about the anatomical strength of a connection. From the shape and location of an overlap of arborisations, potential connectivity can be inferred. In simple terms, Peters' Rule states that overlap between arborisations of neurons is necessary but not sufficient for direct functional connectivity. If there is no overlap there is no connectivity at all. If there is an anatomical connection, it implies a potential connectivity of a certain strength [12, 23, 62]. A high degree of overlap predicts synaptic connectivity.

2.3 Data Acquisition and Enhancement

These processes are explained in detail in Yu et al. [76]. The following summary will suffice to give a good understanding of the nature of the data.

The digital atlas of the fruit fly neurons was created by non-rigidly registering confocal images onto a common template, a standard brain. nc82 staining was used to visualise brain tissue, providing a spatial relation for the image registration. The standard brain was generated from a carefully selected set of tissue images.

Specific neurons where labelled with the GAL4/UAS system. This biochemical method consists of two parts. Two lines of flies are separately modified – a line is a genetically identical group, i.e. the flies have the same genotype. The *driver line*

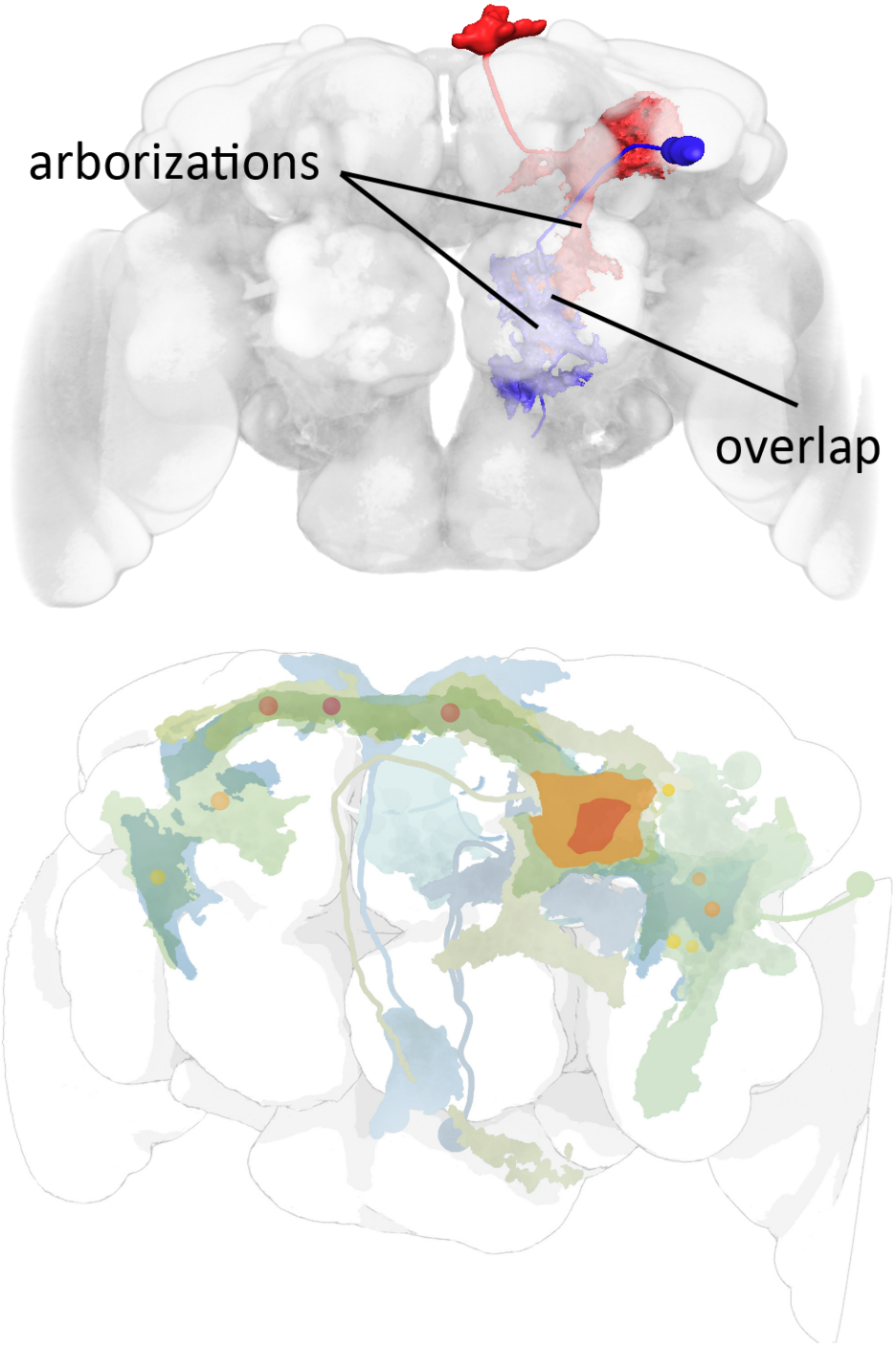


Figure 2.4: Top: two neurons are visualised, one in red and one in blue (image created with BrainGazer [11]). Their arborisations produce an overlap, its exact size and form are difficult to investigate in 3D. Bottom: a mock-up from the final design by Judith Moosburner [50], here overlaps are visualised as flat unicolour blobs.

receives the GAL4 gene, the *responder line* the UAS sequence. Their children have the GAL4 protein activated in the targeted neurons, creating a fluorescence effect. The 3D double channel confocal microscopy images contain brain tissue in one channel (as a reference) and the highlighted neurons in the other channel.

The confocal microscopy is done on the dissected brain of the fruit fly. The volumetric image has a size of $420\,\mu m \times 420\,\mu m \times 165\,\mu m$. Images were taken at 768×768 pixels with 165 slices, resulting in a resolution better than $1\,\mu m^3$. Although this is quite a high resolution, synapses are still smaller than the diffraction limit. With specialized labelling methods [77], synaptic information could be acquired to specify cells as pre- or post-synaptic. Our digital atlas does not, however, contain such information.

After the registration of the raw images, neuronal structures were semi-automatically segmented with support of the Amira software [5]. Cell body, projection, and arborisations were segmented on single images or GAL4 averaged images. The segmented neurons are stored in a relational database, along with meta data on sex, driver and reporter line, neuron type, etc. The segmented data is available as both binary 3D mask and surface geometry. Cell body and arborisation are separately stored as triangle mesh data while projections are represented as centre lines with varying radii.

The database includes limited overlap information. Volumes for pairwise arborisation overlaps are calculated from the binary masks and stored in the database. Also, the overlap's distribution to the neuropils is calculated and stored. The data is available via the public database BrainBase [10].

2.4 Scientific Questions

In their search for neural circuits the scientists analyse the available data in BrainBase using different tools. Among them are BrainGazer [11] and its integrated graph visualisation software neuroMap [65]. These will be discussed in the next chapter.

With the available data neuroscientists can not directly explore functional connectivity but potential connectivity instead. By employing Peters' Rule the experts can investigate overlaps of arborisations and, from their location, size, and shape, formulate new hypotheses. The neuroscientists are interested in getting quick answers to the three **core questions**

- **Which groups of neurons overlap?**
- **Where do they overlap?**
- **What is the significance of the overlap?**

Volumes of arborisations are measured in μm^3 and so are overlaps, which are intersections of arborisation meshes. The scientists, however, judge the importance of an

overlap not solely on absolute volumes, but on volume ratios. The essential numeric overlap values are the ratios between the volume of the intersection and the respective volumes of the participating arborisations. According to Peters' Rule, the largest of these ratios, displayed in percentages, is likely to be most interesting to the user. Supported by these ratios and extensive domain knowledge, the scientists can judge an overlap's significance.

Answering the three questions above becomes more and more complex as the number of arborisations contributing to an overlap grows. The idea that the combination of a good information and interaction design with online computation of overlaps can substantially alleviate this analytical process, provides the guideline for this thesis.

CHAPTER 3

Related Work

After briefly touching on the rendering techniques, this chapter compares a multitude of tools for the visualisation of connectivity. The relevant state-of-the-art methods are summarised and followed by a short discussion of what they mean for the development of our tool.

3.1 Rendering Techniques

Our developed methods are influenced by works from several fields. We combine several non-photorealistic rendering (NPR) techniques to convey a Focus+Context metaphor. These techniques meet the requirement of scientific visualisation, to communicate information efficiently, without adhering to realism [51]. In this sense, we employ silhouettes and watercolour rendering. Silhouettes are commonly used for abstraction or to improve recognition of objects in medical imaging [22, 57, 70] and other scientific visualisations [46], while watercolour techniques are mostly reserved for artistic work. We have carefully adapted these techniques to closely imitate the design created by Judith Moosburner (compare Chapter 4).

For some features, such as overlap rendering, we employ A-buffers. This technique, typically used to achieve order-independent transparency, is essential for this thesis. Section 5.3 details the data structure and discusses the relevant related work. Chapter 6 describes how we use the A-buffer as a rendering technique and for volume computation.

3.2 Visualisation of Neural Connectivity

There exists a wide range of tools today to visualise different aspects of neural connectivity. The many diverse sources of data produce quite diverse visualisations. Many data sources differ substantially, as there is no consensus on what data should be included in a database [16]. Several data collections related to neurocircuit research are publicly available. Such *brain atlases* exist for various species at various levels (even down to the synaptic level) and have spawned a multitude of tools for the exploration of images and annotated neurons [10, 15, 32, 49, 64, 71]. As circuit neuroscience is a relatively young field, its imaging techniques for acquiring connectivity information are still improving. These atlases may collate images of different scales, from the macro scale (connectivity between neuropils) down to the micro scale (synapse information), some also annotate neurons.

Atlases are essential for research and education. They illustrate and annotate organs and bring coherency by defining a nomenclature. Flylight [32], Flybrain [64], and Flybase [71] provide only confocal microscopic images, but no annotated neurons. They offer typical atlases describing the *Drosophila* brain and interactive navigation in their respective databases of images and sketches.

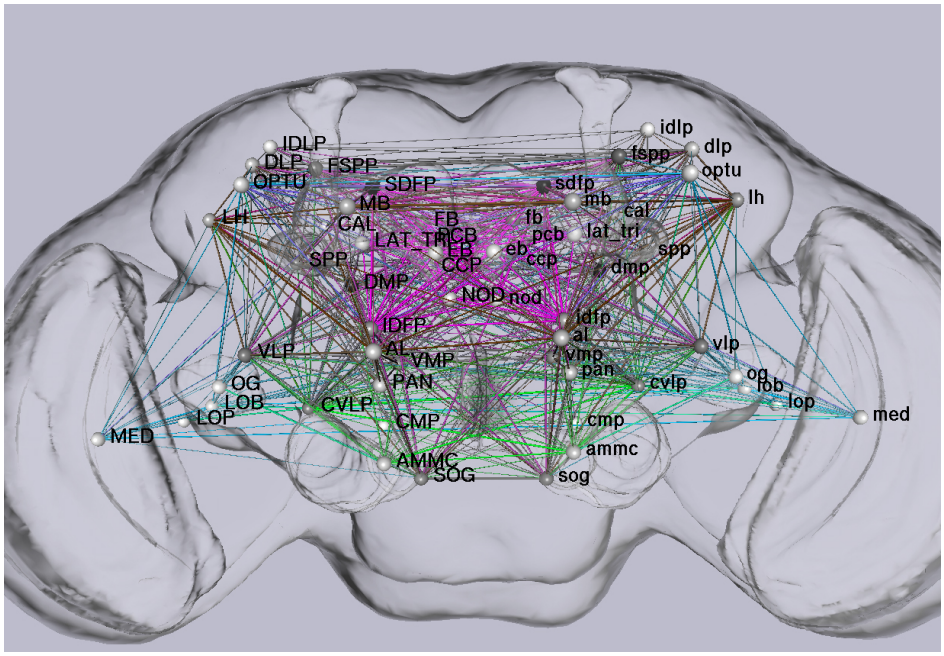
The online portals FlyCircuit [15] and Virtual Fly Brain [49] provide access to public databases as well. With graphically driven ontology queries, they offer a simple tool for finding neuron overlaps in *Drosophila*. FlyCircuit's static wiring diagram displays connectivity between neuropils. Neuropils can be interactively selected to filter connections (Figure 3.1). A web service also offers a tract finding feature for neural tracts that connect neuropils to each other.

Although FlyCircuit and Virtual Fly Brain support interactive searches for connectivity, they are both limited to connectivity information between neuropils. They exclusively find neurons projecting to the same neuropil and do not detect arborisation overlap, i.e. their overlap information exists on the brain region level and is thus far coarser than direct overlap computations.

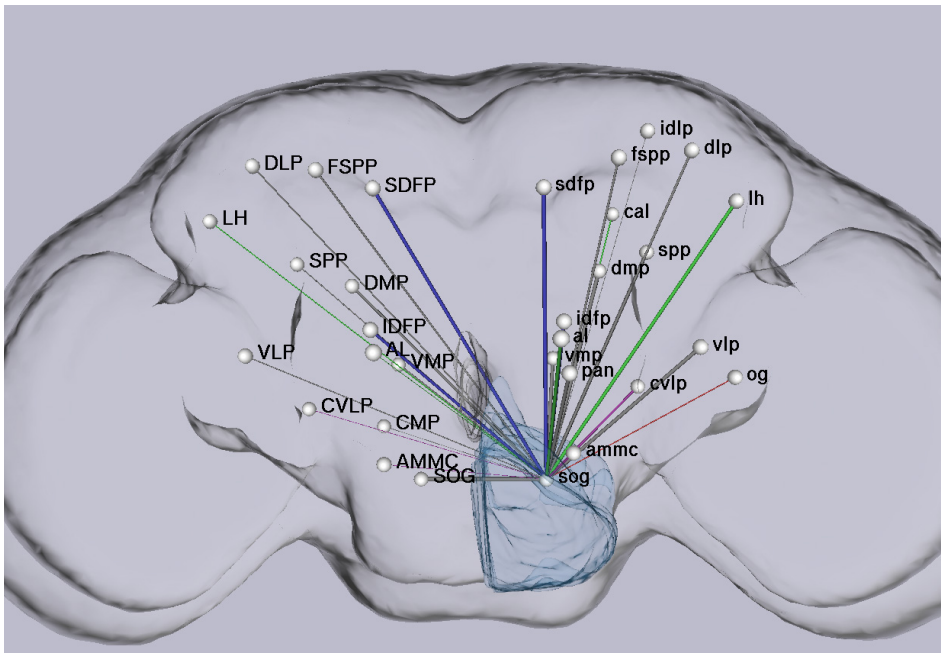
Neuron Navigator [45] is a visual query tool that operates on one of these atlases. It helps explore FlyCircuit in 3D for potential connectivities with an array of visual queries (Figure 3.2). Neurons are drawn as complete tracing lines, sacrificing volumetric information for an un-occluded visualisation. As spatial context the brain's mushroom body—made up of all neuropils—is rendered in black, optionally opaquely or transparently.

The database BrainBase [10] provides precomputed overlap information for pairwise overlaps. It also offers a parallel coordinates based neuron search that queries neuron-neuropil overlaps. Like FlyCircuit and Virtual Fly Brain, it offers 3D online rendering of neurons on demand, allowing users a visual inspection of the results.

BrainGazer [11] operates on the data from BrainBase [10], another database like FlyCircuit and Virtual Fly Brain. Like Neuron Navigator it provides high quality 3D



(a)



(b)

Figure 3.1: The online portal of FlyCircuit [15] shows connections between neuropils. The static wiring diagram (a) draws straight lines between neuropils: line colours indicate functional modules, line thickness encodes the number of connections. (b) A single neuropil has been selected. All its connections to other neuropils are drawn.

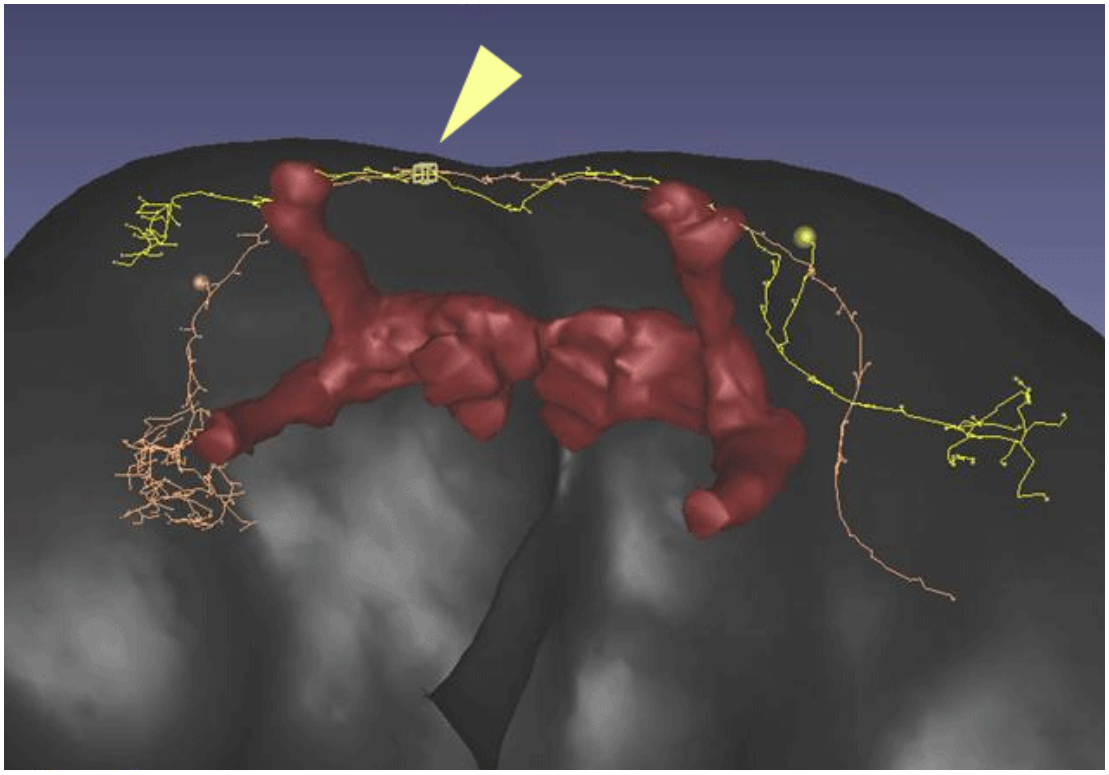
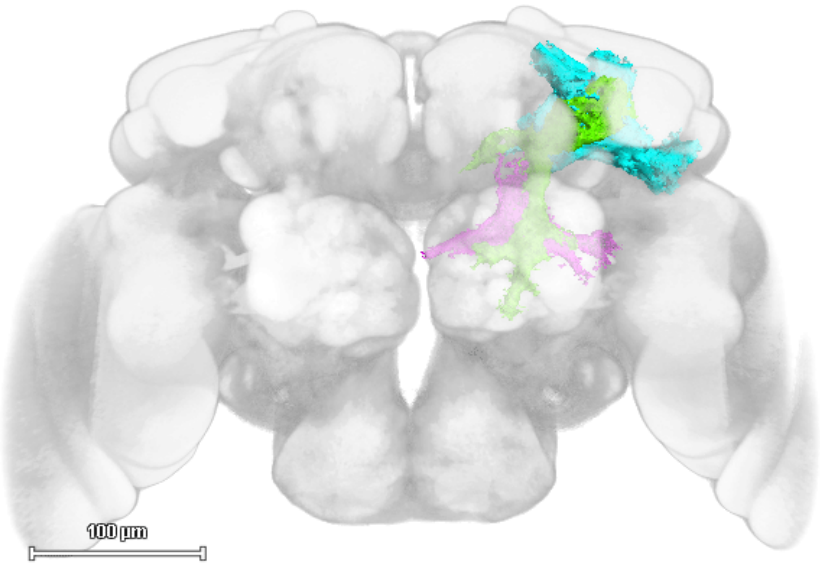


Figure 3.2: Neuron Navigator [45] is a 3D interaction tool for FlyCircuit [15], which includes multiple visual query options. In this image the user selects a region of interest (small yellow box). Only neurons passing through it are rendered. They are drawn as tracing lines to avoid occlusion.

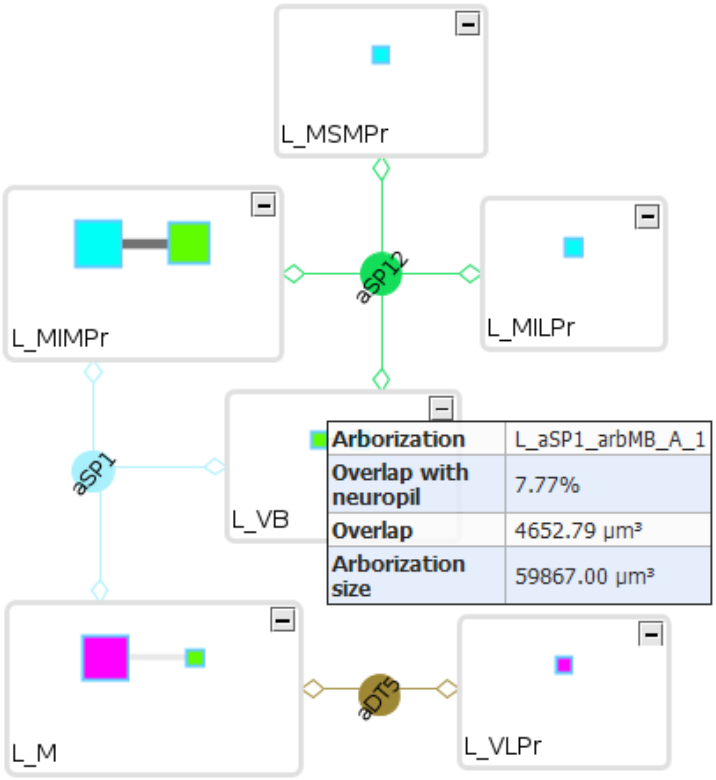
visualisations and sophisticated spatial queries. Together with neuroMap [65], it can be used to explore and analyse a potential connectivity between pairs of neurons. Precomputed volume information derived from pairwise arborisation overlaps is visualised to help judge the significance of overlaps (Figure 3.3).

Jianu et al. [33] developed an abstract representation of axonal tracts in the human brain (Figure 3.4). The interactive exploration system visualises tractography datasets as two-dimensional paths. The useful colourisation of axon bundles was indeed found by trial-and-error.

Li et al. [43] implemented a tool for the quantitative analysis of brain connectivity (Figure 3.5). It is based on defining regions of interest (ROIs) to create connectivity graphs. The tool differentiates functional connectivity networks and effective connectivity networks and works with multi-modal neuro-imaging data. ROIs are represented as spheres and connectivity strength between them is intuitively indicated by the thickness and opacity of the edges. The resulting 3D graph structures preserve the spatial



(a)



(b)

Figure 3.3: BrainGazer [11] offers multiple query and visualisation tools for the exploration of neurons in *Drosophila*. (a) Three arborisations are rendered. (b) The same data can be investigated for overlaps using neuroMap [65]. It includes quantitative volume information for overlaps.

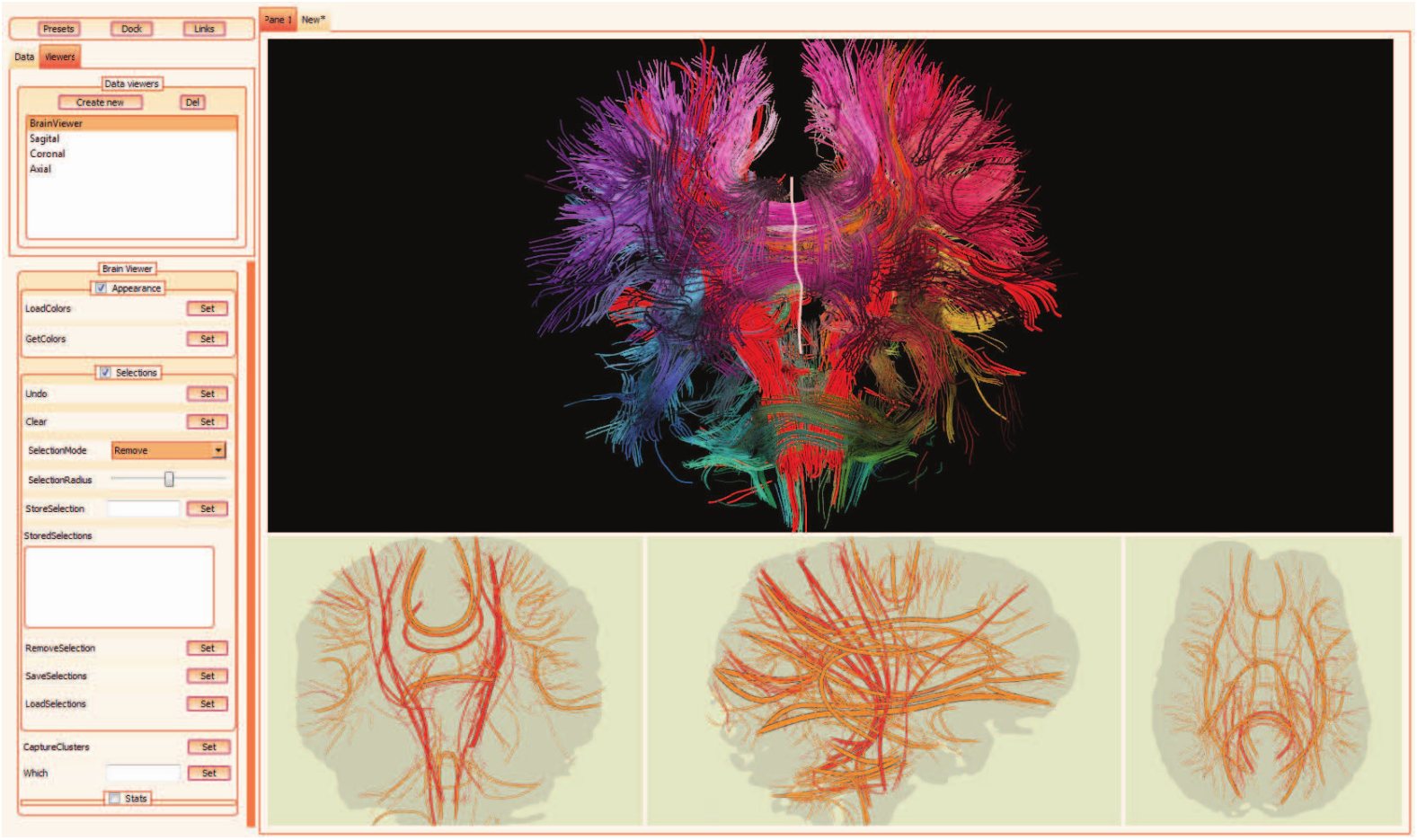


Figure 3.4: An interactive exploration system for axonal tracts in the human brain, developed by Jianu et al. [33]. The linked views of the brain in 2D and 3D support selections by mouse interaction.

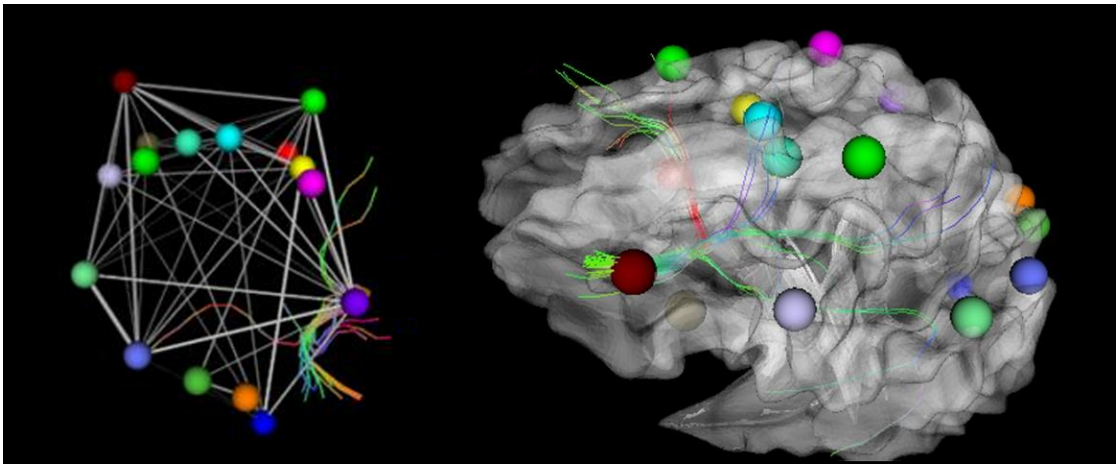


Figure 3.5: The tool by Li et al. [43] visualises connectivity as a 3D graph. The spheres are regions of interest interactively defined by the user.

context.

NeuroLines [2] abstracts mouse brains at the synaptic level into a subway map. It hierarchically visualises the connectivity between dendrites and axons. The mouse brain data is acquired by scanning slices via electron microscopy. This analysis tool is designed for scalability and tries to keep some spatial context, as the subway map can preserve relative distances.

The tool UpSet [41] uses a 2D visualisation for the quantitative analysis of sets, their intersections, and aggregates of intersections. Sets are visualised in a matrix structure. It is very abstract and focuses on scalability. Similarly, Radial Sets [3] visualises intersections of sets in radial graphs. In general, visualising sets often necessitates visualising their intersections. In these tools it is almost always done in 2D without preserving spatial context [4].

A specific example for *Drosophila* is the previously mentioned neuroMap [65]. In conjunction with BrainGazer it visualises intersections as 2D graphs. neuroMap provides an optional anatomical layout to keep some spatial context. Its abstract view explores potential connectivity with a highly sophisticated encoding of pre-computed overlap information of pairs of arborisations. Nevertheless, higher order overlaps are difficult to detect here too, and no quantitative information is available either.

Recently a tool supporting structural connectivity analysis of a model of neuron populations in the barrel cortex has been proposed by Dercksen et al. [19]. Here a hypothesis on potential connectivity is derived from the distribution of pre- and post-synaptic sites groups of neurons.

3.3 Occlusion and Intersection in Visualisation

Our work centres around visualising intersections of objects. The major obstacle in this regard is certainly occlusion. Various techniques tackle the problem of occlusion in visualisation as made evident by Elmqvist and Tsigas [26].

Using their classification on our work reveals that it fits in quite well with the *virtual x-ray techniques* but breaks this pattern when it comes to multiple (integrated) views. Nevertheless techniques following this pattern provide valuable input when approaching the problem of visualising object intersection information. Classically they employ cut-outs or cutaways [17, 27, 74] and rely heavily on transparency to reduce occlusion [21, 25]. Visualising intersections however is not the focus of these techniques.

3.4 Discussion

Connectivity information may be presented as a connectivity matrix. Its binary or weighted versions can be visualised as a heatmap or a graph, with or without spatial context. When considering set intersections of higher order, these visualisations fall short. As Dercksen et al. [19] put it, creating easily interpretable visualisations of brain networks is non-trivial, as such a network is often a complex graph embedded in 3D.

Concerning the occlusion management discussed by Elmqvist and Tsigas [26] our work is unique, occlusion is integral to our visualisation as we concentrate on object intersections. Also unlike many visualisations dealing with heavy occlusion we use (virtually) no transparency.

BrainBase [10] and BrainGazer [11] are the base of our new tool. This pairing was obvious as our clients were already using these systems. A benefit is certainly the synergy with other tools implemented in the same system, see Section 5.1 for details. The new visualisation could however be integrated with other tools and databases mentioned above as well.

The current implementation of our tool may benefit in the future from linking it to a customised version of UpSet [41] or even Radial Sets [3]. These 2D visualisations could complement our system by providing additional views on the same data for different purposes.

Some of the referenced tools operate on quite different data sources such as fMRI or electron microscopy [2, 33, 43]. This makes some of their contributions not easily adaptable to our system. Nevertheless they provide inspiration for the design and implementation.

From its intention the work by Dercksen et al. [19] comes closest to ours. It also combines 3D visualisation with quantitative and qualitative elements, but the underlying data and therefore also the methodology differs substantially from our solution. The visualisation in 3D of higher order overlaps is still unique to our system.

CHAPTER 4

Information and Interaction Design

This chapter outlines the design work by Judith Moosburner [50], which laid the foundation to this thesis. Judith Moosburner, a graphics designer from the *Zürcher Hochschule der Künste*, worked in cooperation with biologists from the Institute of Molecular Pathology (IMP) in Vienna to create multiple artistic design studies for the visualisation of higher order arborisations. The work, published as a Diploma Thesis in 2011, includes a final design proposal—the guideline of this thesis' project.

4.1 Background

The goal of the design work was to increase and make accessible knowledge about neural connectivity. To achieve this task the designer drew inspiration from different fields, e.g., cartography and medical illustration, always keeping in mind and consulting the target audience—the neurobiologists at the IMP.

Conservative hand-drawn visualisations of dendrites, namely those of Camillo Golgi (1843-1926) and Santiago Ramón y Cajal (1852-1934), were a starting point for the design (Figures 4.1 and 4.2). From 1920 on, initially monochromatic illustrations were increasingly replaced by more and more colourful images, conveying more complex content. Technical advances in microscopy replaced hand drawings by high-resolution photographs, which were superseded by 3D images.

There already exist various ways to scientifically visualise neurons or neural connectivity (see Chapter 3). The involved colouring schemes are central to these visualisations and are often heavily influenced by the actual appearance of neurons in light microscopy. For example in the GAL4/UAS system (compare Section 2.3) the protein markers produce a green fluorescence when scanning under the influence of laser light.

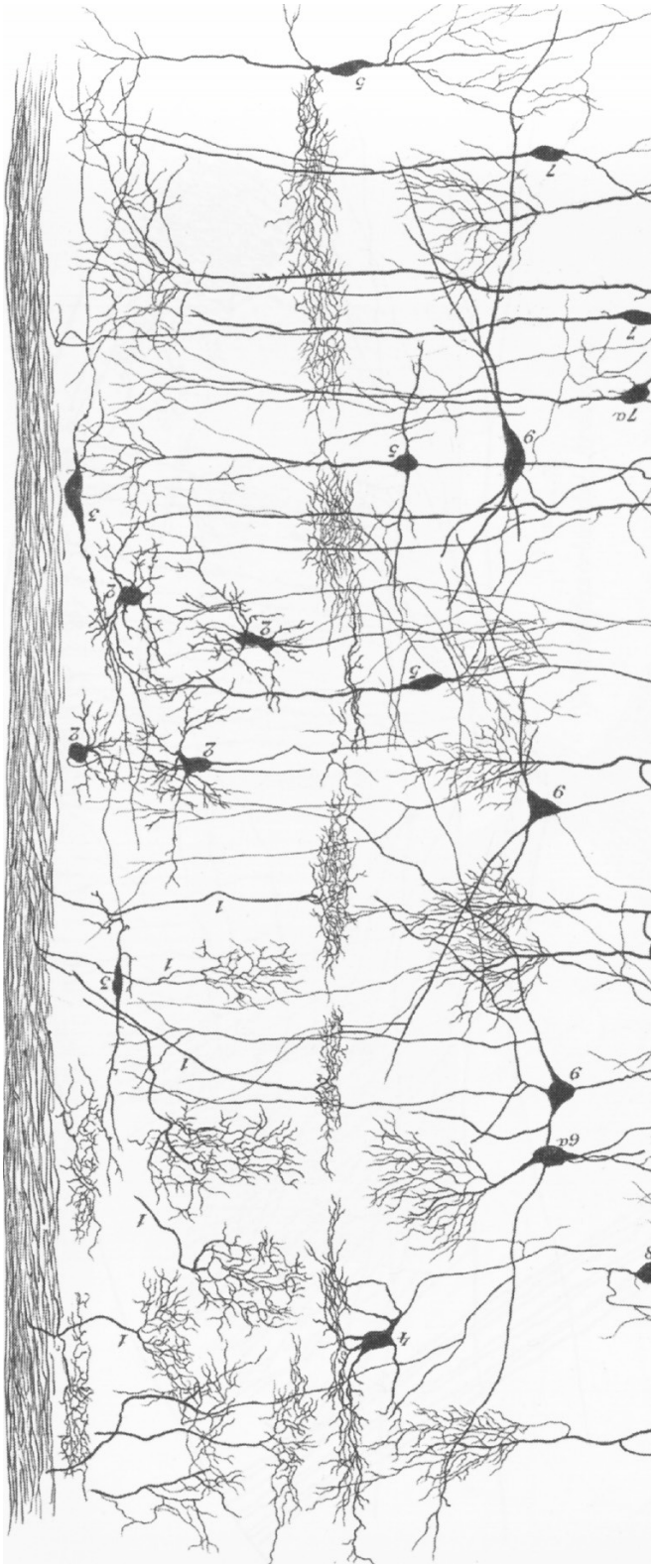


Figure 4.1: Drawing of a bird's optic lobe by Santiago Ramón y Cajal [55]

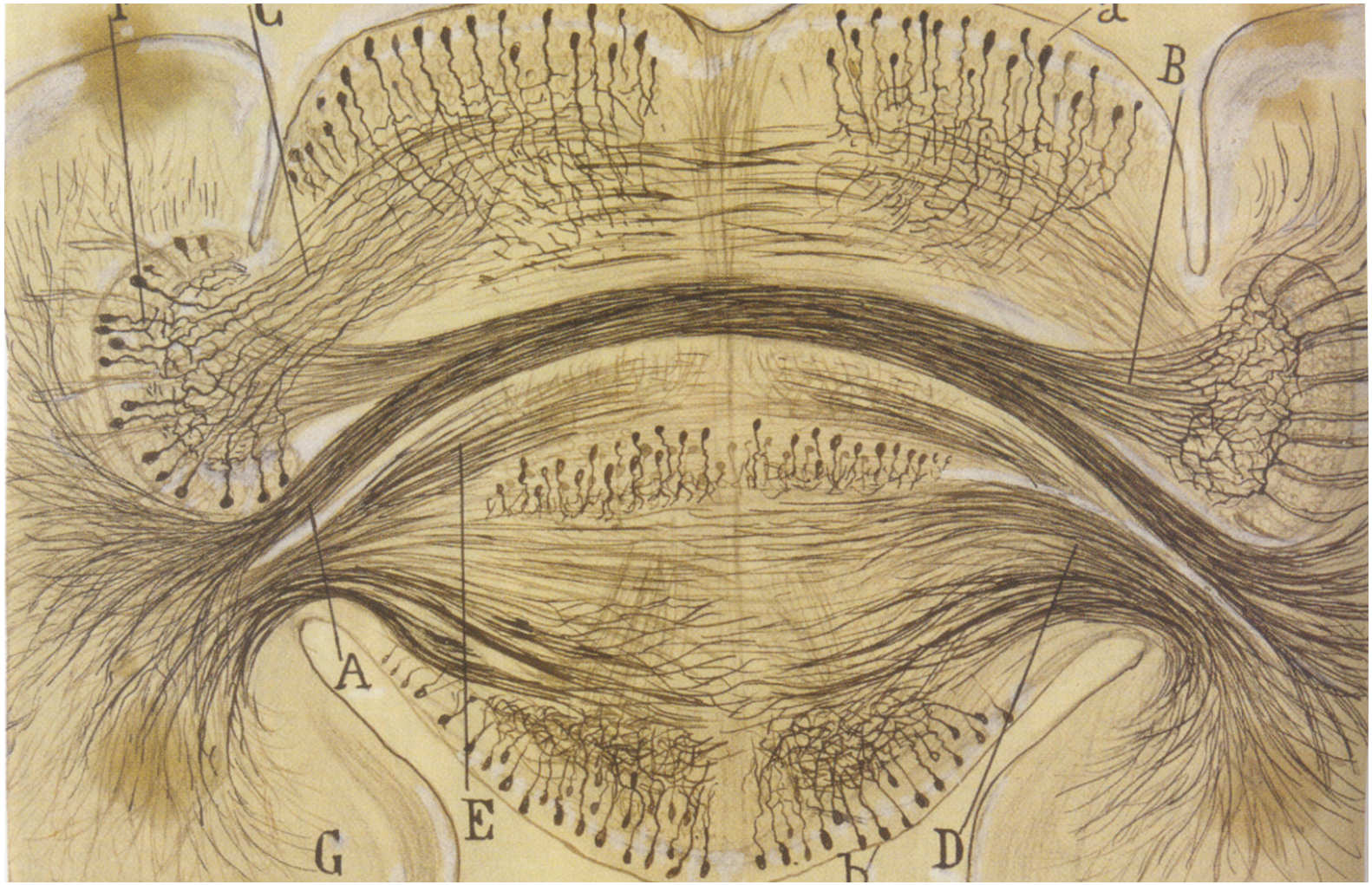


Figure 4.2: Drawing of a cuttlefish's cerebral ganglion by Santiago Ramón y Cajal [56]

The designers goal is always to create aesthetic images while conveying all necessary information. In this sense abstraction from the real world may become desirable.

4.2 Creating the new Design

Moosburner developed the new design in a close feedback loop with the domain experts, always guided by the question *'How can colour design and interaction support the visualisation of neuronal connectivity?'*, starting from the 'classical' visualisation BrainGazer offers for the available data (compare Figure 2.3 and Figure 2.4). The goal was to produce a design for a tool that would support the scientists in answering their questions listed in Section 2.4. The focus was on optimizing perceptional aspects of the information flow while technical considerations or limitations were not central in the design.

The brain atlas's data are registered onto a standard brain template made up from brain tissue as seen in Figure 2.3. This template includes a partitioning into about 50 neuropils, serving as a spatial context. Onto this template neurons and their arborisations are registered, providing a second layer of spatial context for the overlaps. The complete brain atlas includes quantitative information on potential connectivities and additional meta-data. The view on the brain template resembles cartographic systems [13], thus the design work takes inspiration from concepts of cartography and information design.

Moosburner drew inspiration from multiple sources to tackle the design problem. In the context of insights from perception and colour theory [31, 36], interaction design [66] and cartography [1, 13], current 3D depictions of neurons (compare Chapter 3) suffer from substantial flaws. They are plagued by too much detail, visual clutter, adverse colouring, and missing representation of connectivity. To overcome these issues, the new design was based on the main principles of information design [8, 37]:

- **Reduction** to decimate dispensable information and to confine information to its essentials, especially focusing on already existing visualisations and the representation of the brain.
- **Abstraction** to make connections visible, to identify clusters and core areas, and to depict connectivity in order to highlight and rank it faster and more easily.
- **Information scaling** through interactivity to reduce the amount of information while still providing access to details on demand.

In a first step these topics have been approached by the designer through several artistic studies. From figurative to more and more abstract representations (compare Figures 4.3 to 4.7), the design process culminates in the final design discussed in the

Figure 4.3: Design study as presented in the thesis by Moosburner [50].

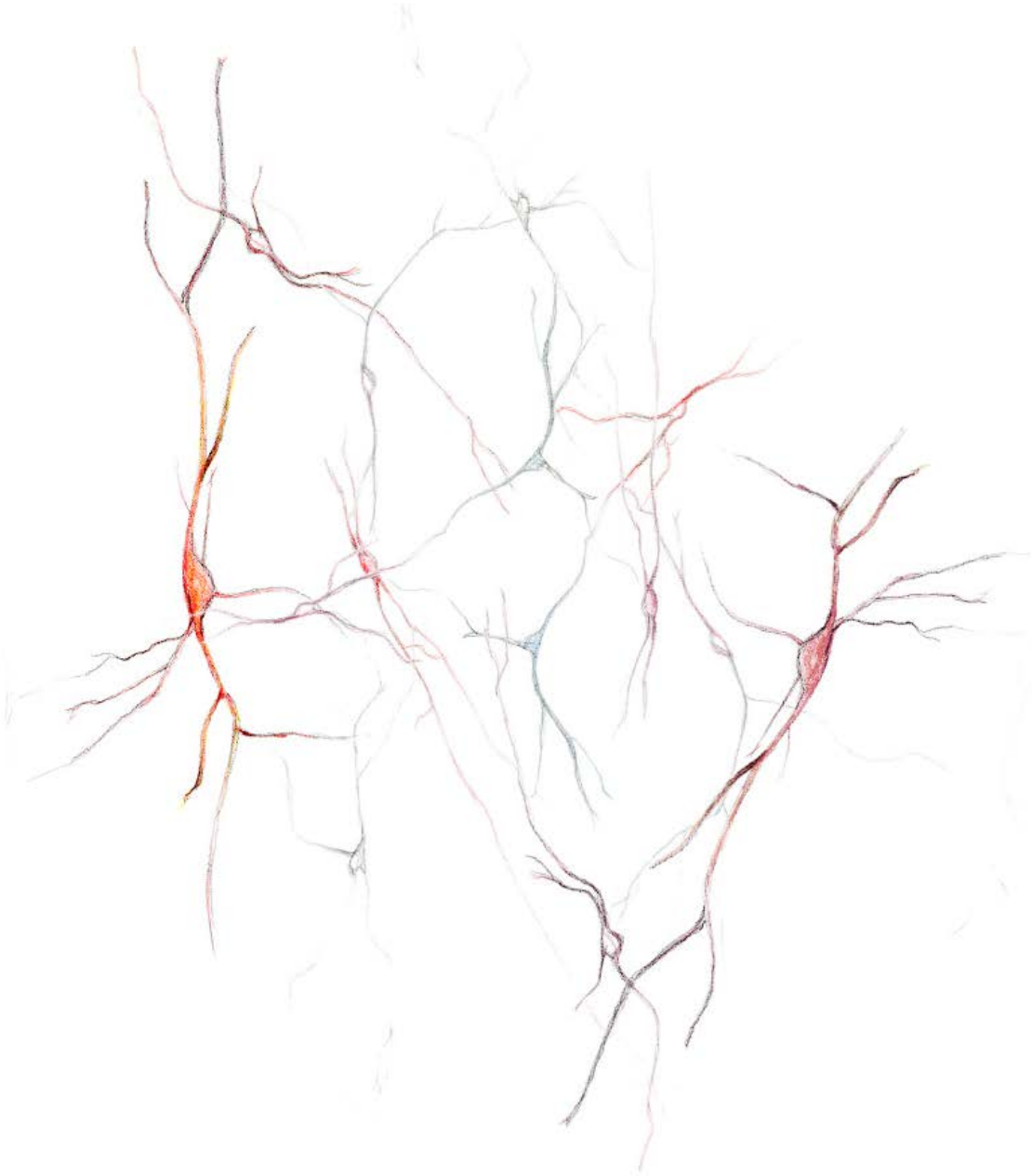


Figure 4.4: Design study as presented in the thesis by Moosburner [50].

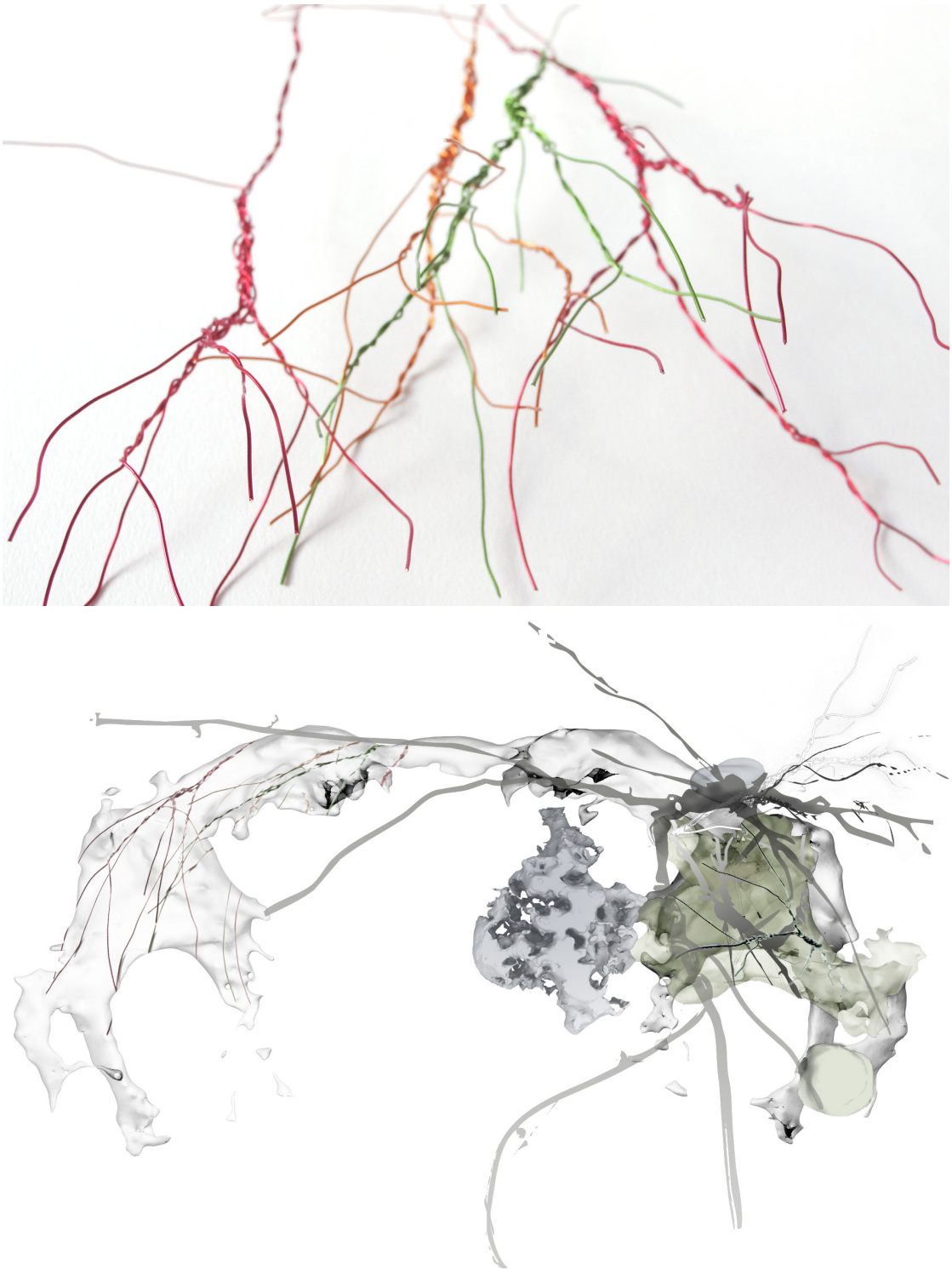


Figure 4.5: Various design studies as presented in the thesis by Moosburner [50].

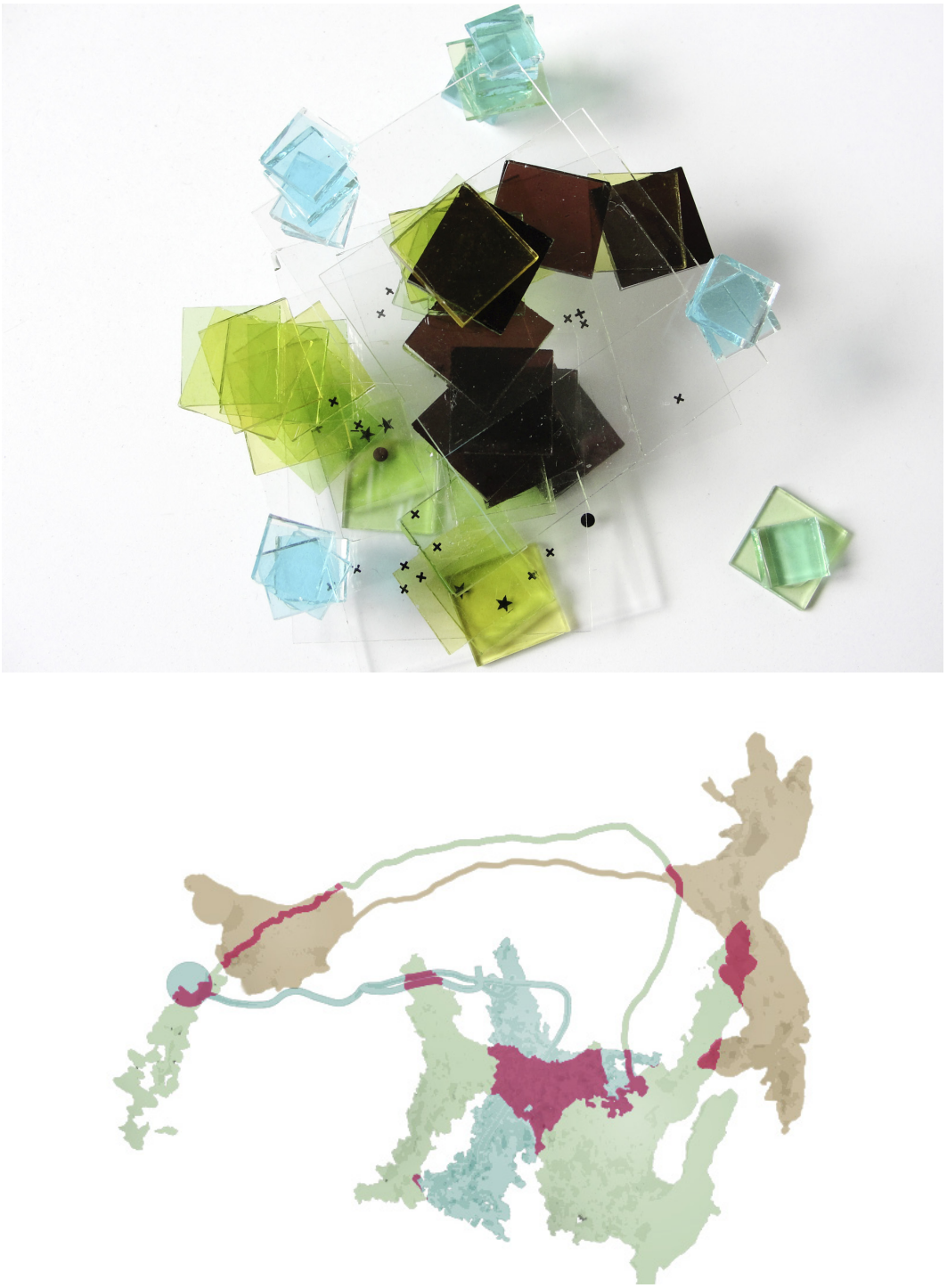


Figure 4.6: Various design studies as presented in the thesis by Moosburner [50].

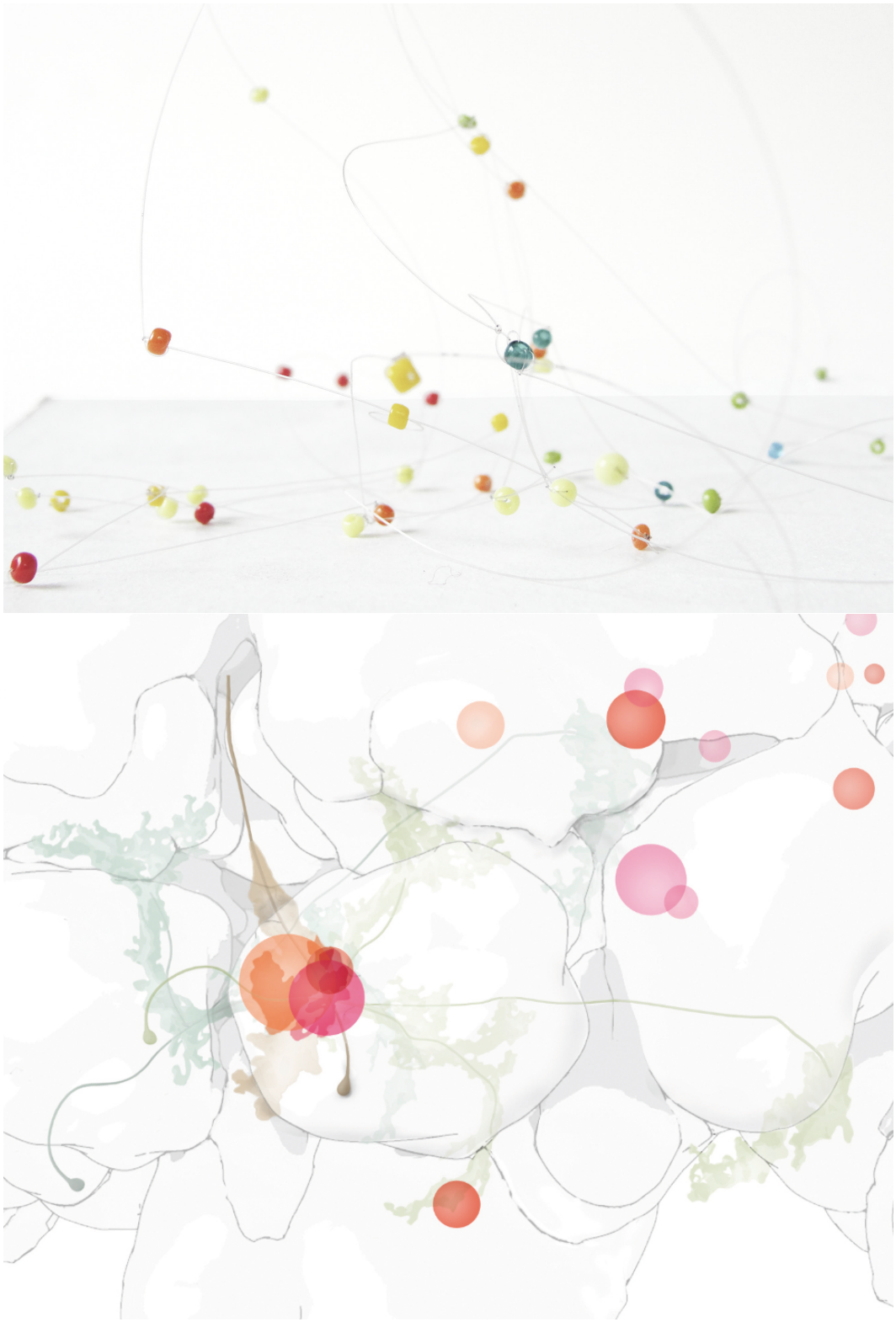


Figure 4.7: Various design studies as presented in the thesis by Moosburner [50].

following sections (compare Figures 4.8 and 4.9). Here it is made clear where along the way these main principles were applied. For an in depth look at the interactivity aspect of the information scaling we refer to Chapter 7.

4.3 Object, Shape, and Colour Design

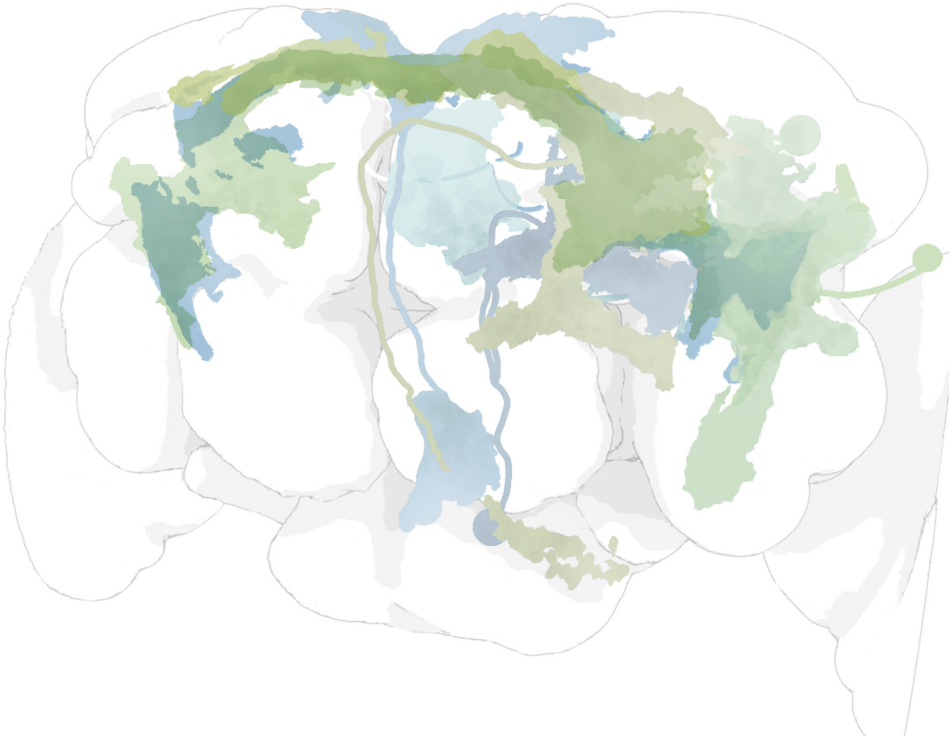
The brain tissue, its neuropils, and neurons provide hierarchical layers of context for representations of potential connectivity information. Together, these support finding answers to the scientific core questions (Section 2.4), in particular the first and second one.

The available geometric representations of neuron, neuropil, and brain surface have been directly extracted from the 3D images (compare Section 2.3) resulting in seemingly well-textured object surfaces. The first layer of spatial context—the brain and its neuropils—has its surfaces widely reduced or even omitted. As global context, Moosburner chose to represent the brain tissue with an abstraction of the brain surface, coloured in a neutral white to grey tone with high transparency and enhancing silhouettes (compare Figures 4.8b and 4.9). Neuropils are depicted similarly transparent but with a slight blue hue to establish differentiation (compare Figure 4.10a).

Neurons provide immediate context for overlaps, which are considered to be the central information for the users. The surfaces of the arborisations are particularly heterogeneous and structured, they were reduced as much as possible to prevent distraction by unnecessary details. Instead slight texturing is used to obtain a lightweight effect of organic appearance of neurons. Projections appear as thin lines and cell body locations as spheres. For neurons, Moosburner chose a colouring scheme from the cold colour spectrum of brown / green / blue—colours which are mostly observed as neutral (compare Figure 4.8b). The blueish neuropil colour was chosen from the unsaturated, pale end of the spectrum (compare Figure 4.10a).

Regions of overlap are defined by intersecting arborisations and are highlighted—in opposition to the contextual information—in an eye catching manner. As the detection of higher order overlaps is in the centre of interest, a gradient colouring system with intense colours was designed, supporting the detection of significant overlaps (compare Section 2.4, specifically the third *core question*). It colours pairwise overlaps in yellow, triple ones in orange, and higher order overlaps in red and dark red (compare Figure 4.9). The bright, vivid colours have a high signalling effect and can be easily recognised and identified by their contrast to the rest of the system. Such gradient colouring schemes are used in cartography to depict growing values (e.g. temperature) and are discussed at length in the more general field of information visualisation (e.g. for the use in heat maps).

(a)



(b)

Figure 4.8: The mock-ups are part of the final design proposal by Moosburner [50]. Neurons stored in the atlas as segmented 3D data are depicted with very reduced features (a). The highest layer of spatial context, the brain tissue, is reduced to a silhouette (b).

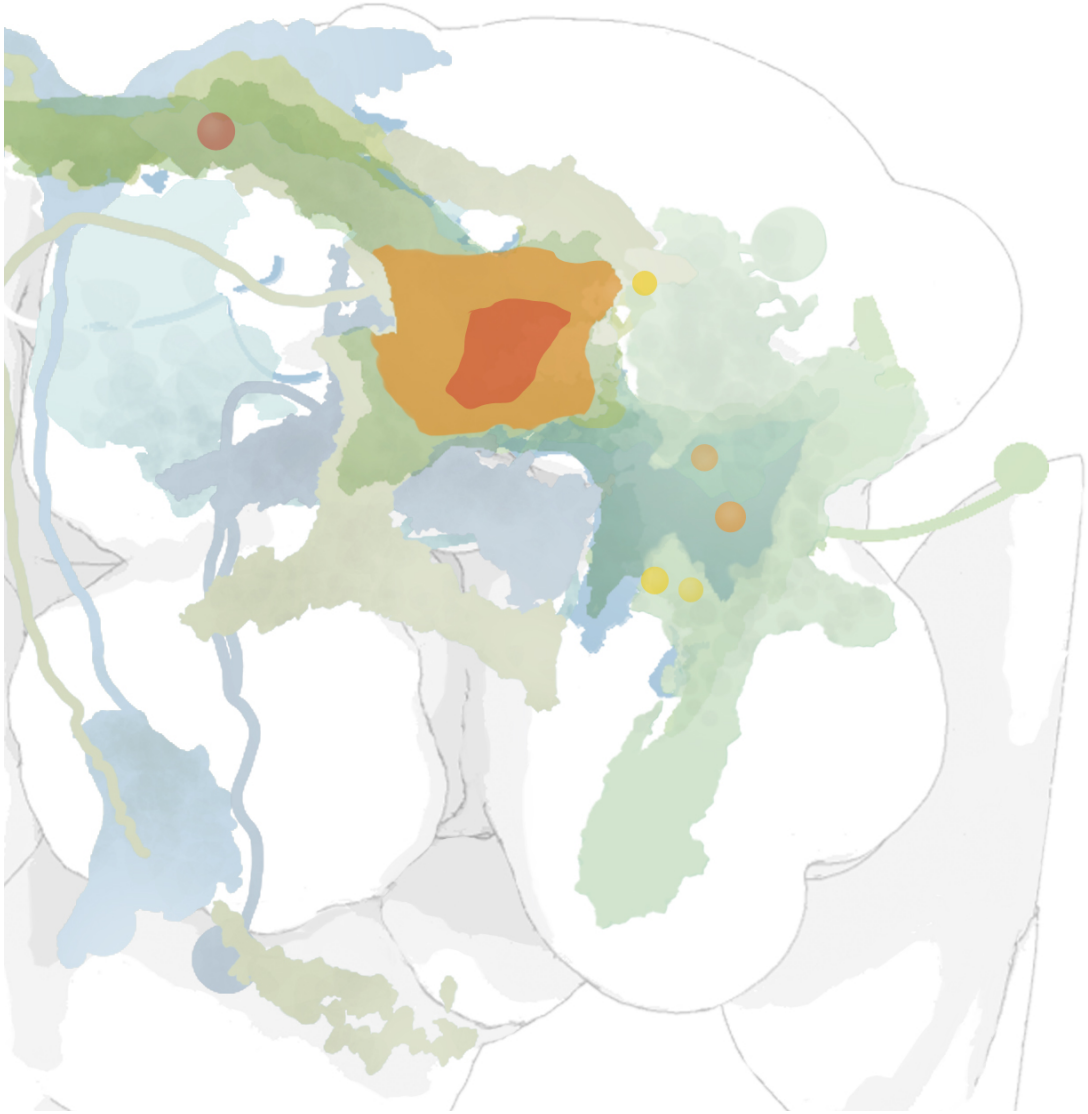


Figure 4.9: This mock-up is part of the final design proposal by Moosburner [50]. Neurons are coloured in blue, green, and brown tones. In contrast, overlaps and their abstracting glyphs are drawn in bright, vivid colours. The colour scheme encodes the order of overlap.

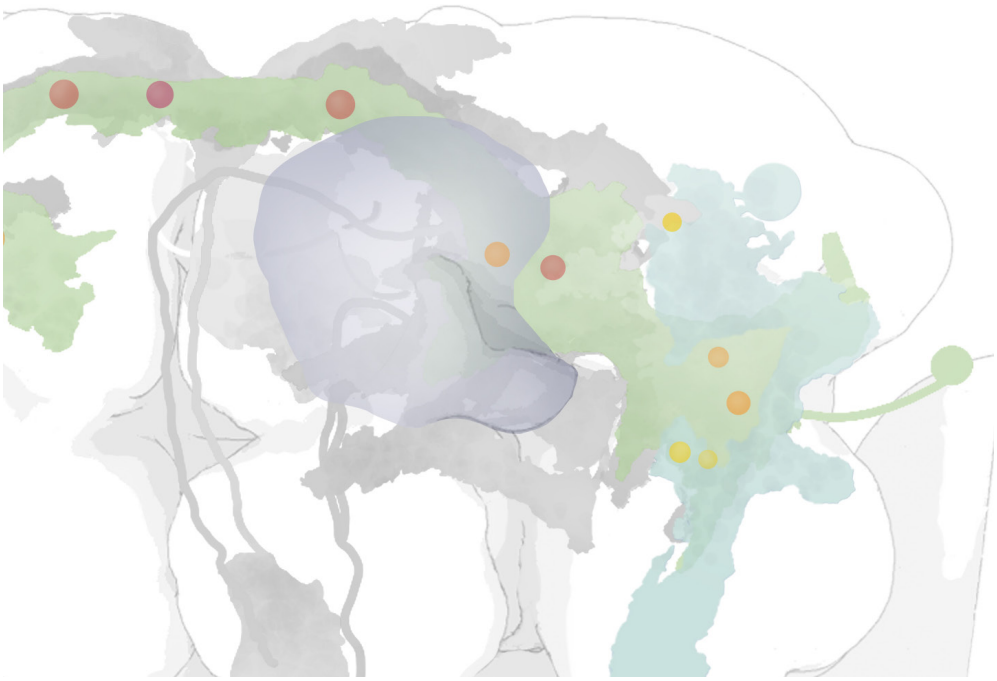
4.4 Connectivity and Interaction Design

Using abstraction, information representation in interactive systems can be properly scaled [66]. Most important in depicting potential connectivity is the information on the existence and significance of overlaps. Visualising all overlap regions at once in 3D would lead to a completely cluttered view. To avoid this, the existence of an overlap is initially only displayed using a glyph in the form of a small dot. Its colour indicates the order of overlap (compare the overlap colour scheme, Figure 4.9). The glyphs are easily recognizable within the contoured brain and are placed roughly at the position of the overlap. Clusters and core areas can thus be easily recognised.

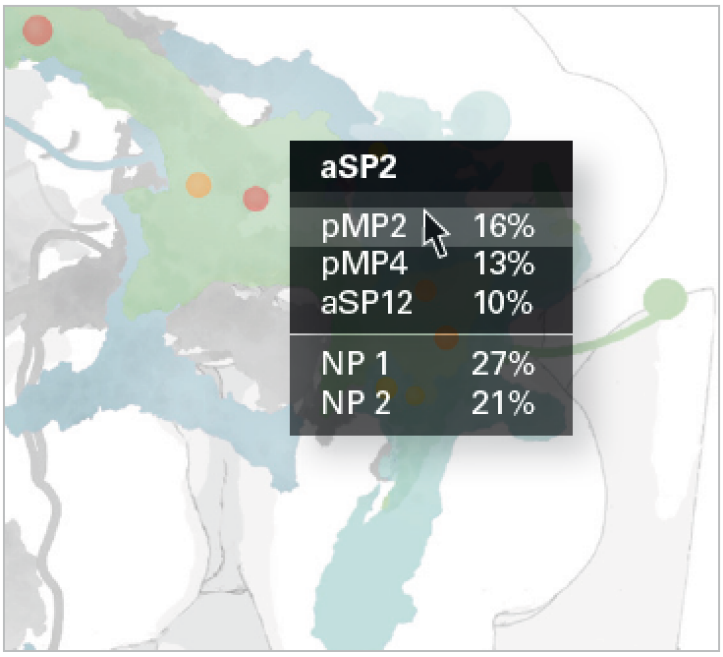
The glyphs not only encode overlap information, they are also the central interaction element. Hovering over a glyph immediately visualises its corresponding overlap using the overlap colour scheme. Clicking the glyph makes this a permanent selection, revealing the overlap in the same way as the hover action. Multiple overlaps may be selected at a time.

Two types of menus offer quantitative information on overlaps: the *tree menu* and a *tooltip menu* (see Figures 4.10b and 4.11). The quantitative overlap information consists of the list of participating arborisations and their relative volumes, i.e. how much of the arborisation's volume is part of the overlap, as well as the distribution to neuropils. The tree menu is located on the left side of the screen, it provides a complete and structured view on the quantitative data, including a perspicuous summary of all overlaps for each arborisation. The menu uses the same colour scheme and symbols as the 3D rendering. The menus are linked to the 3D visualisation, i.e. mouse interactions like selecting overlaps are synchronised.

Investigating the quantitative data combined with interaction in 3D helps find answers to all core questions (compare Section 2.4). The interaction design is more comprehensively described in Chapter 7 using the practical examples developed during the implementation.



(a)



(b)

Figure 4.10: The mock-ups are part of the final design proposal by Moosburner [50]. Neuropils are drawn on-demand. They are very reduced with slight transparency to provide some additional spatial context (a). The *tooltip menu* provides quantitative data for overlaps on mouse over (b).

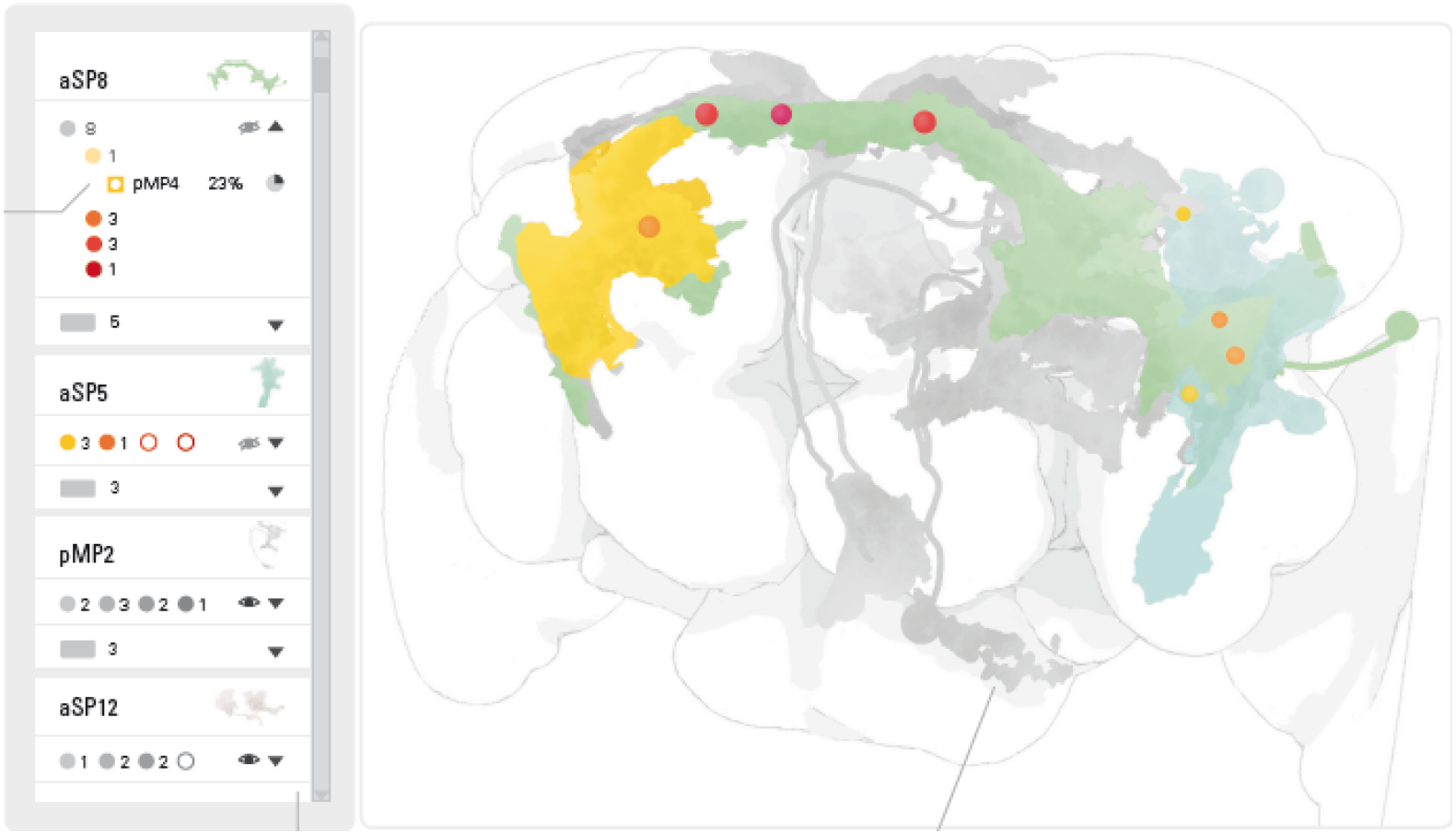


Figure 4.11: This mock-up is part of the final design proposal by Moosburner [50]. The *tree menu* offers a complete list of all found overlaps. Overlaps are initially abstracted as glyphs, then selectively shown in their full extent (here a pairwise overlap in yellow).

CHAPTER 5

Implementation

The implementation follows closely the design proposed in Chapter 4 but additionally takes into account the need for interactive performance. This requirement is reflected in the choice of highly efficient computational methods for volume calculation and interactive visualisations. The implementation was integrated into a larger system which already offered a multitude of tools for investigating the digital atlas. This chapter gives an overview of the implemented pipeline and specific technologies used to achieve an interactive system.

To summarise, a simple implementation of an A-buffer requires OpenGL 4.3, and due to the integration into the BrainGazer framework we use C++ with GLSL and compute shaders. Qt 4 helped make most of the graphical user interface elements.

5.1 Existing Environment: BrainGazer

The tool was implemented in the existing framework offered by *Brain** [11] (pronounce 'brain star'). This framework was started in 2008 in collaboration with the Institute of Molecular Pathology to support their research workflows. Brain* does this via multiple tools working together in unison. The three most integral parts of the framework are

- *BrainBase*, the database managing the complete brain atlas created during the *Drosophila* research by the IMP,
- *BrainBaseWeb*, a web interface to perform complex searches on the data, and,
- *BrainGazer*, a desktop application providing multiple tools for advanced search and filter strategies on the atlas and their visualisation.

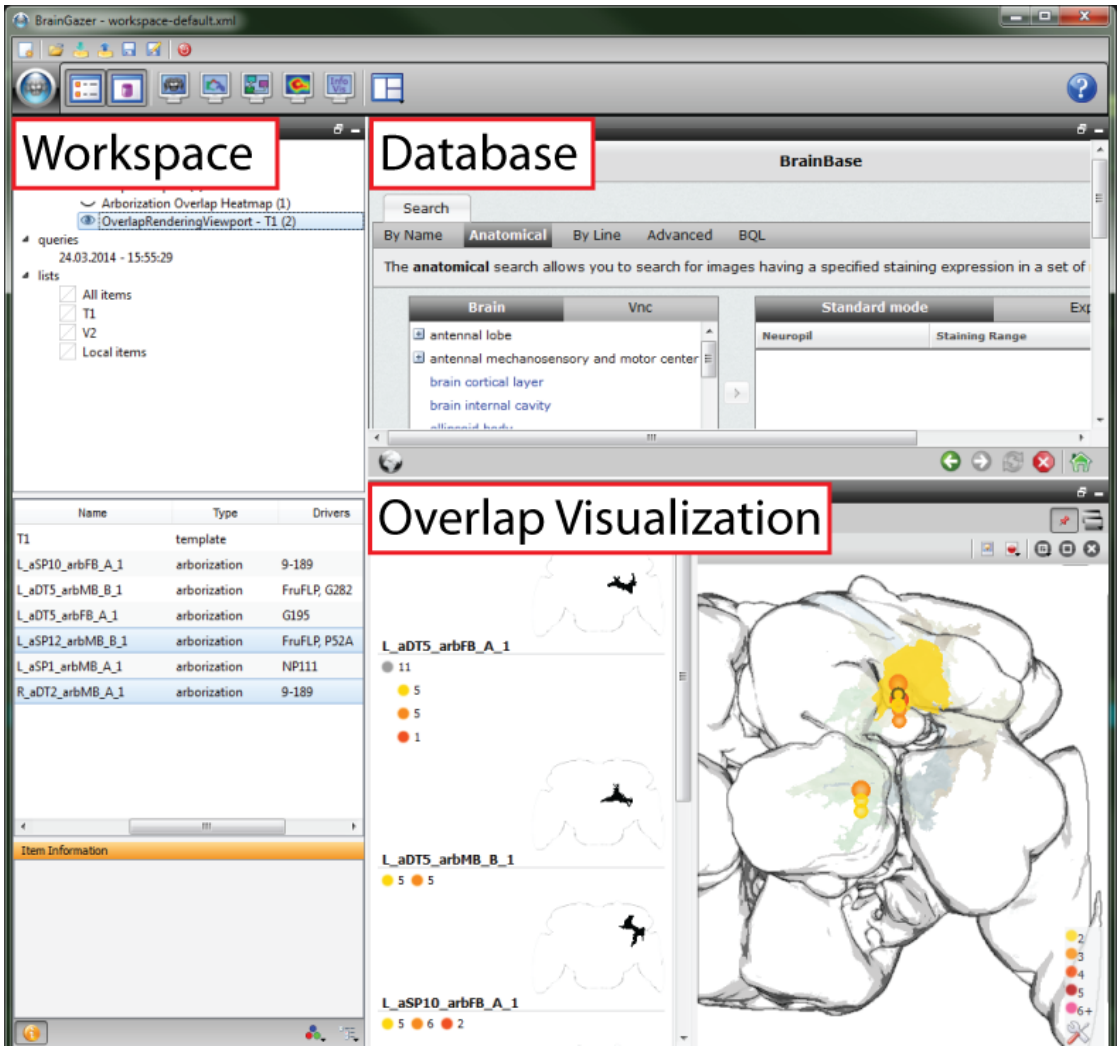


Figure 5.1: View on BrainGazer with only one viewport (our overlap visualisation) open. The workspace (left) handles all loaded arborisations, stores queries, and links multiple viewports together. An html view on BrainBaseWeb (top right) can be used to load data into the workspace. All opened viewports can then share the loaded data.

The desktop application BrainGazer provided the environment for implementing the new tool. BrainGazer accesses the database by including a view on BrainBaseWeb. This brings with it simple operations such as searching for arborisations of a specific name and many more complex search operations on the database.

The software is plug-in based with a central workspace. With it, the user can, during runtime, adjust the system by creating new viewports showing different tools. All viewports/tools interact with each other through this central workspace. It also controls loaded work elements such as neuron arborisations or GAL4 staining images.

Some of the tools, which the workspace of BrainGazer loads as separate viewports, are

- A *3D rendering* viewport that combines triangle mesh rendering (e.g. for segmented arborisations) and volume rendering (mainly for staining images),
- A *heatmap* for pairwise arborisation overlaps,
- *neuroMap*, a graphical representation of pairwise arborisation overlaps.

These are the major tools relevant to use cases related to potential overlap information. A heatmap can quickly visualise a huge number of pairwise overlaps (compare Figure 5.2). The sophisticated graphing tool neuroMap (compare Section 3.2 and Figure 3.3) visualises potential connectivity stored in our database *BrainBase* but it is limited to pairwise overlaps as well.

All these tools are linked across the workspace (compare Figure 5.1). This way they share loaded work elements. Among others, the workspace can load arborisations, cell bodies, neuropils, and whole neurons from the database. The workspace link also communicates selection information between viewports. The *cross-selection*—the propagation of selections to all other tools—is thoroughly discussed in Section 7.3. The concept of a central workspace makes a modular plug-in based architecture possible. Our new overlap visualisation tool was implemented as a plug-in. As such it benefits from the complete *Brain** architecture:

- We could reuse a large amount of the infrastructure, such as querying the database and loading data to the GPU.
- Users already familiar with BrainGazer, mainly our partners at the IMP, do not have to learn a completely new system.
- The tight integration with existing tools creates synergies. This is made apparent, e.g., by the *cross-selection*.

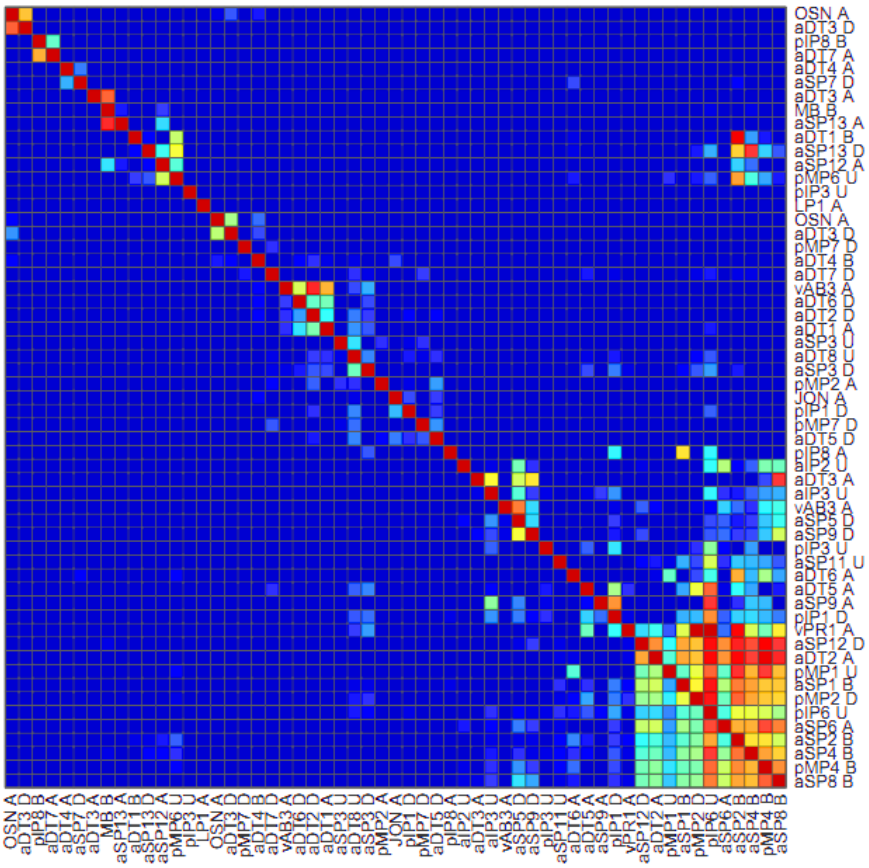


Figure 5.2: A heatmap available in BrainGazer visualises a large number of pairwise overlaps. Each pair of arborisations corresponds to two ratios, mapped to a rainbow colour scale from blue (0%) to red (100%). Per Peters' Rule (compare Sections 2.2 and 2.4) the larger one is likely to be the more interesting one.

5.2 Computational Pipeline

Figure 5.3 depicts a high level view on our processing pipeline. Once the neurons are loaded into the application's workspace, their arborisations undergo a volume estimation process. It calculates volumes of arborisations and neuropils and all arborisation overlaps (compare Section 6.1 for a detailed description). This information is stored and later used to feed the menus containing the quantitative information. Data loading and calculation take a few seconds, after which the rendering process starts. This entails rendering of context information such as brain surface, neuropils, and neurons, as well as drawing arborisation overlaps and rendering their representing glyphs (Section 6.3 discusses rendering techniques in detail). Loading additional arborisations initiates the volume estimation for the newly introduced overlaps.

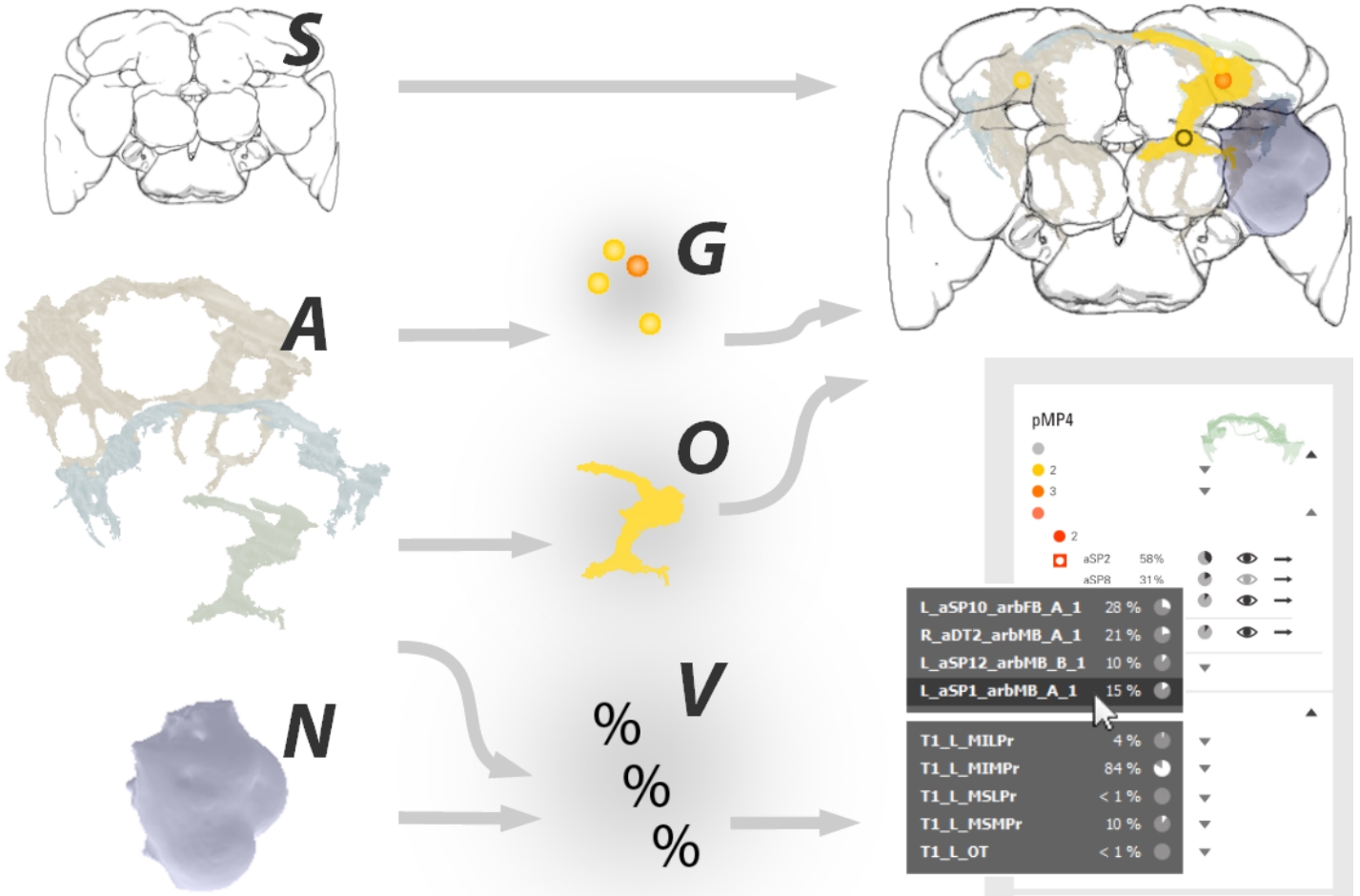


Figure 5.3: From left to right, this image describes our computational pipeline. Mesh information on the left is used to create the output on the right. Arborisation meshes A are needed to define overlaps O and glyphs G. For the volume calculation V we need arborisations and neuropils N. All these parts, along with the silhouette S contribute to the final rendering on the right. Quantitative information from the volume calculation can be explored in the menus, also depicted on the right.

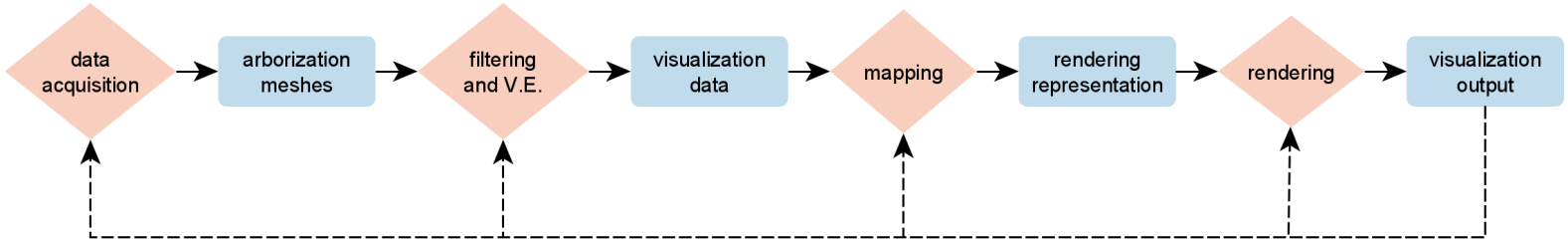


Figure 5.4: A simplified view on the system through the visualisation pipeline: The BrainBase database helps **acquire** the arborisation mesh data and meta information. The **filtering and V.E.** step computes additional information via the volume estimation. Then data are **mapped** to, e.g., abstracting glyphs, silhouettes, and a colour palette. The GPU **renders** the output at interactive framerates. The most vital interactions (indicated by the dotted line) are loading arborisations (input at the data acquisition step) and changing how and which overlaps are abstracted as glyphs, and which items are highlighted via selection/hover (input at the mapping step).

Figure 5.4 provides a view on the tool through the visualisation pipeline. It is completely integrated into the BrainGazer framework. This makes it easy to use in combination with other tools such as neuroMap [65].

5.3 Basic Data Structure: A-buffer

As previously mentioned, the database makes pre-calculated volumes available for arborisations and pairwise arborisation overlaps. This data even extends to a list of intersecting neuropils for each overlap—albeit without volume data—, giving important information on the spatial distribution of an overlap. However the database lacks any volumetric or other information on higher order overlaps, meaning overlaps between three arborisations or more. Pre-computing all these values and storing them is simply infeasible. For n intersecting arborisations the number of possible distinct sets of overlap is $2^n - n - 1$. For three arborisation meshes, there can be three pairwise overlaps and one additional three-overlap (i.e. where all three meshes overlap). Once we start looking at higher order overlaps, this number of potential distinct overlaps rises drastically. Five meshes can produce up to 26 distinct overlaps. Even if we cap the pre-computation at four-overlaps, a database of only 10000 arborisations results in up to $9999 + \sum_1^{9998} + \sum_1^{9997} = 99970003$ distinct overlaps.

Kalkofen et al. [35] use a G-Buffer data structure by Saito and Takahashi [58] to composite multiple renderings, including illustrative renderings. Both our central method of calculating volumes and the rendering of connectivity information use the more sophisticated A-buffer data structure. First described by Carpenter [14], an A-buffer can be used for order independent transparency or any kind of complex compositing. As an example of a system in a related field we mention Nowke et al. [52]. They use A-buffers for alpha-blending to visualise large-scale neural activity data of the Macaque monkey.

A-buffer Variations

Both for rendering connectivity information in 3D and for calculating intersection volumes we use a convenient A-buffer. An A-buffer is a list of fragments per pixel [14]. There exist variations of A-buffer implementations on the GPU tailored to specific applications [18, 39, 73].

Commonly this data structure is used for sorting fragments by depth to achieve true *order independent transparency* on modern graphics hardware. We however use it for *constructive solid geometry* (CSG) operations (intersections) and volume calculations. When rendering to the A-buffer, objects in view-space are projected to a 2.5D representation. All this projection's depth information is held in global shared memory on

the GPU and stored in linked lists. Each pixel points to its own linked list. In a second render step the fragments are sorted by depth.

The sorted lists are then used to calculate volumes and render arborisation overlaps, coloured according to the design (see Chapter 4 for details on the design decisions). The list iteration during rendering can be interpreted as ray casting. Once a linked list has been sorted by depth, all operations on it correspond to ray traversals from front to back.

To find a suitable implementation of an A-buffer, we first looked at the work of Crassin [18]. Based on this code we implemented paged per-pixel linked lists, a variant that improves cache coherency compared to naïve implementations. Storing more than one datum per pixel to create a linked list requires a locking mechanism. Just like Crassin we solve this *pixel synchronisation* with atomic operations available on OpenGL 4. This gives us one semaphore per pixel to ensure sequential write operations per pixel. Such pixel synchronisation exists natively on the Haswell architecture (4^{th} Generation Intel Core Processor) and above as a shader extension for DirectX 11. We opted for the software implementation in OpenGL 4 because it is used in BrainGazer, our framework of choice.

A simple A-buffer

An A-buffer must provide functions to read and write to the per-pixel linked lists. In each pass an A-buffer can only be used for one of these operations since we cannot rely on memory fences on the GPU. We use multiple A-buffers for storing mesh data, including depth values, as described in the following chapter. Compare Figure 6.2 for the illustration of an A-buffer.

Our specific OpenGL implementation uses four separate shader storage buffers to create a single A-Buffer. The first two are screen-size buffers. One stores links to the heads of the per-pixel linked lists, the other counts of stored elements/fragments. The other two buffers store the fragment data (e.g., depth, mesh identifier, ...) and a link to the next fragment. Their allocation size has to be big enough for all fragments. The easiest way to ensure that all fragments fit is to use an atomic counter or query buffer to count the fragments. If a shader pass exceeds the limit of the allocation (i.e. there is more geometry in the scene than anticipated) this pass fails. In this case the pass will be repeated immediately after a reallocation: the exact number of required fragments can be queried from the failed pass. The two buffers that store fragment data and links are resized with a conservative safety margin. All these buffers may be written to or read from in fragment (or compute) shaders but never both. They may also be downloaded to the CPU.

Saving a single fragment is quite a complex endeavour. When it first arrives in the fragment shader, we try to get a semaphore lock on this specific screen coordinate. This is necessary for pixel synchronisation and requires an atomic operation available in

OpenGL 4. Then the shader, again atomically, acquires the next free spot in the storage buffers and stores this fragment and a link to its address. Finally the shader releases the semaphore lock.

CHAPTER 6

Volume Estimation and Rendering

This chapter describes algorithms employed by the system and their integration into pipelines/workflows. The focus lies on explaining a novel method used for computing intersections of brain regions. Following the section on this Volume Estimation is a brief section that shows how the user interface represents these computed values. The last section discusses various rendering techniques, which, combined, create the 3D visualisation. These rendering techniques are chosen to achieve an approximation to the design (compare Chapter 4) while still maintaining interactive performance.

6.1 Volume Estimation

The Volume Estimation method is an integral part of the system. It is essential for the visualisation to include volume information on arborisation overlaps. The design displays relative volume information as percentages to help the scientists judge the importance of overlaps, i.e. by applying Peters' Rule (compare Sections 2.2 and 2.4).

The system calculates relative volumes for arborisation intersections, as well as for their intersections with neuropils on-the-fly. This section, however, solely explains the volume calculation of arborisation meshes and their intersections, the extension to include neuropil meshes is trivial.

As previously described in the pipeline overview (Figure 5.3), the system locates overlaps, can abstract them with glyphs, and eventually calculates the intersection volumes. To do this Volume Estimation, first all arborisations in question are rendered to the A-buffer. These segmented neuronal structures are available as mesh information. Although the neuronal structures exist as segmentation masks as well, we decided on calculating intersection volumes from their mesh representation. Firstly, we already have the mesh information on the GPU since we use it for the viewport rendering, and

secondly, available memory limits the number of segmentation masks we can load at any one time. In a typical use case, we expect about ten meshes to be loaded at one time, although we performed the accuracy tests mentioned in this chapter with 50 arborisation meshes.

Following the mesh rendering pass most subsequent calculations run on the GPU to take advantage of its parallelism. Mesh depth information stored in the A-buffer suffices to determine regions of overlap and even to estimate volumes. Our Volume Estimation method can be briefly explained with a comparison to ray casting. The A-Buffer provides us with all relevant depth values per pixel. For each pixel it stores a linked list of depth value and mesh identifier pairs. In the simplest case of a convex mesh this is one value for a front face and one value for a back face, compare Figure 6.3 for other cases. Similarly to ray casting, the GPU iterates all per pixel depth values front to back and computes their differences. The sum of these depth differences approximate the mesh volume. As mentioned in Section 5.3 we also use these depth values to compute intersection volumes of meshes. Our method is fast and accurate in calculating volumes of meshes and their intersections. It is flexible in application and could be easily extended to the other CSG operations, union and difference, should the use case so require. These operations are performed by the shader with the A-buffer data. For these intersections no separate meshes need to be calculated, saving the time to do CSG calculations on the CPU or pre-build a CSG tree. The meshes can be arbitrarily complex, but they must bound a finite volume, i.e. they must not have holes.

We refer to the volume calculation itself as an estimation because it does not produce exact results. *Summing signed volumes of polyhedrons* would produce exact results but it requires all the mesh faces to be either clockwise or counter-clockwise in orientation. Our method, however, works irrespective of face orientations and normal information.

The following two subsections describe in detail how the A-buffer is used to calculate mesh volumes. The last subsection presents evaluation results on the accuracy of the method.

Volume Estimation in Orthographic Projection Space

We use the depth samples stored in the A-Buffer to compute depth differences. In orthographic projection space these values are sampled over a regular grid, compare Figure 6.2 for a schematic representation. Figure 6.1 shows a flow diagram of the Volume Estimation pipeline. It consists of at least three shader passes. In the first pass (*render meshes*), meshes are rendered and stored to A-buffers. In this example we use two separate A-Buffers to store depth values from arborisations and neuropils respectively. In a second pass (*sort by depth*), the values are sorted by depth into two further A-Buffers. At this point the figure shows one optional shader pass (*find sets*) that determines all combinations of arborisations that intersect. In the third required pass (*depth differences*), we compute per-pixel depth differences from the mesh depths. Optionally

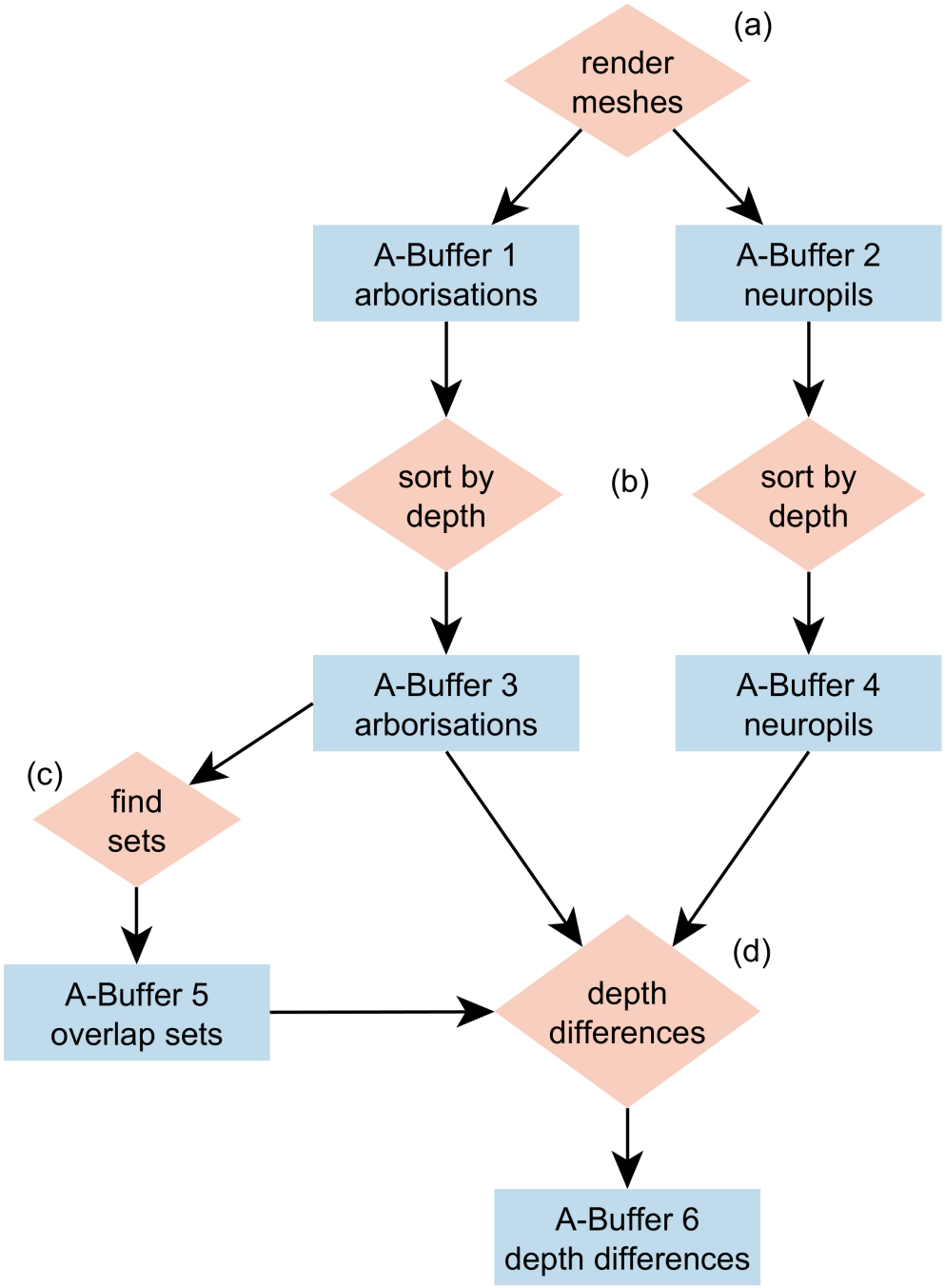


Figure 6.1: A flow diagram of the Volume Estimation process on the GPU. Shader passes from top to bottom: (a) the necessary arborisation and neuropil meshes are rendered to their respective A-buffers in a single pass; (b) in a separate pass the A-buffers are sorted into new A-buffers; (c) this pass iterates the sorted depth values to determine all existing arborisation overlap combinations; (d) the depth differences are computed for all meshes, and optionally for all existing mesh overlap combinations.

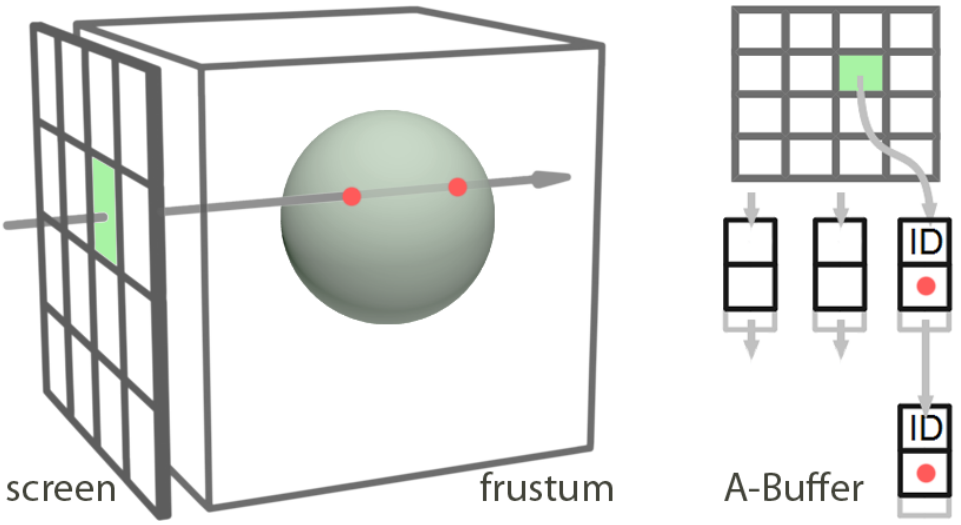


Figure 6.2: The mesh (green sphere) in orthogonal frustum has been rasterised to the A-buffer. The A-buffer stores the depths of the entry and exit point (red dots). The depth values of a single pixel as depicted here, are stored in their own linked list, along with a mesh identifier *ID*. The order of the depth values (red dots) is not guaranteed.

this pass can use the output from the *find sets* pass to also compute depth differences for intersection volumes. All of these differences are stored to a final A-Buffer. This buffer, along with the optional A-Buffer from the *find sets* pass, is then downloaded to the CPU where the sum of the depth differences, the estimated volume, is calculated. The addition is done on the CPU because it does not profit from the parallelism of the GPU. In theory it could be performed by reduce operations on the GPU. However, this would require many additional A-Buffers which would undo any potential performance gain of a GPU reduce operation compared to the sequential addition on the CPU.

These steps of the Volume Estimation pipeline are described in more detail in the following paragraphs. Section 6.3 explains how some of these shaders are used to create a rendering of the mesh overlaps.

Storing mesh data in the A-buffer. Meshes are rendered to the A-buffer using an orthogonal projection matrix, which is fitted to the scene, in this case the brain's bounding box. In the first shader pass, *render meshes*, they are projected along the z-axis. The z-values are stored in the A-buffer along with mesh identifiers. The detailed pseudo code in Algorithm 6.1 shows how a value is stored in a paged A-Buffer. We do not cull back faces as we need all depth information later on. The orthographic projection to the A-buffer samples the meshes over a regular grid, in our case the A-buffer has a size of 512 by 512 pixels to accommodate the entire bounding box (compare

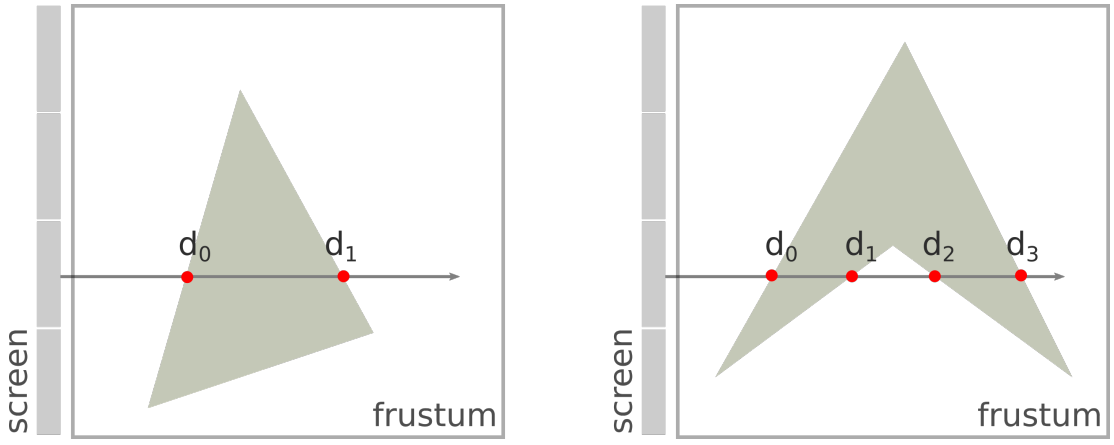


Figure 6.3: The rasterisation of the mesh (green object) in an orthogonal frustum is easily understood in a schematic 2D view. The A-buffer stores the depths of the entry and exit point (red dots). Concave objects result in multiple depth values per pixel. When sorted by depth the values with even indices d_{2i} are entry points, those with odd indices are exit points d_{2i+1}.

Figure 6.2 with a schematic grid of 4 by 4 pixels). As shown below in the Subsection Accuracy and the accompanying Table 6.1, smaller and thus faster A-buffers are also sufficient.

Each pixel indexes its own linked list of mesh information. A simple convex mesh would create two entries in each pixel it occupies: one data point for the front face and one for the back face. A data point records the depth (the linear view-space depth, in our case of the orthogonal projection this is a z-value in model space) of where the mesh is located in 3D-space and, of course, a mesh identifier to match the depth values to their arborisation or neuropil. The GPU does not guarantee the order in which the polygons are rendered, the depth values may or may not be sorted at this point. Compare Figure 6.2 for a simple overview of the projection and the organisation as linked lists in the A-buffer, and Figure 6.3 for an illustration for non-convex meshes.

Sorting the A-buffer. Before doing any further operations on the depth information we must sort the lists. We perform a separate shader pass for sorting the lists into a newly allocated A-Buffer. Instead we could sort the lists every time they are read by a shader, without saving the sorted list itself. We have not noticed any difference in performance for these two approaches. Vasilakis and Fudos [73] use either insertion sort or shell sort depending on list length, in our use cases insertion sort turned out to be efficient enough.

Calculating depth differences from the A-buffer. At this stage of our pipeline all meshes have been projected to the A-buffer and are therefore available in global GPU memory. The A-buffer stores the sorted mesh depths, with these we now calculate

Algorithm 6.1: Pseudo code of the shader that stores a value to an A-Buffer. This section may be executed for each pixel multiple times. The semaphore, implemented with atomic operations, ensures that only one instance per pixel writes to its linked list. An atomic add safeguards the allocation of new pages in the shared memory. For allocating space this version uses pages that fit four fragments. We have tested the code with and without paging and have not experienced any performance changes.

```
Data: One value val (e.g. consisting of depth value and mesh identifier),
      writeable A-Buffer, screen position XY.
Result: The value has been stored to the A-Buffer.
1  pageIndex ← NULL
2  numFragmentsInCurrentPage ←NULL
   // get a pointer to the shared pool:
3  while True do
4      if AcquireSemaphore (XY) then
           // get last page index for this pixel:
5          pageIndex ←GetLastPageIndex (XY)
6          numFragments ←GetFragmentCount (XY)
           // our pages store up to four values
7          numFragmentsInCurrentPage ←numFragments mod 4
8          if numFragmentsInCurrentPage == 0 then
               // allocate a new page:
9              newPageIndex ←AtomicAddSharedPageCounter (4)
               // save the link to the previous page:
10             SetLinkSharedPool (newPageIndex,pageIndex)
               // save the new page as current for this
                  pixel:
11             SetLastPageIndex (XY,newPageIndex)
12             pageIndex ←newPageIndex
13         end
14         SetFragmentCount (XY,numFragments + 1)
15         ReleaseSemaphore (XY)
16         break
17     end
18 end
   // save the new fragment in the shared pool:
19 SetValueSharedPool (pageIndex + numFragmentsInCurrentPage, val)
```

differences. In a pixel, a single mesh has the depth of $d = d_1 - d_0$, where d_0 is the first depth value in the A-buffer and d_1 is the second one. More complex, non-convex meshes may have a multiple of two depth entries in a single pixel (compare Figure 6.3 for an illustration of this naming schema). After sorting, these n depth values are alternating entry d_{2i} and exit d_{2i+1} points to the mesh interior. For every entry point there is exactly one exit point, thus n is even. The depth d_{pix} of a mesh in a single pixel of the A-buffer is

$$d_{pix} = \sum_{i=0}^{n-1} d_{2i+1} - \sum_{i=0}^{n-1} d_{2i} \tag{6.1}$$

For each mesh, these depth differences are written to a float buffer on the GPU (compare shader pseudo code in Algorithm 6.2) and later summed up by the CPU. When scaled by the area of a pixel, the sum of all depth differences is a representation of the mesh volume. With $width_{bb}$, $height_{bb}$ the width and height of the brain's bounding box, which was used to create the frustum, and $width_{ab}$, $height_{ab}$ the A-buffer's sizes in pixels, the final estimated mesh volume is

$$V_{orthographic} = \frac{width_{bb} \cdot height_{bb}}{width_{ab} \cdot height_{ab}} \sum d_{pix} \tag{6.2}$$

Algorithm 6.2: Pseudo code of the compute shader that calculates depth differences for all meshes, this code is executed once for each pixel. A fragment is a pair of depth and mesh identifier.

Data: A-Buffer A with sorted depth values and mesh identifiers, empty Buffer B.
Result: Depth differences for each mesh in Buffer B.

```
// accumulated depth difference of each mesh:
B ← []
// Inside/Outside array initialised to zeros:
inout ← [0]
fragments ←GetFragments(A)
foreach meshId, depth of fragments do
    inout[meshId] ← inout[meshId] + 1
    if inout[meshId] is odd then
        entryDepth ←depth
    end
    else
        B[meshId] ← B[meshId] + depth − entryDepth
    end
end
```

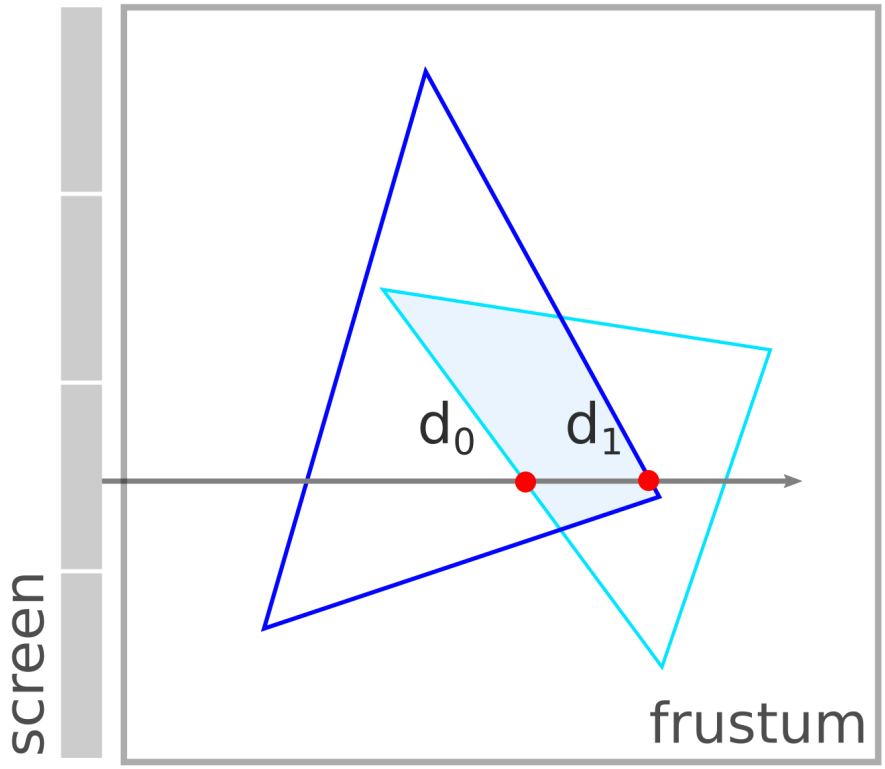


Figure 6.4: To compute the intersection volume, the meshes (cyan and blue) are rendered to an A-Buffer in an orthogonal frustum. The A-buffer stores pairs of depth and the mesh identifier. To determine which depth value constitutes an entry (or exit) point to the intersection volume, we need additional information. Previously we stated that the values (sorted by depth) with even indices d_{2i} are entry points, those with odd indices are exit points d_{2i+1}. In the case of intersections we must keep track of meshes along the tracing ray. Once we step inside all (here two) meshes, we record d_0 (red dot) as the first entry point. When we leave one of the meshes, we record d_1 as an exit value.

Intersecting meshes in the A-buffer. This Volume Estimation can not only be done for a single mesh, but also for intersections of multiple meshes. The computation of such volumes on the GPU on demand is the keystone of this thesis. The sorted linked lists of the A-Buffer let us achieve this goal quite easily. We need to find in the A-buffer the correct depth values bounding an intersection. Sorted by depth value, the data pairs of mesh identifier and depth make this set operation a simple matter of iterating the linked list at each pixel. The next paragraphs make this clear for an intersection of two meshes, the extension to three or more is trivial. In the same manner the Volume Estimation may be extended to other set operations, thus it can be used for all CSG operations directly on the GPU.

Algorithm 6.3: Pseudo code of the compute shader that calculates depth differences for a mesh intersection, this code is executed once for each pixel. A fragment is a pair of depth and mesh identifier. This example extends the simple pseudo code in Algorithm 6.2

```
   Data: A-Buffer A with sorted depth values and mesh identifiers, the
         mesh combination OverlapSet to intersect, empty Buffer B.
   Result: Depth differences for the mesh intersection in Buffer B.
   // accumulated depth difference:
 1 B ← 0
   // Inside/Outside array initialised to zeros:
 2 inout ← [0]
 3 order ←number of meshes in OverlapSet
   // Inside/Outside counter:
 4 i ← 0
 5 fragments ←GetFragments(A)
 6 foreach meshId, depth of fragments do
 7     if meshId in OverlapSet then
 8         inout[meshId] ← inout[meshId] + 1
 9         if inout[meshId] is odd then
10             i ← i + 1
11             if i == order then
12                 entryDepth ←depth
13             end
14         end
15         else
16             if i == order then
17                 B ← B + depth − entryDepth
18             end
19             i ← i − 1
20         end
21     end
22 end
```

Calculating depth differences (and subsequently volumes) of an intersection requires knowledge about which depth value represents an entry or exit to the intersection interior. For a pairwise overlap, i.e. an intersection of two meshes A and B, the shader needs only their two respective mesh identifiers as they are stored in the A-buffer alongside the depth values. Figure 6.4 gives a 2-dimensional schematic overview of a pairwise overlap.

Algorithm 6.3 shows pseudo code for the intersection of an arbitrary number of

meshes, a set of their identifiers must be provided as input *OverlapSet*. While iterating the values of the A-buffer (sorted by depth), the shader continuously updates a list of meshes that have been entered. If, at a point, both meshes A and B have been entered, the depth value must be an entry point for the intersection. This way entry and exit points are determined and eventually, as described above for a single mesh, the depth differences are written to a buffer to be later summed up by the CPU.

Determining the input for this Algorithm 6.3, i.e. the set of identifiers describing the intersection (input *OverlapSet*), is a separate step. This *find sets* pass illustrated in Figure 6.1 (and, for the rendering pipeline, in Figure 6.7) creates a collection of all possible sets of mesh intersections, excluding mesh combinations that do not intersect. The pass writes this information to a separate A-Buffer. Both of these figures refer to such an overlap set buffer. It stores a dense encoding of the available intersections in each linked list. An intersection between three meshes with identifiers 23, 24, and 25 would be encoded as $3-23-24-25$, and thus taking up four entries in the linked list. Each overlap set is stored as a list of integers: first the order of the overlap, then a list of mesh identifiers.

The following Section 6.3 also refers to the collection of all these overlap sets with an A-Buffer, in respect to the rendering pipeline. Here, this buffer serves to decide on the CPU, which mesh intersections actually exist. This way the shader pass calculating the depth differences is not done for mesh combinations that do not produce any overlap. Once this Volume Estimation is concluded for all meshes and their overlaps, the relative volumes are calculated. These are later presented in the user interface.

Volume Estimation in Perspective Projection Space

It is important to note that the fitted orthogonal projection is not necessary to calculate volumes. With a modest loss of accuracy we can instead use a perspective projection. This makes it possible to calculate volumes directly from the A-buffer we use for rendering overlaps and glyphs (compare Section 6.3).

The only notable change is the calculation of the depth differences. Instead of a depth difference we must calculate a difference of pyramid volumes in each pixel. Figure 6.5 gives a schematic 2D view of this computation. From each fragment's depth in the A-buffer, we must create a pyramid, which extends from the eye to the pixel in projection space. Its height is the (linear) depth of the fragment in view space (d). Its rectangular base (width a and height b) is the representation of the pixel in projection space. $V_{pyramid}$ describes the volume of the pyramid.

$$V_{pyramid} = \frac{d \cdot a \cdot b}{3} \tag{6.3}$$

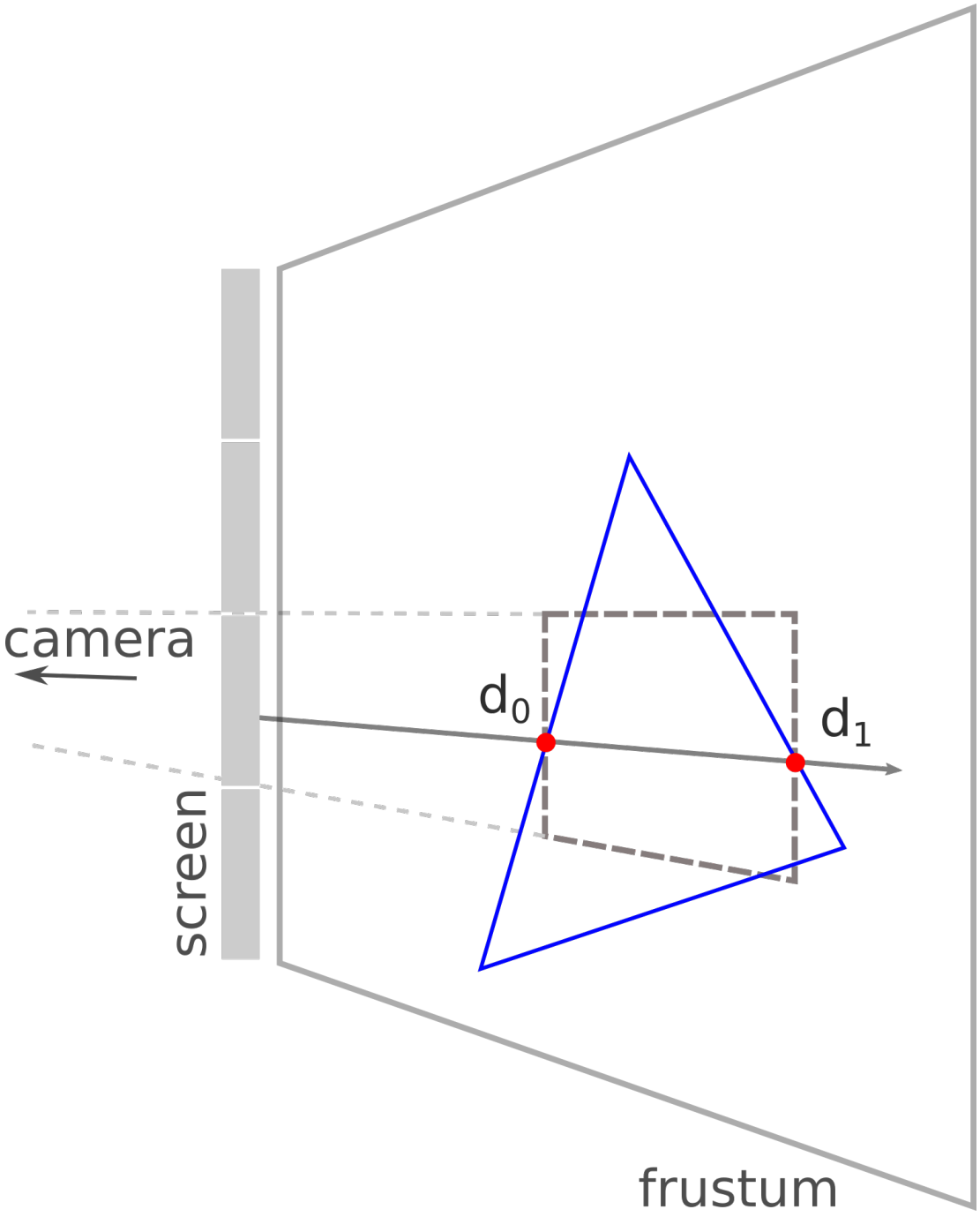


Figure 6.5: The rasterisation of the mesh (blue) with a perspective projection in a schematic 2D view. The A-buffer stores the depths of the entry and exit point (red dots). The volume can be computed from two pyramids. They extend from the camera to the depth values. The smaller one's height is d_0. Its base is the representation of the screen pixel in projection space, a rectangle $a_0 \cdot b_0$ (represented in the image by the dashed grey line crossing d_0). The volume computed in this pixel is $V_{pixel} = d_1 a_1 b_1/3 - d_0 a_0 b_0/3$.

With two pyramids (constructed from an entry point d_0 with a_0 and b_0, and an exit point d_1 with a_1 and b_1) we can compute the volume in a pixel.

$$V_{pixel} = \frac{d_1 \cdot a_1 \cdot b_1}{3} - \frac{d_0 \cdot a_0 \cdot b_0}{3} \tag{6.4}$$

In our specific use case we care about relative volumes. Thus we do not need to address calculating a and b. We have instead tested a simplified version of this Volume Estimation by ignoring constants. Since a and b are directly proportional to d, we replace them in the equation. We omit the division by 3, resulting in the per-pixel volume difference $V_{relative}$.

$$V_{relative} = {d_1}^3 - {d_0}^3 \equiv V_{pixel} \tag{6.5}$$

A drawback of this calculation is of course the loss of absolute values. A quick way to find a conversion ratio back to the actual volume is to do this Volume Estimation on a mesh of known volume. We implemented this Volume Estimation to make use of the same A-buffer that was employed for rendering overlaps and glyphs, but eventually decided against this approach. It shows major disadvantages compared to the orthographic version. The frustum is not perfectly fitted to the relevant meshes (to the brain's bounding box), which reduces accuracy. Even more problematic, the frustum can completely or partially miss meshes, because it is controlled by the user via zooming and panning actions. Completely omitted meshes would not pose a problem, partially omitted meshes would distort the Volume Estimation. The perspective projection's frustum also does not provide the same resolution everywhere. The closer we get to the far plane, the more accuracy we lose in all dimensions (depth, width, and height).

The only advantage it has over the orthographic version is that it saves a few shader passes, including the render pass. Essentially it operates on the same pipeline described in Section 6.3 (compare Figure 6.7). Because of the major disadvantages we decided against using the perspective version. Creating additional A-buffers for the orthographic projection comes at a low cost.

Accuracy

The Volume Estimation employed here leads to percentages of integer precision, describing the relation of two meshes A and B (or more) to their intersection. The value of 100% for A means that it lies entirely inside the investigated overlap, a value of 1% means A is barely part of the overlap—very likely making this an overlap of low importance for the neurobiologist. 0% cannot occur, since meshes not part of the specific overlap are not even considered; thus the smallest values are denoted by '$< 1\%$'.

Calculating the per-pixel depths d_{pix} (or per-pixel volumes) as well as summing them up to $V_{orthographic}$ introduces a negligible numerical error. We have tested the Volume Estimation's accuracy using 50 arborisation meshes (randomly selected from the

A-buffer Size [pixel]	$\bar{r}$	s_r	t_1[ms]	t_2[ms]	t_3[ms]	t_4[ms]
64x64	0.988697	0.077697	16	5	2	1
128x128	1.003902	0.028103	16	17	4	1
256x256	1.003049	0.008036	21	58	14	6
512x512	0.999934	0.003300	26	73	44	24

Table 6.1: The table gives Volume Estimation results of 50 arborisation meshes, of all together over 8 million triangles. The first column shows the resolution of the A-buffer. $\bar{r}$ and s_r denote mean and variance of the estimation's divergence from an exact calculation. The timings are approximate upper limits on an NVIDIA GeForce GTX 670 and an Intel Core i7 920. t_1: allocate A-buffer memory and render meshes, t_2: calculate depth differences on GPU, t_3: download buffer of depth differences to CPU, t_4: sum on CPU.

database, all together over 8 million triangles), rendered to an A-buffer of resolution 512 by 512 pixels. The resolution of the A-buffer controls the number of depth differences that a particular mesh is divided into. Generally, more samples in x and y direction make for a more exact estimation.

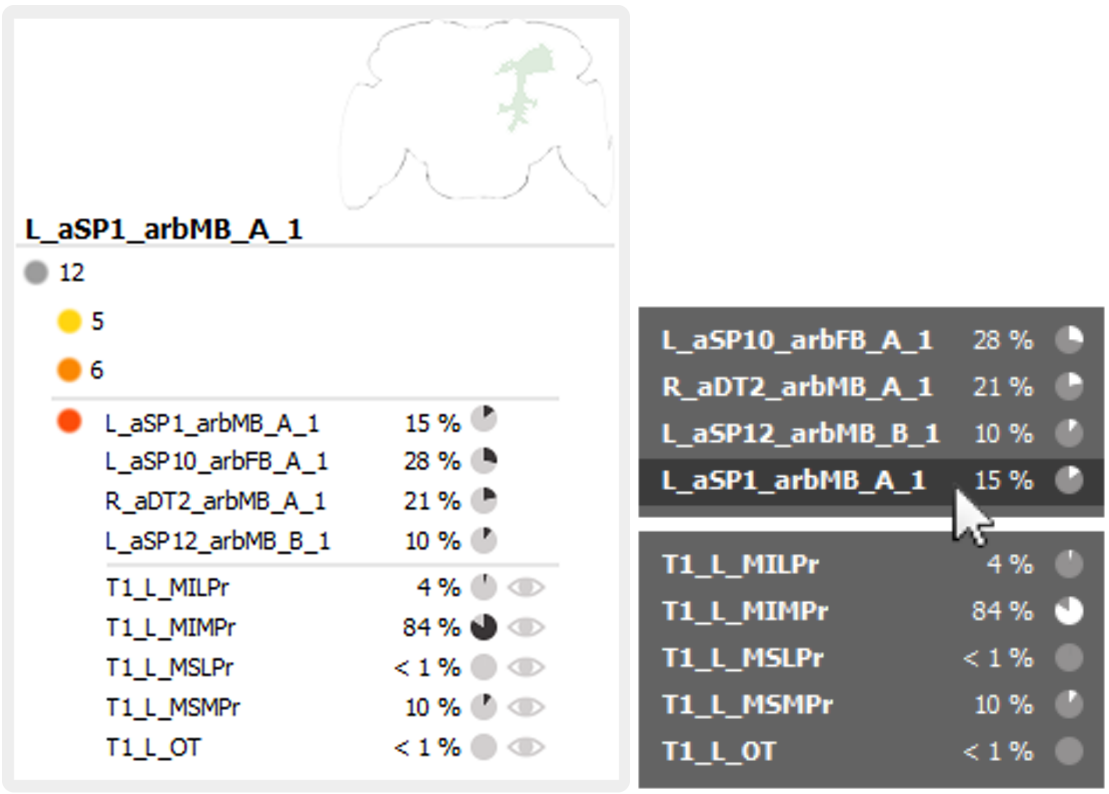
As ground truth, we calculate the triangle mesh volumes $V_{groundtruth}$ using *signed volumes of tetrahedra* [78]. Dividing each $V_{groundtruth}$ by the corresponding $V_{orthographic}$ results in 50 ratios, which we expect to be 1 each. The test leads to a mean $\bar{r} \approx 0.999934$ and a variance $s_r \approx 0.003300$. The method of estimating volumes proves by far sufficient for our use case: to quickly compute volumes with integer accuracy. Compare Table 6.1, listing timings and accuracy of different A-buffer resolutions.

We also tested the Volume Estimation in perspective projection space. The method is sufficiently accurate for our needs. The following values have been acquired with an A-buffer of 512 by 512 pixels and the same 50 arborisations. Although $\bar{r}_{perspective} \approx 43.813915$ cannot be compared to the values in Table 6.1, $s_{rperspective} \approx 0.142829$ is comparable. Because our use case requires only relative volumes of integer precision, this variance is sufficiently low. The loss of absolute values is irrelevant. We eventually decided against this perspective method, however, as mentioned in the preceding subsection.

6.2 Quantitative Information

The volumes computed by the Volume Estimation are, as shown above, quite precise. This estimation suffices to give us integer exact volumes in μm^3 with Equation 6.2. In the user interface, however, we only want to show relative volumes.

In two different user interfaces, we show the exact same quantitative information relating to a single overlap. Figure 6.6 shows an example of the tree view and the



(a) tree view (b) tooltip

Figure 6.6: (a) The tree view lists each arborisation by name, here we see only L_aSP1_arbMB_A_1. To add some spatial context, next to the name is an icon showing the arborisation's approximate location in the brain. Below, we list all overlaps this arborisation participates in. They are sorted into categories by their order/colour. Here it shows three: The first category (yellow, i.e. pairwise overlaps) contains five different overlaps. The second category (orange, i.e. three-overlaps) shows six distinct overlaps. Both of these are collapsed. The last category (four-overlaps) is expanded to display quantitative information on a single four-overlap, separated into arborisation and neuropil info. (b) The tooltip shows the same quantitative information.

tooltip menu with a four-overlap (an intersection of four meshes). Both views separate the quantitative information into two blocks. The example shows four arborisations in the upper block, and five neuropils in the lower block.

The percentage next to an arborisation denotes how much of it is participating in the overlap in question. These are the relevant values that the experienced user needs to render a judgement on the importance of the overlap (via Peters' Rule, compare Section 2.2). Next to each percentage, a small circles shows the same value. These graphics makes it easy to quickly see outliers and compare percentages.

The amount next to a neuropil shows how much of the overlap is inside of this neuropil. These numbers add up to 100% and provide additional spatial information to users familiar with the neuropils.

6.3 Rendering Techniques

The analysis of higher order overlaps in 3D requires smart abstraction choices to avoid occlusion. Our design (see Chapter 4) describes separate elements, which are eventually combined to a final rendering. Figure 5.3 offers a high level summary of this pipeline. The silhouette and neuropil meshes directly contribute to this final image. Rendering of arborisations, overlaps, and glyphs requires the use of the A-buffer. The graph in Figure 6.7 abstracts our customised rendering pipeline: Up to five textures are created and composited to the canvas.

The following subsections describe the selected rendering techniques that achieve an approximation to the design while maintaining interactive performance. The focus of the first subsection lies on the visualisation of overlaps, the remaining subsections summarise other techniques used in the rendering viewport. All of these fit together in the pipeline overview depicted in Figure 5.3.

Connectivity Information

The essential information we need to encode in the 3D rendering is the location and size of overlaps. The abstraction choice of using glyphs makes it easy to visualise many overlaps at a time. Loading multiple intersecting arborisations creates an exponential number of overlaps, each abstracted by its own glyph. Here follows a description of this part of the rendering pipeline, as summarised in Figure 6.7. Please compare its similarities with the Volume Estimation pipeline as described in the preceding Section 6.1 and summarised in Figure 6.1. The pipelines share some shader code.

Rendering meshes to the A-buffer. First, as in the Volume Estimation, we employ an A-buffer to store all mesh data. The viewport should support 3D interaction, so the rendering step uses an appropriate perspective projection matrix and model view matrices, as opposed to the Volume Estimation.

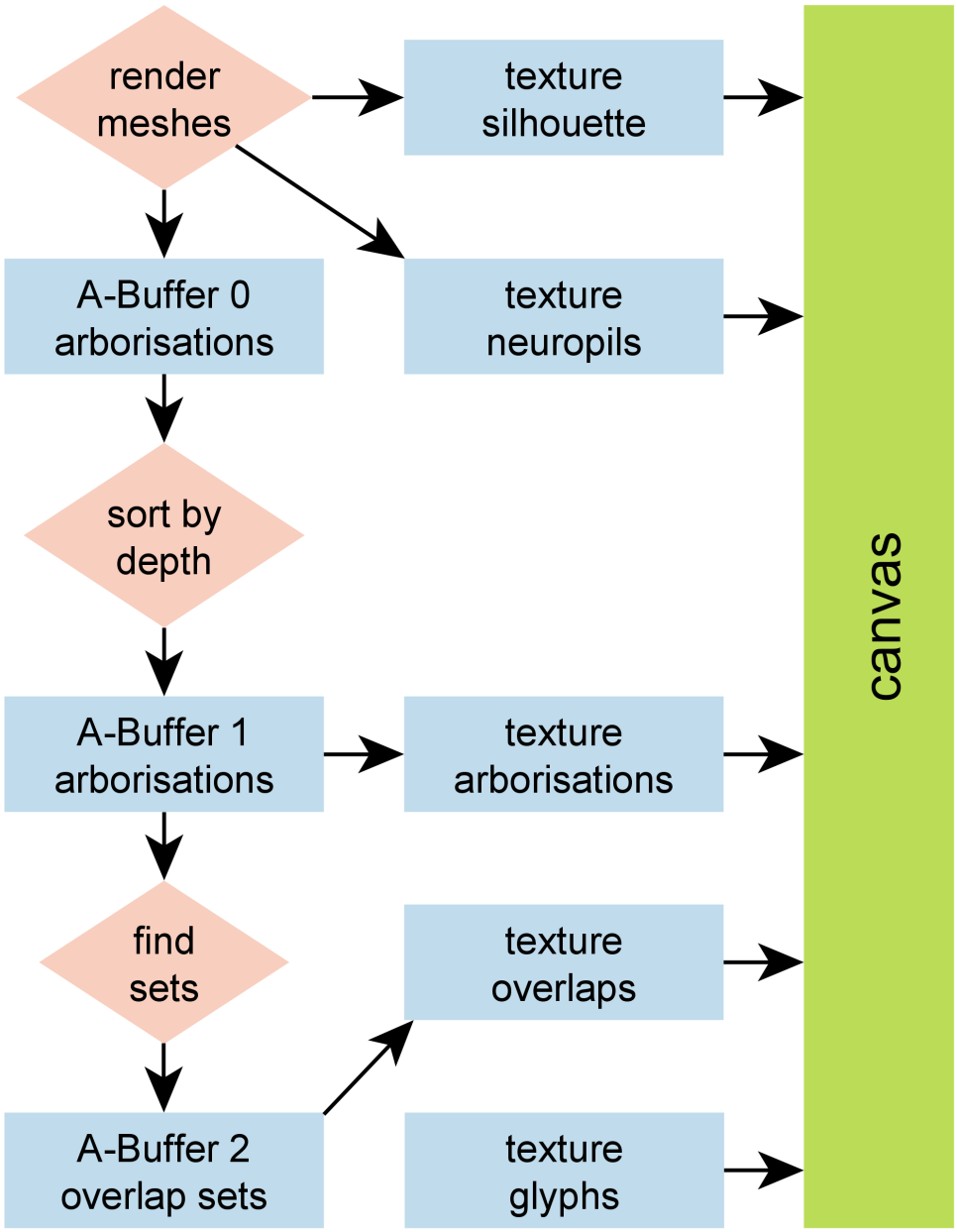


Figure 6.7: This simplified graph summarises the rendering pipeline that produces the 3D visualisation. Arborisations are rendered from a sorted A-buffer, while silhouette and transparent neuropils are rendered directly. Selected overlaps are rendered directly from A-buffer 2, which stores the overlap sets. The same buffer is downloaded from the GPU to compute glyph positions. Then the glyphs are rendered and all textures are composited to the canvas.

Sorting the A-buffer. Second, analogous to the Volume Estimation, the A-buffer must be sorted by depth, again using a simple pass of insertion sort.

Creating and downloading the overlap set buffer. In the next render pass, after the A-buffer has been filled with all arborisation meshes, we render its contents to a separate A-buffer. Section 6.1 also refers to such an A-Buffer that stores overlap sets. Compare A-Buffer 2 in Figure 6.7 and A-Buffer 5 in Figure 6.1. In this step we combine meshes to overlaps. The resulting A-Buffer stores in each pixel a list of overlaps that exist there. An overlap is encoded as a list of integers, namely the participating meshes' identifiers preceded by the order of the overlap. This dense encoding would refer to an intersection of, e.g., three meshes with identifiers 23, 24, and 25 as $3-23-24-25$ (an overlap of order three). In the Volume Estimation (compare Section 6.1) this step determines which overlaps exist. We use it there to limit the calculation of depth differences to only those intersections that actually bound a volume. Here, we recreate this step in perspective projection space. Then we download this overlap set buffer to the CPU to later use it to calculate glyph positions.

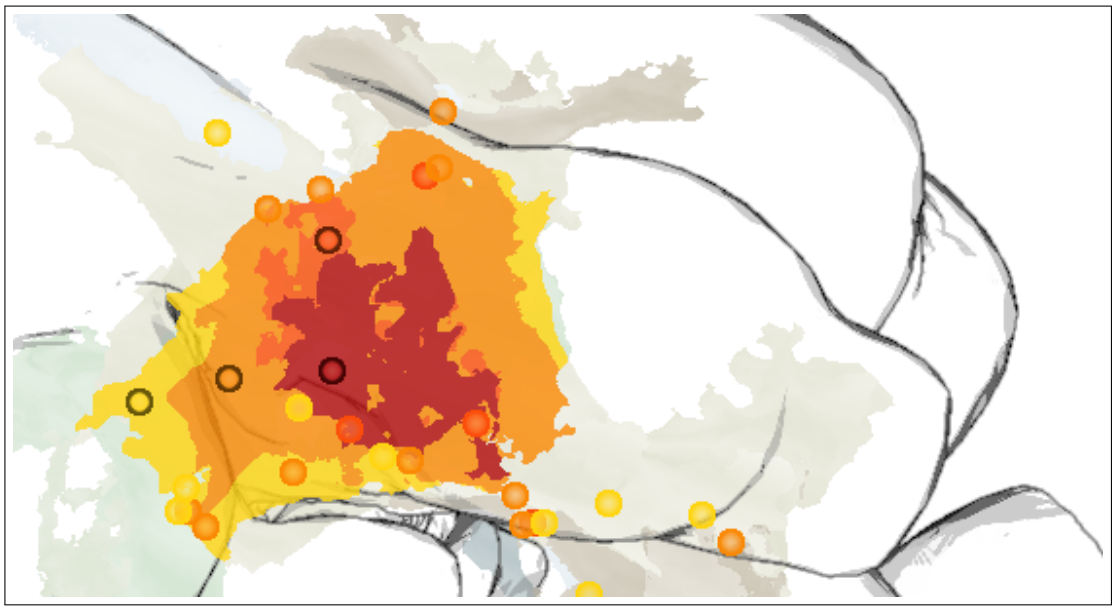
Rendering Overlaps. To render an overlap our shader needs only the overlap set buffer and the identifiers of the meshes constituting the overlap. For each pixel, it searches the A-buffer's linked list to discover if the overlap in question exists. Except for a subtle transparency, the overlaps receive no special shading. The colouring is done according to the design, the colour indicates the order of the overlap. Note that overlaps of higher order are rendered on-top, since they occupy less area (compare Figure 6.8a).

Mesh intersections could be rendered with flat or Gouraud shading, using the normal information from the original meshes. Since the design does require this, the normal information is not retained in the A-buffer. The overlaps intentionally appear in a single colour.

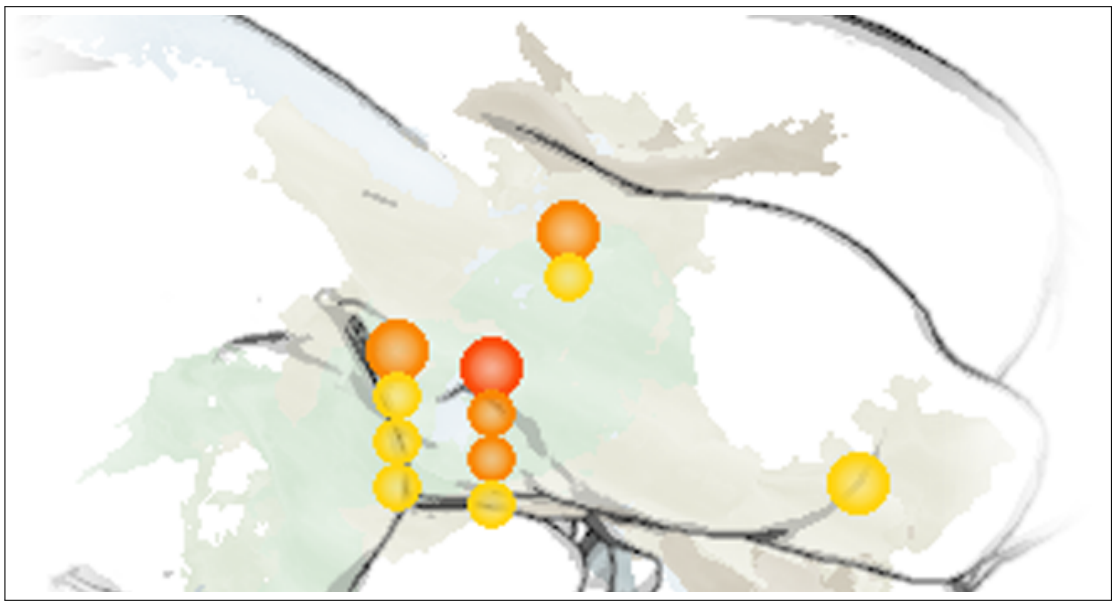
Abstracting Overlaps as Glyphs. Concerning the glyphs it was necessary to find a suitable way to position them in screen space. Two approaches with distinct advantages and disadvantages have been implemented.

Glyph Positioning in 2D. The first attempt to positioning follows a few simple requirements. Glyphs ought to appear on top of the overlap they represent, while keeping their distance to other glyphs, to remain discriminable from each other. Additionally, there should be some sort of position coherence when navigating the rendering by zoom or rotation.

We find glyph positions by calculating a centre point in 2D for each overlap. As described above, the overlap set A-Buffer is downloaded from the GPU. We map this A-Buffer to a 2D grid, i.e. we reduce each of the linked lists to a single overlap–the one with the highest order. This mirrors the overlap rendering (compare Figure 6.8a): we have multiple distinct areas of overlap. To each area, we apply a city block distance transform to determine the centre point. This way each overlap's glyph is set far away from its area's edge.



(a) glyph positioning in 2D



(b) glyph positioning in 3D

Figure 6.8: (a) Four glyphs are selected, the glyphs are positioned in 2D in the centre of the largest continuous 2D area of its overlap. Note that the glyphs are positioned on top of their respective overlaps, away from other overlaps. (b) The glyphs are positioned in 3D. Those that occupy approximately the same 3D position are stacked vertically.

Glyph Positioning in 3D. The second attempt to positioning follows the simple directive that the visualisation must be stable when rotating and panning. To achieve this it was necessary to calculate 3D positions for glyphs. Instead of using the overlap set A-Buffer of the rendering pipeline (Figure 6.7), which has to be downloaded for every frame for the 2D positioning, we use data from the Volume Estimation. The buffers there (compare A-buffer 5 overlap sets in Figure 6.1) are of lower resolution than the screen space but sufficient to provide 3D coordinates for all overlaps.

The 3D coordinates for an overlap should be close to its centre of mass (centroid). We compute this only once for each overlap: Since we already have the A-Buffer from the Volume Estimation which provides depth values for an overlap, we use it here to estimate x, y, and z of the centroid. For a single overlap, looking at all the linked lists sequentially, we find the (x,y) position corresponding to its highest depth difference. Here, we take the depth value of the entry point and add half this depth difference. This value stands in as (z) for the 3D position, the point is most likely inside of the overlap and a good estimation of the centroid.

More than one overlap may return the same or almost the same 3D position. These are mainly overlaps directly related to each other: A four-overlap (the intersection of four meshes) may have a very similar centroid as the three three-overlaps created from subsets of its four meshes. By overlaying a 3D grid all glyphs that are very close are clustered together. The grid is quite coarse (about 25 buckets in each dimension), the clustering is a simple bucketing. The 3D positions in normalised device coordinates are then on the CPU transformed to their screen space position. The glyphs in a single bucket are visualised stacked vertically (compare Figure 6.8b). The top-most glyph in this stack is always the one of highest order at this grid position. It is drawn at its computed 3D position, while the others are drawn beneath in screen space.

Rendering Glyphs. After computing glyph positions with either approach (2D or 3D), they are rendered at their computed positions and composited over all other textures on the canvas. Their colours refer to the order of the respective overlap. The glyphs are rendered with some transparency (approximately 0.7 opacity in the centre, with an opaque border), which makes them visible when unavoidable occlusions occur. The glyphs of selected overlaps are indicated by a surrounding black circle.

Choosing a Glyph Positioning. While the 2D positioning produces beautiful results for static images, which may be used to publish findings, it also has its drawbacks. The major disadvantage of the 2D positioning is certainly the temporal incoherence when rotating the scene. Another disadvantage is overlapping glyphs, especially where overlaps of order four or more exist. This approach does not scale well with very high order overlaps (intersections of four or more meshes).

When rotating the view of the brain to explore the overlaps in more detail, glyphs will jump erratically from one location to another one. This highly distracting behaviour results from calculating centroids in 2D screen space instead of 3D view space. To al-

leviate this problem, we interpolate glyph positions between frames. When the user changes the camera view, the glyphs are not repositioned at their computed 2D position immediately. Instead they lag behind slightly. Within 600ms of the last camera movement they reach their calculated positions. Over this time their movement is decelerated by a cubic Hermite interpolation.

The 3D positioning does not suffer these disadvantages. Instead the stacked layout may require user manipulation of the viewport to clarify where exactly a glyph belongs to. This glyph layout can be overloaded visually just like the 2D positioning if too many overlaps occur, i.e. this glyph layout also does not scale to very high order overlaps. It does, however, perform better in this respect and works fine with overlaps of order five and often six. This problem is however not anticipated with a standard use case, which rarely exceeds order four.

With the stacked layout we also tried collapsing stacks to a single big glyph. This approach was quickly abandoned however as it hides vital information and is not intuitive.

Silhouette Shading

To understand the information in the rendering, a spatial context is required. The design asks for a silhouette of the fruit fly brain used as backdrop to the rendering. In similar tools, this spatial context may be provided by a foggy or transparent [45] rendering of the brain.

The silhouettes of the brain regions are calculated by applying a Sobel filter on a depth buffer, as suggested by Saito and Takahashi [58]: We render a simplified mesh of the brain template, on which all neuronal data has been registered, to a texture. This texture stores only the depth values of the rendering, which we the apply the Sobel filter to. The filter response is used to create a silhouette ranging from dark to light grey, with non-continuous transitions between three grey values. The method achieves real-time performance and closely resembles the initial design (compare Chapter 4).

Neuron and Arborisation Rendering

The colouring style and texture chosen for the arborisations resemble watercolour images (compare the colour design in Section 4.3). Looking for an interactive technique that avoids the shower door effect, we found a technique implemented by Bousseau et al. [9] very useful. In this paper they assembled multiple techniques to create a very convincing watercolour style. Their pipeline includes effects to imitate the paper for watercolour images, and a few different effects to simulate the way watercolour disperses on such a paper: e.g., edge darkening, pigment dispersion, etc.

Out of all the effects they describe, we chose two: the low frequency turbulent flow and the high frequency pigment dispersion. Both of these add some irregularity to the

Figure 6.9: Highlighting an arborisation via a menu results in an abstracted rendering of the complete neuron create spatial context and make it easy for a user to recognise its distinct structure. The image shows one large green arborisation that is not highlighted. A second arborisation is drawn slightly darker, almost grey, because it is highlighted. Note the glyph that indicates an existing two-overlap between these arborisations. Due to this arborisation highlighting, its complete neuron is rendered in an abstract way: Multiple lines emerge from it, these are all projections of this neuron. At the end of these, note a large dot, this is an abstract representation of the neuron's cell body. In this instance the projection connecting the cell body and its arborisation is drawn four times because our database contains four slightly different instances of this neuron.

colouring of our arborisations to imitate real water colouring. This results in a rendering with limited depth and structure information. As intended, the arborisations look flat and unobtrusive to keep the visual focus on the overlaps (compare Figure 6.9).

Arborisations are rendered from a sorted A-buffer. To realise the described water-colour effects the A-buffer must store, in addition to mesh identifiers and depths, three float values for the model space vertex positions. Then, a shader samples a Perlin noise texture to apply both of these colour irregularities. By using model space coordinates to sample the noise textures, the camera may be rotated without creating the so-called shower door effect.

A single arborisation at a time can be highlighted in the viewport, thus rendering it a shade darker to make it stand out. To create more spatial context for a highlighted arborisation, its complete neuron is drawn in an abstract way (see Figure 6.9): All its projections are drawn with a constant diameter, and its cell body is rendered, abstracted as a large dot. The dot is placed in the centre of the cell body bounding box.

Neuropil Rendering

The brain of *Drosophila melanogaster* is divided into 51 brain regions, called neuropils. Overlaps between arborisations happen inside one or more neuropils. The silhouette we draw is an abstraction of the outer shell of the entire brain, the union of all 51 neuropils. We draw our neuropils with a blueish hue, their rendering comes close the intended design (compare Figure 6.10 and Figure 4.10a).

One of the rendering passes is dedicated to creating a neuropil rendering, which is composited to the final image. Compare Figure 5.3 for a simple illustration of the composition. Via the menus neuropils can be selected to trigger rendering in the viewport. They are the only feature to use a high transparency. In the design, we intentionally used transparency sparsely: Glyphs and context menus have an opacity slightly lower than one, but the only feature to use very low opacity (here we use the value 0.2) is the neuropil rendering. The rendering is done in a separate pass and contributes to the final compositing (compare Figure 6.7). Note that the rendering uses mesh normal information to create a 3-dimensional look, as opposed to the arborisations' flat look.

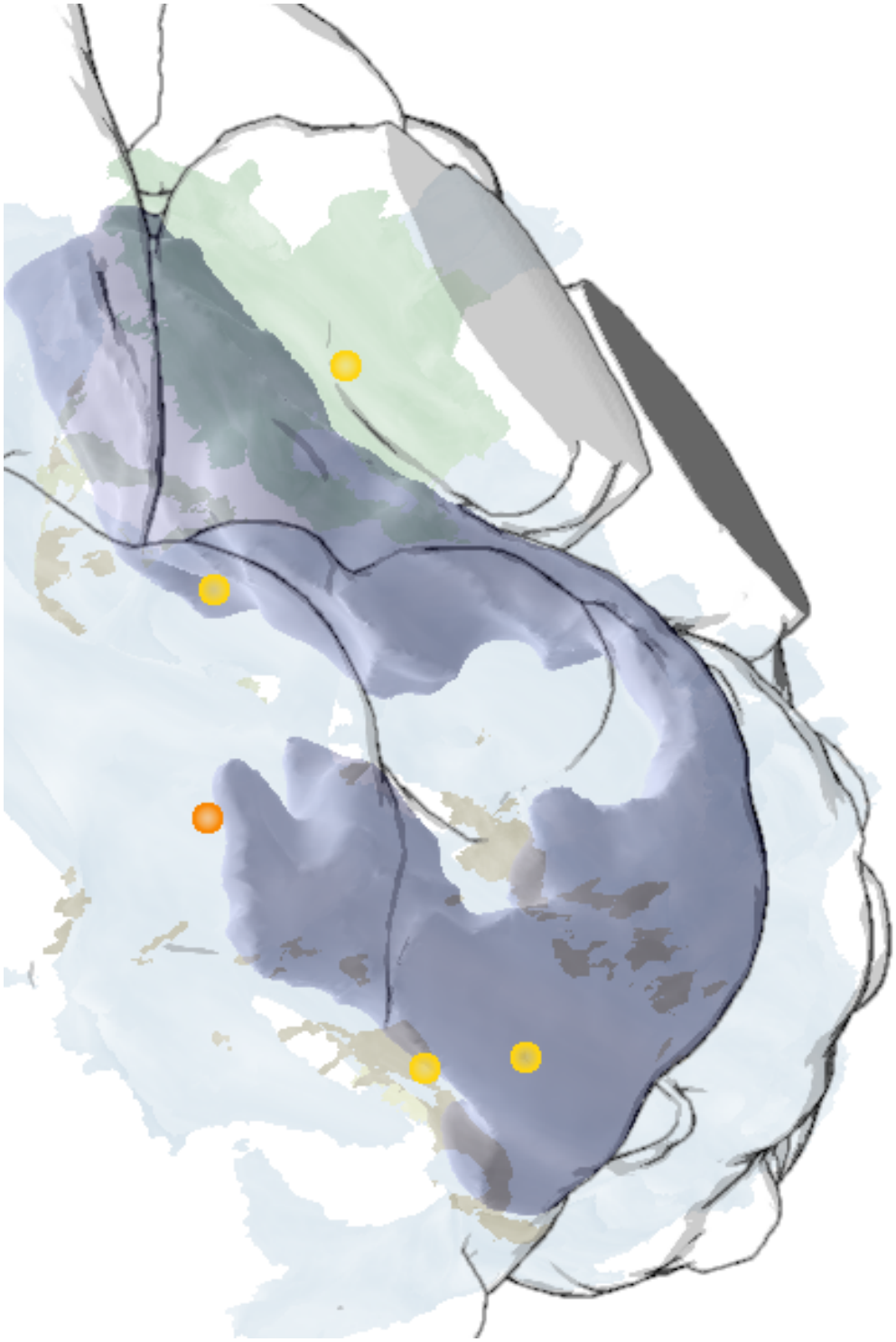


Figure 6.10: Selecting neuropils via the menu triggers their rendering. Neuropil rendering is intentionally the only feature in the system that uses a large amount of transparency. They are rendered with this blueish colour and use the mesh normal information to achieve a 3-dimensional impression, as opposed to the arborisations.

CHAPTER 7

Interaction

The carefully designed interaction elements work in unison with the 3D visualisation where essential information is embedded in the form of glyphs. This chapter describes our design choices concerning interaction, both in the 3D visualisation and in the linked menus. For a video of the implemented interactivity, the interested reader can look up the supplemental materials published with the CGF paper [69].

The first section summarises the capabilities of the 3D viewport. Then follows a long section on glyph interactions. Glyphs are the central interaction element, the section details mouse-over and selection behaviour. We explain in detail the interaction options offered by the two types of menus, the tree menu and the tooltip menu. Both of these provide quantitative information not directly conveyed in the 3D visualisation.

The third section details selection of neuronal structures. It discusses the interactive linking with other tools in the BrainGazer framework. The fourth and last section explains how we filter overlaps that the user deems unimportant.

Design choices and their realisations are discussed in Chapters 4 and 6 respectively. The interaction design choices concerning selection of multiple neuronal structures are discussed in this chapter. The tool follows the primary interaction concept of linked views/multiple integrated views. These are all centred around glyph interaction. The glyphs abstract overlaps as discussed in Section 4.4. They are also used to blend in overlaps in 3D or show quantitative information as a tooltip, directly integrated in the viewport.

7.1 Exploration in 3D

Navigation in 3D must work with interactive frame rates for a visualisation described by the design. The viewport offers this interactive exploration in 3D of arborisations and

their overlaps. Neuronal structures can be loaded (e.g., by drag-and-drop) and investigated in detail: the view can be zoomed, panned and rotated fully by traditional mouse interaction.

7.2 Glyph Interaction

The glyphs not only encode overlap information, they are also the central interaction element. The glyphs must offer direct and immediate interaction. On interaction, a glyph must display its overlap, and show quantitative information on demand. Glyph interaction must be linked with other views on the same data.

The following three subsections detail how the glyph interaction works with the three distinct views we implemented: the 3D visualisation, the tooltip menu, and the tree menu. Compare the design chapter, specifically Section 4.4 for an explanation of glyphs and these three views.

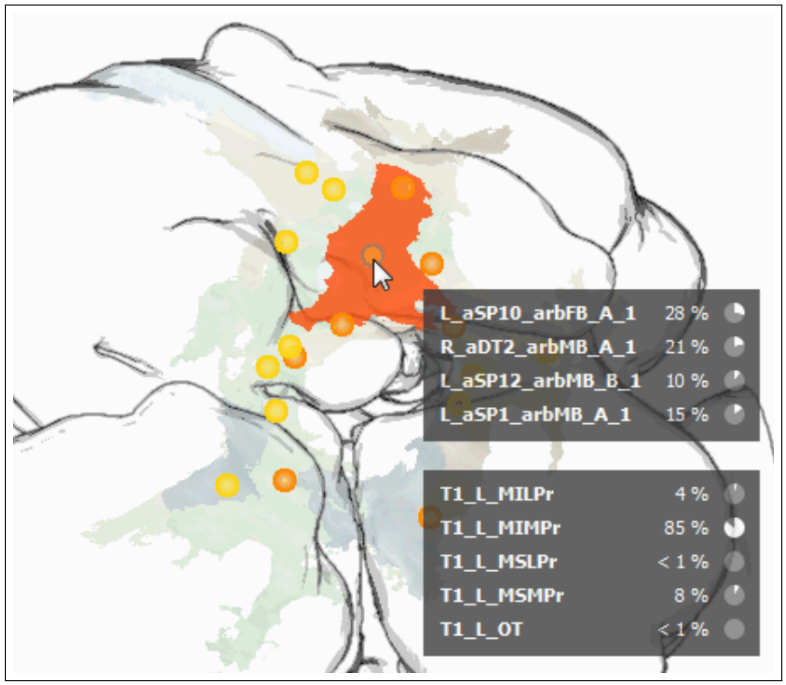
3D Visualisation

The integration of the interaction into the glyphs makes it possible to gain valuable insights inside the 3D visualisation itself. Frequent switching between different views on the data becomes unnecessary. The number of existing overlaps is immediately observable as well as their approximate locations, promptly giving indicators for two of the three scientific questions (compare Section 2.4).

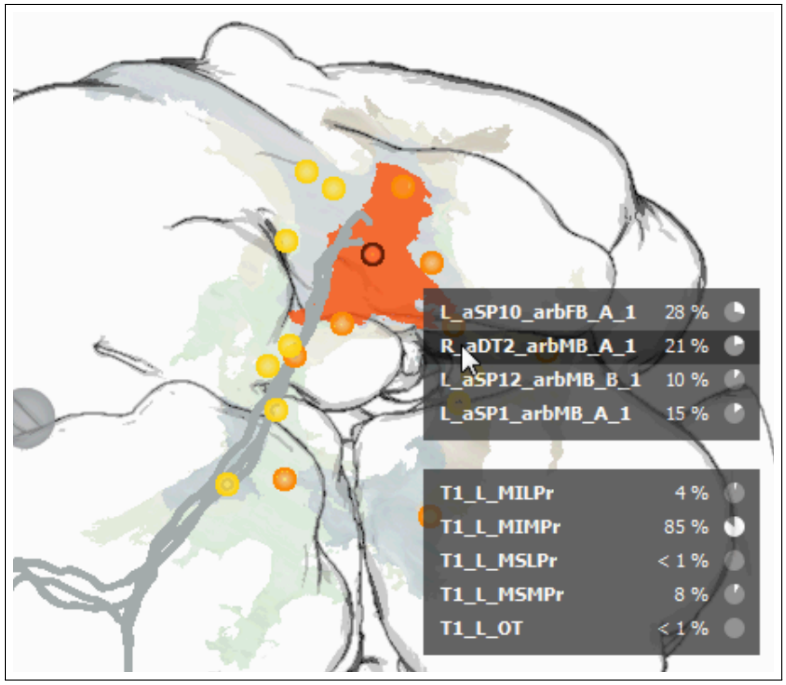
As discussed in previously in Section 6.3, the glyphs move with the objects when we change the camera position, to always stay on top of their respective overlaps. An overlap is immediately displayed when the user simply moves the mouse on top of the corresponding glyph (compare Figure 7.1a). The overlap can be selected by left-clicking the glyph, this renders the glyph with the pronounced surrounding black circle and shows the overlap (compare Figure 7.1b). These two interactions (mouse-over shows the overlap, left-click adds it to the selection to keep it visible) are intuitive when the user gets immediate visual feedback. Because of the real-time performance, the link between glyph and overlap becomes obvious.

Tooltip Menu

Hovering with the mouse over a glyph opens an interactive tooltip menu (compare Figure 7.1), which provides in-depth information on the overlap. For a single overlap this information answers which neurons are participating. It also gives quantitative information to render a judgement on the significance of the overlap (compare scientific questions in Section 2.4).



(a) glyph mouse-over



(b) arborisation highlight

Figure 7.1: (a) The tooltip menu pops up, when the user hovers a glyph for a second. (b) The user moves the mouse across the arborisation R_aDT2_arbMB_A_1 (21% of it participate in this overlap) in the tooltip menu. This interaction immediately highlights the arborisation. Moving the mouse away from a tooltip entry removes the highlight.

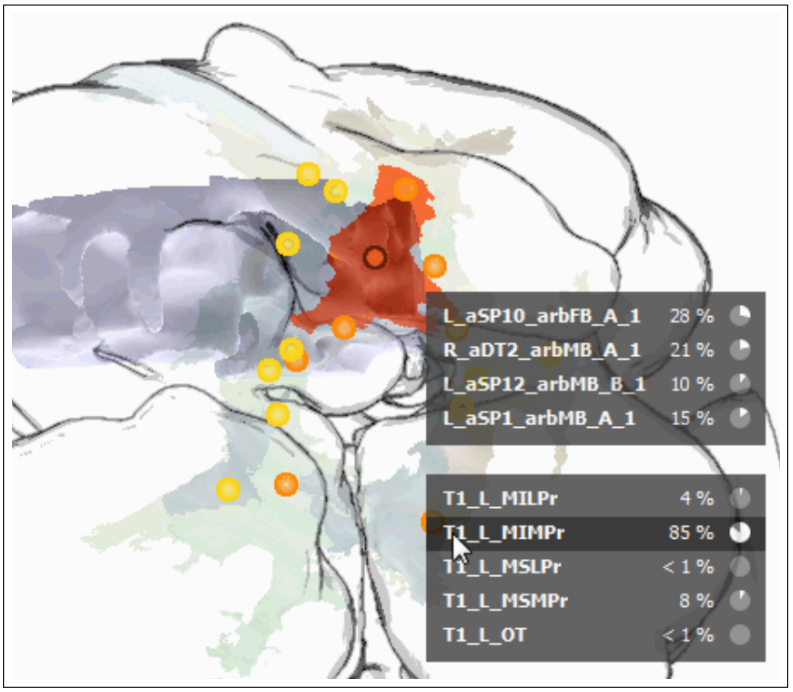
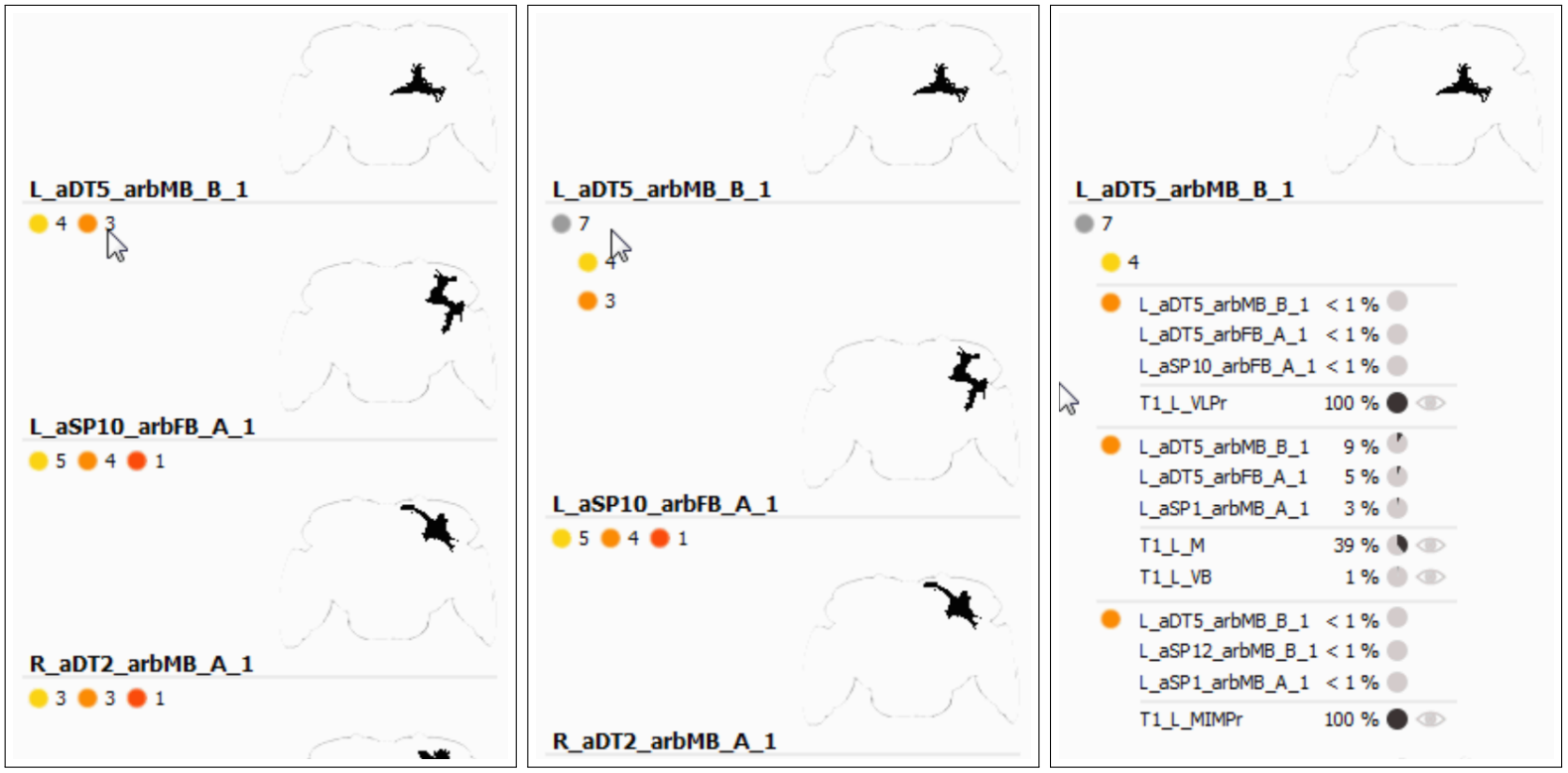


Figure 7.2: Continuation of the figure on the preceding page: The user continues moving the mouse across the tooltip menu. Here, we hover over the neuropil T1_L_MIMPr (85% of the overlap are located inside it). This interaction immediately triggers its rendering. Moving the mouse away from a tooltip entry removes the rendering again.

The on-demand integration of the tooltip into the viewport is a straight-forward implementation of an *overview-plus-detail* interaction concept, which is used virtually everywhere when dealing with some complexity.



(a) (b) (c)

Figure 7.3: (a) The tree menu shows arborisation L_aDT5_arbMB_B_1 participates in seven overlaps. Here, we left-click on these glyphs to expand the tree menu (b). Another click on the orange glyph (overlaps of order three) expands (c) the tree menu to show detailed information on each of these overlaps.

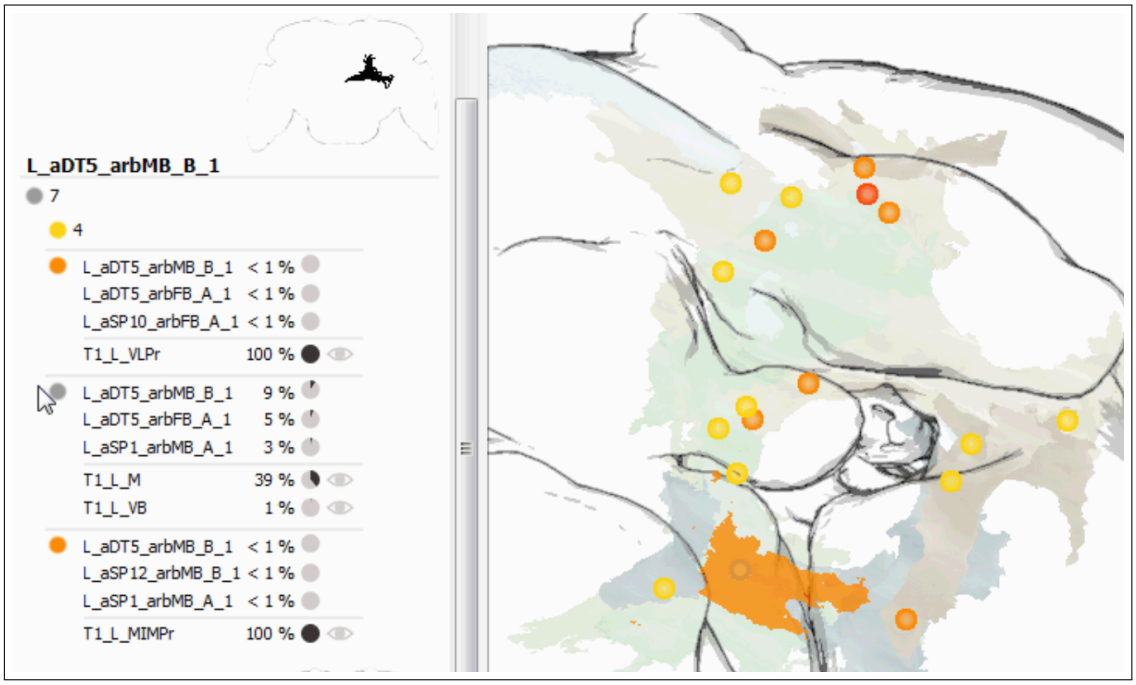


Figure 7.4: The tree menu allows some mouse interaction with glyphs. Here, we move the mouse over a glyph in the tree menu, the 3D visualisation immediately renders the corresponding overlap. Analogous to the tooltip menu, this mouse-over behaviour works with the listed arborisations and neuropils. Also, a left click on a glyph here selects the overlap, in the same way as clicking a glyph in the 3D visualisation does.

Tree Menu

The tree menu lists, for each arborisation, all existing overlaps. It can be expanded and collapsed by left-clicking glyphs. The glyphs in the tree menu appear in the same shape and shade as the corresponding glyphs in the 3D visualisation.

Figures 7.3 and 7.4 illustrate glyph interaction in the tree menu. Once the tree is expanded, the glyph interaction works similar to the glyph interaction in 3D: a mouse-over shows the overlap immediately, a left-click selects the overlap. This mouse-over highlight and selection action is linked to the 3D visualisation, i.e. when a user selects an overlap, both corresponding glyphs in the 3D viewport and in the tree menu are rendered with the distinctive black surrounding circle to indicate a selection. This consistency of the glyph interaction in the tree menu and in the 3D visualisation helps to make the interface intuitive.

7.3 Selection of Neuronal Structures

An integral part of this tool and the BrainGazer framework (see Section 5.1) is the selection interaction. Users typically select multiple neuronal structures to take advantage of linked views, i.e. the selection of a structure in one tool is automatically carried over to other tools in the framework. This includes other visualisation tools such as neuroMap (compare Section 3.2 and Figure 3.3), 3D rendering viewports, slice rendering viewports, and heatmaps. It also includes the BrainGazer framework's workspace tool, which manages data loading and communicates selection information across all other tools. With many different views on the same data, the user gains a deeper insight to the data.

This section details how our tool handles the selection of neuronal structures. The initial design (see Chapter 4) does not exhaustively cover how the selection system should work. With some trial-and-error and constructive feedback from biologists, we implemented the following modes of selection.

Selection of Neuropils

Selected neuropils are rendered in the 3D visualisation to give additional spatial context. The selection and de-selection of neuropils is only possible via the tree menu. The tree menu displays the volumetric data and offers buttons to toggle neuropil rendering permanently on or off (see the small eye icons in Figure 7.4 next to each neuropil entry). While the tooltip menu offers the option to highlight a neuropil (compare Figure 7.2), it does not give the option to select it to make this rendering permanent. This option was intentionally omitted from the tooltip to keep it uncluttered.

Selection of Arborisations

The tree menu gives the option to select one or more arborisations. The user can left-click an arborisation's name or its icon (see Figure 7.3a). The 3D viewport visualises selected arborisations in the same way as highlighted arborisations: The complete neuron is rendered, to give more spatial context (compare Figure 6.9). The selection of arborisations is communicated to other tools in the BrainGazer framework bi-directionally via the workspace tool.

Selection of Overlaps

The workspace tool in the BrainGazer framework limits the selection of overlaps to a single overlap. The way the selection of a pairwise overlap is communicated to other tools, is by selecting the two participating arborisations. Other tools can then inter-

pret this as an overlap selection. Because of this selection logic, it is not possible to communicate the selection of two or more overlaps to other tools in the framework.

In our tool, however, the selection of multiple overlaps is certainly possible, and necessary, e.g., to inspect and compare them in 3D. As discussed, overlaps can be selected by clicking glyphs in the 3D visualisation or in the tree menu. In both views selected glyphs are drawn with an enclosing black circle, all selected overlaps are rendered. Clicking such a marked glyph deselects it, a mouse click on some whitespace in the 3D visualisation drops the whole selection at once.

7.4 Filtering

During the evaluation process (see Chapter 8) we received plenty of feedback from users experienced in the field of neurobiology. One crucial request pertains to the so-called unimportant overlaps. As per the definition of Peters' Rule (compare Section 2.2), overlaps that are small in absolute or relative (to their arborisations) size are not important.

Even extremely small overlaps produce glyphs. In a practical use case with about ten arborisations, this can already lead to a clutterd view in the 3D visualisation. We implemented four filter options (compare Figure 7.5). Overlaps, which match one (or more) of these, are completely removed from the interface:

1. The first filter matches highest relative volume. If the highest percentage (for the participating arborisations) is below the lower threshold (or above the upper threshold), it matches. For instance, a lower threshold of 5% filters overlaps that do not have any arborisation which participates with $> 5\%$.

2. Another filter matches the lowest relative volume. It works analogous to the first filter, but instead of checking the highest percentage for participating arborisations, it checks the lowest percentage. For instance, a lower threshold of 5% filters all overlaps that include an arborisation which participates only with $\leq 5\%$.

3. The third threshold filter tests absolute volume of an overlap. It also offers a lower and an upper bound, but the values the user sets here are in μm^3, not in percent. Any overlaps not within the set bounds are filtered out.

4. The last filter does not target the so-called unimportant overlaps. It gives the option to filter all overlaps based on their order. If a user is not interested in, e.g., overlaps of order two at all, they can left-click the yellow glyph in the filter interface to cross it out. This action removes all pairwise overlaps from the user interfaces.

This filtering vastly improves the scalability of the whole interface regarding the number of loaded arborisations. A reasonable default setting for 'highest relative vol-

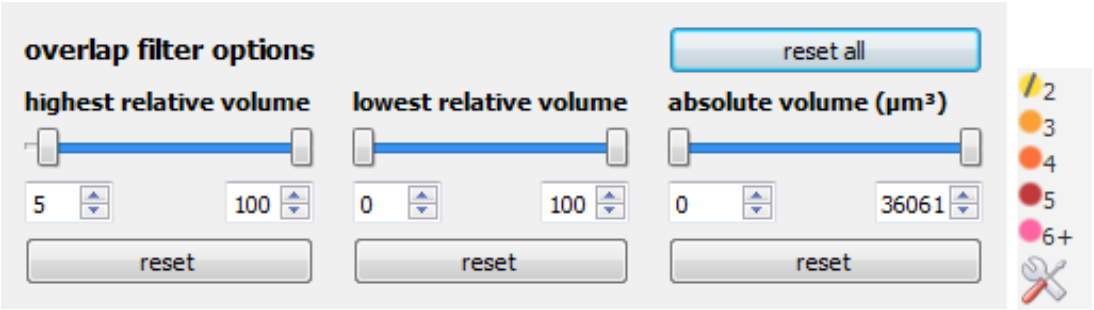


Figure 7.5: These filters remove unimportant glyphs from the visualisation. The user may define upper and lower thresholds to remove what they consider unimportant overlaps. The filter menu to the very right acts as a legend for the glyphs (pairwise overlaps are yellow, three-overlaps orange, etc.). Each glyph icon there may be toggled on or off to filter all glyphs of this colour/order. The first filter (highest relative volume) has its lower threshold set to 5% – our default value which, according to experts, removes most so-called unimportant overlaps.

ume' (lower threshold at 5%) already combats most of the clutter that experts criticised during the feedback sessions. The figures we use to show glyph positioning in Section 6.3 demonstrate this very well: Figure 6.8a was made without any filtering and shows glyphs for all 26 overlaps in the area. Figure 6.8b shows only 11 glyphs, because the default filter identified 15 unimportant overlaps.

CHAPTER 8

Evaluation

We performed a quantitative and qualitative evaluation of the software to gain insight into its usability and usefulness by assessing effectivity, efficiency, and user satisfaction. The following sections cover the evaluation process and the discussion of its results.

8.1 Evaluation Setup

The evaluation was performed in one-on-one interviews with a small group of test persons. The interviews were guided by tasks that the users had to perform.

Test Persons We limited the number of test users to ten to be able to perform intensive user feedback sessions. Based on their background knowledge, we considered five persons experts and five persons non-experts. Two of the five experts were highly experienced post-docs with strong background in neurocircuit research. Three expert users had a bioimage informatics and visualisation background and decent knowledge on workflows related to neurocircuit research. Non-expert users had no background on neurocircuit research, but varying experience with user interfaces and 3D tools in general. Both user groups gave useful feedback.

Test Setup Tests were performed in front of a computer in our lab running the system. All test persons received a brief introduction into the test setup and—if necessary—the application background of the tools. During the test previously defined scenes were loaded into the system to provide unified starting points for all users. The users could freely interact with the system and were guided through the test by a set of tasks and questions. On occasions they received hints on how to proceed. For test persons that were not able to come to our lab in person, we provided remote access to an installation of our system on a GPU instance of Amazon's Elastic Cloud. This allowed interviewer

and test person to observe and interact with the system while performing the interview via telephone.

8.2 Quantitative Evaluation

Following the test, we asked the users to fill out the *User Experience Questionnaire* [38], a catalogue of 26 questions developed by a group of usability experts to quantitatively measure user experience in a simple and immediate way. While evaluating the results, however, we learned that this questionnaire does not work well with very few test persons. Due to the low number of test persons we believe these results to be, although very favourable, of low reliability and thus skip them in this evaluation. It should be mentioned that given the resources for testing the software at a large scale, this evaluation method would likely prove useful.

8.3 Qualitative Evaluation

Testing was split into two tasks with an optional prior introduction to the system for novices. These unspecific tasks let the interviewees explore the system freely:

1. Investigate a pairwise overlap.
2. Investigate multiple overlaps, including higher order overlaps.

Both task were set up by the interviewer, with arborisations pre-loaded into the workspace. The users were instructed to try out the user interface on their own. During the evaluation and afterwards, the interviewer asked detailed questions on the user interface and overall questions about perspicuity and efficiency. For non-expert users we did not explain the domain extensively. Instead, we briefly stated they were to look at overlaps between meshes inside the brain. Few of the non-experts required a short introduction prior to the testing, consisting of explanations regarding the basic concepts that the biologists investigate.

We followed the *think-aloud method* [40] to capture thoughts and feeling of the test persons while they interacted with the system. On agreement by the test person the interview was recorded for later transcription.

8.4 User Feedback

The accumulated feedback from the interviews boils down to a few important issues summarised by the following four captions. The section following this one discusses the more vital issues raised here.

Connectivity Exploration As expert users stated, analysis of overlaps is commonly done by looking at 2D slices and/or pre-calculated intersection volume data derived from 3D segmentation masks. Their unanimous impression is that our tool accelerates both finding new overlaps and analysing them. One expert made it clear that the possibility to investigate higher order overlaps in detail for the first time may spawn new research questions in the future.

Visual Design and Colour Scheme The colour scheme and overall visual design was highly praised by the users. According to most users, the focus and context visualisation realised by the reduced representations of brain, neuropils, and neurons supports perfectly the search for higher order overlaps.

One expert user criticised the colouring of arborisations as too faint but especially liked arborisation highlighting. This highlighting includes blending in the corresponding neuron's cell body and connecting projection(s). As such it was appreciated by the expert users. All non-experts requested an explanation for this unexpected additional context of a highlight action.

Most test persons agree that the colour scheme for glyphs is intuitive. All but one non-expert users understood the colour coding after investigating glyphs for some time on their own.

Interaction Interacting with glyphs was considered intuitive by all users. Three expert users were irritated by erratic movements of glyphs when rotating the view. Of those, one addressed glyphs that occlude each other. This happens where very small overlaps cluster. The same small overlaps are not immediately discernible in the rendering viewport when selecting them in the tree menu, as one non-expert user mentioned. Three expert users want this solved by filtering out unimportant overlaps altogether, e.g., by manually selecting a percentage threshold. Two more experts would like to filter by order of overlap. Note that this feedback prompted our implementation of filters, compare Section 7.4.

Both the tree menu and the tooltip menu were considered overall to be perspicuous. One non-expert did not find them intuitive at all and recommends opening the tooltip menu like a context menu by right clicking the mouse. Another non-expert wants the tooltip to open on left clicks. Other than that the users intuitively used the tooltip as intended. Some users from both groups would like the process of expanding and collapsing in the tree menu to be guided by arrows. One expert user and, as expected, most non-experts are confused by the percentages listed in the menus.

Selection Model Experts welcomed the selection of overlaps across different tools. Non-experts, some of whom do not have any knowledge of those other tools, assumed that biologists would appreciate the functionality. The users liked that the tree menu and viewport are linked and consistent in using the same glyph symbols to represent overlaps.

The users quickly adjusted to the selection and de-selection process, despite different

expectations of some. This includes the selection of multiple overlaps via the menu or viewport.

One non-expert assumed multiple overlap selection would work by pressing the control button while clicking. One expert did not like that selecting more than one overlap at once was even possible. These two users criticised that cross-selection with other tools is only possible with one overlap at a time, as this is inconsistent with multiple overlap selection inside our tool.

8.5 Discussion

In conclusion, the qualitative evaluation was very positive. All users with intimate knowledge of different common workflows report that our tool speeds up and improves the analysis of higher overlaps in comparison. The quantitative evaluation supports this claim of efficiency although we refrain from publishing this unreliable data (compare Section 8.2). The positive feedback is specially valuable coming from two users who are experts in the domain of neurocircuitry. They use different workflows in their daily life and see a clear benefit in using the proposed tool. One such workflow, as described by one of the test persons, is to manually scan through slices of rendered arborisations. In these slices overlaps of arborisations are outlined. In combination with quantitative data—available for pairwise overlaps only—this workflow is very limited and time consuming.

Another field of improvement, revealed by the evaluation, concerns the glyph movement when changing the camera view. Sometimes, the glyphs appear to erratically jump in between frames. Prompted by this feedback, we improved this issue in two ways. First, the 3D positioning for glyphs (see Section 6.3) drastically reduced the erratic movement, compared to the 2D positioning. The second beneficial measure we took is the introduction of a default setting for the overlap filters. The removed unimportant glyphs can not distract with their movement. Nevertheless, temporal coherence with glyph positioning will have to be addressed in the future.

The selection model has some inconsistencies, especially with respect to the cross-selection. So far we have no perfect logic approach to solve the cross-selection across many views. Using the control key to enable the selection of multiple items (overlaps, arborisations, etc.), as realised in a multitude of other tools, may be a good approach. Any new selection model must be very clear how different types of selections are handled across multiple tools. At the least, it has to:

1. Clearly differentiate types of selected objects (arborisations, neuropils, complete neurons, overlaps, etc.).

2. Allow or disallow the selection of these different types of objects in a transparent way.

3. Allow or disallow the selection of multiple objects, and even multiple overlaps, in a transparent way.

4. Be consistent across all tools and, where it cannot, be transparent about tool limitations. The user must be informed if the selection mapping between tools is not 1:1.

Our selection model works well. Especially regarding point four, however, it still lacks clarity. To satisfy all these points, we will have to extend the selection model built into the BrainGazer framework.

The fact that even non-experts were able to use the tool with barely any guidance, shows the intuitiveness of the tool. This is especially appreciated with respect to the glyph colour scheme and glyph interaction, which were quickly picked up on. The experts believe the tool delivers: It helps to find answers to the scientific core questions (Section 2.4). They praised the convenience and speed of the tool, compared to previous workflows. The overall positive feedback gave hints on further small improvements and minor missing features, some of which are discussed in the following chapter. One example is a colour picker for arborisations. We decided against adding this minor feature however, because it would break the carefully crafted colour scheme.

CHAPTER 9

Conclusion and Future Work

9.1 Conclusion

The thesis presented the implementation of a new design by Judith Moosburner [50]. Both the visualisation and the on-demand volume computation enable the analysis of overlaps of arbitrary order.

The qualitative evaluation suggests that the described tool is both effective and efficient in aiding the required tasks. Our benchmarks (compare Table 6.1) show that the algorithm is sufficiently accurate and fast to fulfil the required computations. The design was developed in collaboration with domain experts. In the course of the evaluation, the neuroscientists confirmed that the visualisation and interaction features achieve the tool's aim: to support hypothesis formulation by answering the core questions (Section 2.4). With the on-demand computation of volumes it quickly answers questions on number, location, and significance of overlaps. For the first time neuroscientists have a tool to calculate this potential connectivity between multiple neurons, and interact with it in a spatial context.

The positive feedback on the design encourages us to follow a similar process in the future, i.e. to have a graphics designer lead the design process. Judith Moosburner's artistic perspective on the topic and associated data enriched the technical and biological aspects of the design in a substantial manner. Technical limits barely influenced the process. Without such limitations in mind the unbound experimentation lead to an innovative novel design.

9.2 Future Work

Future efforts will explore options to extend the glyph interaction. This includes improving filtering techniques, better temporal coherence for glyph positioning, and encoding overlap importance as glyph size.

The issue of positioning 2D overlays in a 3D context in general will come up again in the future. The two approaches that we implemented may not be optimal for other projects. Some current and future solutions may include force-based layouts or particle systems with animations and multi-level abstractions to avoid clutter of too many 2D overlays. This includes the apdaption of a solution to place labels in 3D, a topic with a vast amount of literature resources.

The work done here for *Drosophila melanogaster* will be extended to other species. Comparisons of neural circuits between different life stages of a species and even between species is of interest to the biologists. There are currently no tools available to find and visualise higher order arborisation overlaps across developmental stages and species. The extension to other model organisms and the advance of imaging techniques brings new challenges. By now there exist well annotated images from electron microscopy of the fruit fly such as the early larval stage data [24]. Exploring and comparing this multi-modal data will be one of the major challenges in the near future.

Bibliography

[1] Janet Abrams and Peter Hall. *Else/where: mapping new cartographies of networks and territories*. University of Minnesota Design Institute Minneapolis, 2006.

[2] Ali Al-Awami, Johanna Beyer, Hendrik Strobelt, Narayanan Kasthuri, Jeff Lichtman, Hanspeter Pfister, and Markus Hadwiger. Neurolines: A subway map metaphor for visualizing nanoscale neuronal connectivity. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2369–2378, 2014.

[3] Bilal Alsallakh, Wolfgang Aigner, Silvia Miksch, and Helwig Hauser. Radial sets: Interactive visual analysis of large overlapping sets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2496–2505, 2013.

[4] Bilal Alsallakh, Luana Micallef, Wolfgang Aigner, Helwig Hauser, Silvia Miksch, and Peter Rodgers. Visualizing sets and set-typed data: State-of-the-art and future challenges. In *Eurographics conference on Visualization (EuroVis)– State of The Art Reports*, pages 1–21. Eurographics, Eurographics, 2014.

[5] Amira. `http://www.amira.com`. Accessed: 2020-02-25.

[6] Salil S. Bidaye, Christian Machacek, Yang Wu, and Barry J. Dickson. Neuronal control of drosophila walking direction. *Science*, 344(6179):97–101, 2014.

[7] James E. Blankenship and Becky Houck. Nervous system (invertebrate). *AccessScience*, 2012.

[8] Hartmut Bohnacker, Benedikt Groß, Julia Laub, and Claudius Lazzeroni. Generative gestaltung. *Verlag Hermann Schmidt*, 2009.

[9] Adrien Bousseau, Matt Kaplan, Joëlle Thollot, and François X. Sillion. Interactive watercolor rendering with temporal coherence and abstraction. In *Proc. NPAR*, pages 141–149, 2006.

[10] BrainBase. `https://implegacy.brainbase.at`. VRVis Zentrum für Virtual Reality und Visualisierung, Accessed: 2020-02-25.

[11] BrainGazer. http://www.braingazer.org. VRVis Zentrum für Virtual Reality und Visualisierung, Accessed: 2020-02-25.

[12] Valentino Braitenberg and Almut Schüz. *Cortex: statistics and geometry of neuronal connectivity*. Springer Berlin, 1998.

[13] Cynthia Brewer. *Designing Better Maps: A Guide for GIS Users*. Environmental Systems Research, 2004.

[14] Loren Carpenter. The a-buffer, an antialiased hidden surface method. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, pages 103–108, 1984.

[15] Ann-Shyn Chiang, Chih-Yung Lin, Chao-Chun Chuang, Chang-Huain Hsieh, Chang-Wei Yeh, Chi-Tin Shih, Jian-Jheng Wu, Guo-Tzau Wang, Yung-Chang Chen, Cheng-Chi Wu, Guan-Yu Chen, Yu-Tai Ching, Ping Lee, Chih-Yang Lin, Hui-Hao Lin, Chia-Chou Wu, Hao-Wei Hsu, Yun-Ann Huang, and Jenn-Kang Hwang. Three-dimensional reconstruction of brain-wide wiring networks in drosophila at single-cell resolution. *Current Biology*, 21(1):1–11, 2011.

[16] Marina Chicurel. Databasing the brain. *Nature*, 406(6798):822–825, 2000.

[17] Christopher Coffin and Tobias Höllerer. Interactive perspective cut-away views for general 3D scenes. In *IEEE Symposium on 3D User Interfaces*, pages 25–28. IEEE, 2006.

[18] Cyril Crassin. Opengl 4.0+ abuffer v2.0: Linked lists of fragment pages. http://blog.icare3d.org/2010/07/opengl-40-abuffer-v20-linked-lists-of.html, 2010. Accessed: August 13, 2013.

[19] Vincent J. Dercksen, Robert Egger, Hans-Christian Hege, and Marcel Oberlaender. Synaptic connectivity in anatomically realistic neural networks: Modeling and visual analysis. In *Eurographics Workshop on Visual Computing for Biology and Medicine*, pages 17–24, 2012.

[20] Barry J. Dickson. Wired for sex: the neurobiology of drosophila mating decisions. *Science*, 322(5903):904–909, 2008.

[21] Joachim Diepstraten, Daniel Weiskopf, and Thomas Ertl. Transparency in interactive technical illustrations. *Computer Graphics Forum*, 21(3):317–325, 2002.

[22] Feng Dong, Gordon J. Clapworthy, Hai Lin, and Meleagros A. Krokos. Nonphotorealistic rendering of medical volume data. *IEEE Computer Graphics and Applications*, 23(4):44–52, 2003.

[23] Rodney J. Douglas and Kevan A. C. Martin. Neuronal circuits of the neocortex. *Annual review of neuroscience*, 27:419–451, 2004.

[24] Katharina Eichler, Ashok Litwin-Kumar, Feng Li, Youngser Park, Ingrid Andrade, Casey M. Schneider-Mizell, Timo Saumweber, Annina Huser, Claire Eschbach, Bertram Gerber, Richard D. Fetter, James W. Truman, Carey E. Priebe, L. F. Abbott, Andreas S. Thum, Marta Zlatic, and Albert Cardona. The complete connectome of a learning and memory center in an insect brain. *bioRxiv*, 2017.

[25] Niklas Elmqvist, Ulf Assarsson, and Philippas Tsigas. Employing dynamic transparency for 3D occlusion management: Design issues and evaluation. In *Human-Computer Interaction – INTERACT 2007*, pages 532–545. Springer Berlin Heidelberg, 2007.

[26] Niklas Elmqvist and Philippas Tsigas. A taxonomy of 3D occlusion management for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(5):1095–1109, 2008.

[27] Steven K. Feiner and Dorée Duncan Seligmann. Cutaways and ghosting: satisfying visibility constraints in dynamic 3D illustrations. *The Visual Computer*, 8(5-6):292–302, 1992.

[28] Mehmet Fişek and Rachel I. Wilson. Stereotyped connectivity and computations in higher-order olfactory neurons. *Nature Neuroscience*, 17(2):280–288, 2014.

[29] IMP - Research Institute of Molecular Pathology GmbH.

[30] Kei Ito, Kazunori Shinomiya, Masayoshi Ito, J Douglas Armstrong, George Boyan, Volker Hartenstein, Steffen Harzsch, Martin Heisenberg, Uwe Homberg, Arnim A Jenett, Haig Keshishian, Linda L Restifo, Wolfgang Rössler, Julie H Simpson, Nicholas J Strausfeld, Roland Strauss, Leslie B Vosshall, and Leslie B Vosshall. A systematic nomenclature for the insect brain. *Neuron*, 81(4):755–765, February 2014.

[31] Johannes Itten. *Kunst der Farbe. Studienausgabe. Subjektives Erleben und objektives Erkennen als Wege zur Kunst.* Urania, Stuttgart, 2003.

[32] Arnim Jenett, Gerald M. Rubin, Teri-T.B. Ngo, David Shepherd, Christine Murphy, Heather Dionne, Barret D. Pfeiffer, Amanda Cavallaro, Donald Hall, Jennifer Jeter, Nirmala Iyer, Dona Fetter, Joanna H. Hausenfluck, Hanchuan Peng, Eric T. Trautman, Robert R. Svirskas, Eugene W. Myers, Zbigniew R. Iwinski, Yoshinori Aso, Gina M. DePasquale, Adrianne Enos, Phuson Hulamm, Shing Chun Benny

Lam, Hsing-Hsi Li, Todd R. Laverty, Fuhui Long, Lei Qu, Sean D. Murphy, Konrad Rokicki, Todd Safford, Kshiti Shaw, Julie H. Simpson, Allison Sowell, Susana Tae, Yang Yu, and Christopher T. Zugates. A GAL4-driver line resource for drosophila neurobiology. *Cell Reports*, 2(4):991–1001, 2012.

[33] Radu Jianu, Cagatay Demiralp, and David H. Laidlaw. Exploring brain connectivity with two-dimensional neural maps. *IEEE Transactions on Visualization and Computer Graphics*, 18(6):978–987, 2012.

[34] Marcus Kaiser. A tutorial in connectome analysis: Topological and spatial features of brain networks. *NeuroImage*, 57(3):892 – 907, 2011.

[35] Denis Kalkofen, Erick Mendez, and Dieter Schmalstieg. Interactive focus and context visualization for augmented reality. In *6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10, 2007.

[36] Michael Klant and Josef Walch. *Grundkurs Kunst, Band 1-3*. Schroedel Verlag, 2002, 2003, 2005.

[37] Robert Klanten, Sven Ehmann, Nicolas Bourquin, and Thibaud Tissot. *Data Flow 2 - Visualizing Information in Graphic Design*. Gestalten Verlag, 2010.

[38] Bettina Laugwitz, Theo Held, and Martin Schrepp. Construction and evaluation of a user experience questionnaire. *Lecture Notes in Computer Science*, 5298:63–76, 2008.

[39] Sylvain Lefebvre, Samuel Hornus, and Anass Lasram. HA-Buffer: Coherent Hashing for single-pass A-buffer. Research Report RR-8282, Inria Research Centre Nancy - Grand Est, April 2013.

[40] Clayton Lewis and John Rieman. Task-centered user interface design: A practical introduction. A shareware book published by the authors, 1993.

[41] Alexander Lex, Nils Gehlenborg, Hendrik Strobelt, Romain Vuillemot, and Hanspeter Pfister. Upset: Visualization of intersecting sets. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1983–1992, 2014.

[42] Anan Li, Hui Gong, Bin Zhang, Qingdi Wang, Cheng Yan, Jingpeng Wu, Qian Liu, Shaoqun Zeng, and Qingming Luo. Micro-optical sectioning tomography to obtain a high-resolution atlas of the mouse brain. *Science*, 330(6009):1404–1408, 2010.

[43] Kaiming Li, Lei Guo, Carlos Faraco, Dajiang Zhu, Hanbo Chen, Yixuan Yuan, Jinglei Lv, Fan Deng, Xi Jiang, Tuo Zhang, Xintao Hu, Degang Zhang, Lloyd Miller,

and Tianming Liu. Visual analytics of brain networks. *NeuroImage*, 61(1):82–97, 03 2012.

[44] Chih-Yung Lin, Chao-Chun Chuang, Tzu-En Hua, Chun-Chao Chen, Barry J. Dickson, Ralph J. Greenspan, and Ann-Shyn Chiang. A comprehensive wiring diagram of the protocerebral bridge for visual information processing in the drosophila brain. *Cell Reports*, 3(5):1739–1753, 2013.

[45] Ching-Yao Lin, Kuen-Long Tsai, Sheng-Chuan Wang, Chang-Huain Hsieh, Hsiu-Ming Chang, and Ann-Shyn Chiang. The neuron navigator: Exploring the information pathway through the neural maze. *IEEE Pacific Visualization Symposium (PacificVis)*, pages 35–42, 2011.

[46] Eric B. Lum and Kwan-Liu Ma. Hardware-accelerated parallel non-photorealistic volume rendering. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, page 67ff., New York, NY, USA, 2002. ACM.

[47] Trudy F. C. Mackay and Robert R. H. Anholt. Of flies and man: Drosophila as a model for human complex traits. *Annual Review of Genomics and Human Genetics*, 7(1):339–367, 2006.

[48] Richard S. Mann. The Michael Jackson fly. *Science*, 344(6179):48–49, 2014.

[49] Nestor Milyaev, David Osumi-Sutherland, Simon Reeve, Nicholas Burton, Richard A. Baldock, and J. Douglas Armstrong. The virtual fly brain browser and query interface. *Bioinformatics*, 28(3):411–415, 2012.

[50] Judith Moosburner, Jeanne Peter, and Katja Bühler. Visualization of a neuronal atlas. Master's thesis, Zurich University of the Arts, Zurich, Switzerland, 2011.

[51] Zoltán Nagy, Jens Schneider, and Rüdiger Westermann. Interactive volume illustration. In *Vision, Modeling and Visualization 2002*, pages 497–504, 2002.

[52] Christian Nowke, Maximilian Schmidt, Sacha J. van Albada, Jochen M. Eppler, Rembrandt Bakker, Markus Diesrnann, Bernd Hentschel, and Torsten Kuhlen. Visnest, interactive analysis of neural activity data. In *IEEE Symposium on Biological Data Visualization (BioVis)*, pages 65–72. IEEE, 2013.

[53] Shawn R. Olsen and Rachel I. Wilson. Cracking neural circuits in a tiny brain: new approaches for understanding the neural circuitry of drosophila. *Trends in Neurosciences*, 31(10):512–520, 2008.

[54] Alan Peters, Sanford L. Palay, and Henry de F. Webster. The fine structure of the nervous system: Neurons and their supporting cells. *Oxford University Press*, 1991.

[55] Santiago Ramón y Cajal. *Textura del sistema nervioso del hombre y de los vertebrados: estudios sobre el plan estructural y composicion histologica de los centros nerviosos adicionados de consideraciones fisiologicas fundadas en los nuevos descumbrimentos*. Librería de Nicolás Moya, 1899-1904.

[56] Santiago Ramón y Cajal. *Recuerdos de mi vida. Edición de Juan Fernández Santarén*. Editorial Crítica, 2014.

[57] Felix Ritter, Christian Hansen, Volker Dicken, Olaf Konrad, Bernhard Preim, and Heinz-Otto Peitgen. Real-time illustration of vascular structures. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):877–884, 2006.

[58] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-D shapes. *ACM SIGGRAPH Computer Graphics*, 24(4):197–206, 1990.

[59] Casey M Schneider-Mizell, Stephan Gerhard, Mark Longair, Tom Kazimiers, Feng Li, Maarten F Zwart, Andrew Champion, Frank M Midgley, Richard D Fetter, Stephan Saalfeld, and Albert Cardona. Quantitative neuroanatomy for connectomics in drosophila. *eLife*, 5:e12059, 2016.

[60] Rajyashree Sen, Ming Wu, Kristin Branson, Alice Robie, Gerald M. Rubin, and Barry J. Dickson. Moonwalker descending neurons mediate visually evoked retreat in drosophila. *Current Biology*, 27(5):766–771, 2017.

[61] H. Sebastian Seung. Neuroscience: Towards functional connectomics. *Nature*, 471:170–172, 2011.

[62] Gordon M. G. Shepherd, Armen Stepanyants, Ingrid Bureau, Dmitri Chklovskii, and Karel Svoboda. Geometric and functional organization of cortical circuits. *Nature neuroscience*, 8(6):782–790, 2005.

[63] Takashi Shimada, Kentaro Kato, Azusa Kamikouchi, and Kei Ito. Analysis of the distribution of the brain cells of the fruit fly by an automatic cell counting algorithm. *Physica A: Statistical Mechanics and its Applications*, 350(1):144 – 149, 2005.

[64] Kazunori Shinomiya, Keiji Matsuda, Takao Oishi, Hideo Otsuna, and Kei Ito. Flybrain neuron database: a comprehensive database system of the drosophila brain neurons. *Journal of Comparative Neurology*, 519(5):807–833, 2011.

[65] Johannes Sorger, Katja Bühler, Florian Schulze, Tianxiao Liu, and Barry J. Dickson. neuroMap - interactive graph-visualization of the fruit fly's neural circuit. In *IEEE Symposium on Biological Data Visualization (BioVis)*, pages 73–80, 2013.

[66] Robert Spence. *Information Visualization: Design for Interaction*. Prentice Hall, 2007.

[67] Cornelis Jan Stam and Elisabeth C. W. van Straaten. The organization of physiological brain networks. *Clinical Neurophysiology*, 123(6):1067 – 1087, 2012.

[68] Nicolas Swoboda, Judith Moosburner, Stefan Bruckner, Jai Y. Yu, Barry J. Dickson, and Katja Bühler. Visual and quantitative analysis of higher order arborization overlaps for neural circuit research. In *Proceedings of VCBM 2014*, pages 107–116, 2014. VCBM 2014 Best Paper Honorable Mention.

[69] Nicolas Swoboda, Judith Moosburner, Stefan Bruckner, Jai Y. Yu, Barry J. Dickson, and Katja Bühler. Visualization and quantification for interactive analysis of neural connectivity in drosophila. In *Computer Graphics Forum*, volume 36, pages 160–171, 2017.

[70] Christian Tietjen, Tobias Isenberg, and Bernhard Preim. Combining silhouettes, surface, and volume rendering for surgery education and planning. In *Proceedings of the Seventh Joint Eurographics / IEEE VGTC Conference on Visualization*, EUROVIS'05, pages 303–310, Aire-la-Ville, Switzerland, 2005. Eurographics Association.

[71] Susan Tweedie, Michael Ashburner, Kathleen Falls, Paul Leyland, Peter McQuilton, Steven Marygold, Gillian Millburn, David Osumi-Sutherland, Andrew Schroeder, Ruth Seal, Haiyan Zhang, and The Fly-Base Consortium. Flybase: enhancing drosophila gene ontology annotations. *Nucleic Acids Research*, 37(suppl_1):D555–D559, 10 2008.

[72] Albert M. Uttley. The probability of neural connexions. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, pages 229–240, 1955.

[73] Andreas A. Vasilakis and Ioannis Fudos. K+-buffer: Fragment synchronized k-buffer. In *Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '14, pages 143–150, New York, NY, USA, 2014. ACM.

[74] Ivan Viola, Armin Kanitsar, and Meister Eduard Gröller. Importance-driven volume rendering. In *Proceedings of the conference on Visualization*, pages 139–146. IEEE Computer Society, 2004.

[75] Anne C. von Philipsborn, Tianxiao Liu, Jai Y. Yu, Christopher Masser, Salil S. Bidaye, and Barry J. Dickson. Neuronal control of drosophila courtship song. *Neuron*, 69:509–522, 2011.

[76] Jai Y. Yu, Makoto I. Kanai, Ebru Demir, Gregory S. X. E. Jefferis, and Barry J. Dickson. Cellular organization of the neural circuit that drives drosophila courtship behavior. *Current Biology*, 20(18):1602–1614, 2010.

[77] Rafael Yuste. Circuit neuroscience: The road ahead. *Frontiers in Neuroscience*, 2(1):6–9, 2008.

[78] Cha Zhang and Tsuhan Chen. Efficient feature extraction for 2D/3D objects in mesh representation. In *Proceedings of IEEE International Conference on Image Processing*, volume 3, pages 935–938, 2001.