

# neuroMap - Interactive Graph-Visualization of the Fruit Fly's Neural Circuit

DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Visual Computing**

eingereicht von

**Johannes Sorger**

Matrikelnummer 0225843

an der  
Fakultät für Informatik der Technischen Universität Wien

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

Wien, 30.01.2013

\_\_\_\_\_  
(Unterschrift Verfasser)

\_\_\_\_\_  
(Unterschrift Betreuung)



# neuroMap - Interactive Graph-Visualization of the Fruit Fly's Neural Circuit

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

**Master of Science**

in

**Visual Computing**

by

**Johannes Sorger**

Registration Number 0225843

to the Faculty of Informatics  
at the Vienna University of Technology

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

Vienna, 30.01.2013

---

(Signature of Author)

---

(Signature of Advisor)





# Erklärung zur Verfassung der Arbeit

Johannes Sorger  
Wasagasse 31/22, 1090 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

I would like to take the time and thank everyone who accompanied me during my studies and supported me in carrying out the work for this thesis.

I thank Meister Eduard Gröller for his supervision on behalf of the university, his helpful advice during the course of my thesis, and for reviewing my work.

My thanks also go to Katja Bühler for offering me this interesting topic and therefore giving me the great opportunity to work at VRVis, and also for her supervision and her insight during the work on my thesis. I thank Florian Schulze and Florian Deringer for always being available for technical questions, and also the former for proofreading my thesis.

I give my thanks also to Barry Dickson's team at the Institute of Molecular Pathology, especially Tianxiao Liu for his essential input during design, implementation and evaluation phases.

I thank the VRVis, its team, and my colleagues for providing a great place to work as well as a friendly, productive environment.

Finally, I would like to thank my family, who enabled me to accomplish my studies by giving me all the support that I needed.



# Abstract

Neuroscientists study the function of neural circuits in the brain of the common fruit fly *Drosophila Melanogaster* to discover how complex behavior is generated. Through a combination of molecular-genetic techniques and confocal microscopy the scientists are able to highlight single neurons and produce three-dimensional images of the fly's brain. Neurons are segmented, annotated, and compiled into a digital atlas. Brain atlases offer tools for exploring and analyzing their underlying data. To establish models of neural information processing, knowledge about possible connections between individual neurons is necessary. Connections can occur when arborizations (the terminal branchings of nerve fibers) of two neurons are overlapping. However, analyzing overlapping objects using traditional volumetric visualization is difficult since the examined objects occlude each other. A more abstract form of representation is therefore required.

The work in this thesis was motivated by a manually constructed two-dimensional circuit diagram of potential neuronal connections that represents a novel way of visualizing neural connectivity data. Through abstracting the complex volumetric data, the diagram offers an intuitive and clear overview of potential connectivity.

In collaboration with a group of neuroscientists *neuroMap* was designed and implemented in an attempt to deliver the visual features and encoded information of this circuit diagram in an automatically generated interactive graph, with the goal of facilitating hypothesis formation and exploration of neural connectivity.

In this thesis the visual and interaction design decisions that went into *neuroMap* are presented, as well as the result of evaluative discussions that shows that the integration of this novel type of visualization into the existing datamining infrastructure of our clients is indeed beneficial to their research.



# Kurzfassung

Neurowissenschaftler erforschen die Funktionen neuronaler Netzwerke im Gehirn der Fruchtfliege *Drosophila Melanogaster*, um herauszufinden, wie komplexes Verhalten erzeugt wird. Eine Kombination aus molekulargenetischen Verfahren und Konfokalmikroskopie erlaubt es den Wissenschaftlern, einzelne Neuronen in dreidimensionalen Bildern des Fruchtfliegengehirns sichtbar zu machen. Dabei werden die Neuronen segmentiert, annotiert, und in einem digitalen Atlas zusammengetragen. Solche sogenannten Gehirnatlanten bieten in der Regel Werkzeuge, um die darin gesammelten Daten zu explorieren und zu analysieren. Um jedoch Modelle über neuronale Informationsverarbeitung aufstellen zu können, ist es notwendig, über die potentiellen Verbindungen zwischen einzelnen Neuronen Bescheid zu wissen. Solche Verbindungen können auftreten, wenn sich die Arborisierungen (terminale Verzweigungen von Nervenfasern) zweier Neuronen überlappen. Diese Überlappungen anhand von volumetrischen Visualisierungsmethoden zu analysieren ist jedoch schwierig, da sich die betrachteten Objekte durch die Überlappung gegenseitig verdecken. Eine abstraktere Darstellungsform der zu untersuchenden Daten wäre daher vorteilhaft.

Das Thema dieser Diplomarbeit wurde durch ein von Hand konstruiertes zweidimensionales Schaltkreisdiagramm potentieller neuronaler Verbindungen inspiriert. Durch die Abstraktion der komplexen volumetrischen Daten, bietet diese neue Darstellungform eine intuitive und klare Übersicht über die potentiellen Verbindungen der betrachteten Neuronen.

In Zusammenarbeit mit einer Gruppe von Neurowissenschaftlern wurde im Rahmen dieser Diplomarbeit *neuroMap* entworfen und implementiert, mit dem Ziel, visuelle Eigenschaften und kodierte Information des Schaltkreisdiagramms in einem automatisch generierten, interaktiven Graph zu reproduzieren, und dadurch unseren Kollaborateuren das Bilden von Hypothesen über und die Erforschung von neuronaler Konnektivität zu erleichtern.

In dieser Diplomarbeit werden neben den Entscheidungen, die in das Visualisierungs- und Interaktionsdesign von *neuroMap* eingeflossen sind, auch die Ergebnisse der Evaluierung präsentiert, die zeigen, dass die Integration dieses neuartigen Visualisierungsansatzes in die bestehende Datamining-Infrastruktur unserer Kollaborateure tatsächlich hilfreich für deren Forschung ist.

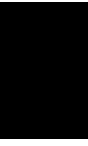




# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Problem Statement . . . . .	1
1.2	Requirements . . . . .	3
1.3	Thesis Overview . . . . .	4
<b>2</b>	<b>Background</b>	<b>5</b>
2.1	Circuit Neuroscience . . . . .	5
2.2	Drosophila Melanogaster . . . . .	6
2.3	The Drosophila Nervous System . . . . .	9
2.4	Data Acquisition . . . . .	9
2.5	Existing Infrastructure . . . . .	11
<b>3</b>	<b>Related Work</b>	<b>15</b>
3.1	Visualization of Neural Networks . . . . .	15
3.2	Biomedical Network Visualization . . . . .	22
3.3	Open Challenges in Biomedical Network Visualization . . . . .	25
3.4	Parallels to Circuit Design . . . . .	28
3.5	Conclusion . . . . .	28
<b>4</b>	<b>Methods</b>	<b>31</b>
4.1	Graph Drawing . . . . .	31
4.2	Abstraction and Visual Encoding . . . . .	44
<b>5</b>	<b>Visual Encoding</b>	<b>49</b>
5.1	Yu's Drawing . . . . .	49
5.2	Abstraction to Graph Elements . . . . .	51
5.3	Visual Encoding . . . . .	54
5.4	Layouts . . . . .	59
<b>6</b>	<b>Interaction</b>	<b>67</b>
6.1	Interface Overview . . . . .	67

6.2	Graph Creation . . . . .	68
6.3	Graph Manipulation . . . . .	69
6.4	Exploration . . . . .	71
<b>7</b>	<b>Implementation</b>	<b>77</b>
7.1	System Overview . . . . .	77
7.2	yFiles AJAX . . . . .	78
7.3	Database . . . . .	79
<b>8</b>	<b>Evaluation and Results</b>	<b>83</b>
8.1	Evaluation Method . . . . .	83
8.2	Discussion of Results . . . . .	84
8.3	Comparison with Yu's Drawing . . . . .	90
8.4	Tackling the Challenges . . . . .	93
8.5	Performance . . . . .	94
8.6	Summary . . . . .	94
<b>9</b>	<b>Conclusion and Future Work</b>	<b>97</b>
9.1	Conclusion . . . . .	97
9.2	Future Work . . . . .	98
	<b>Bibliography</b>	<b>99</b>



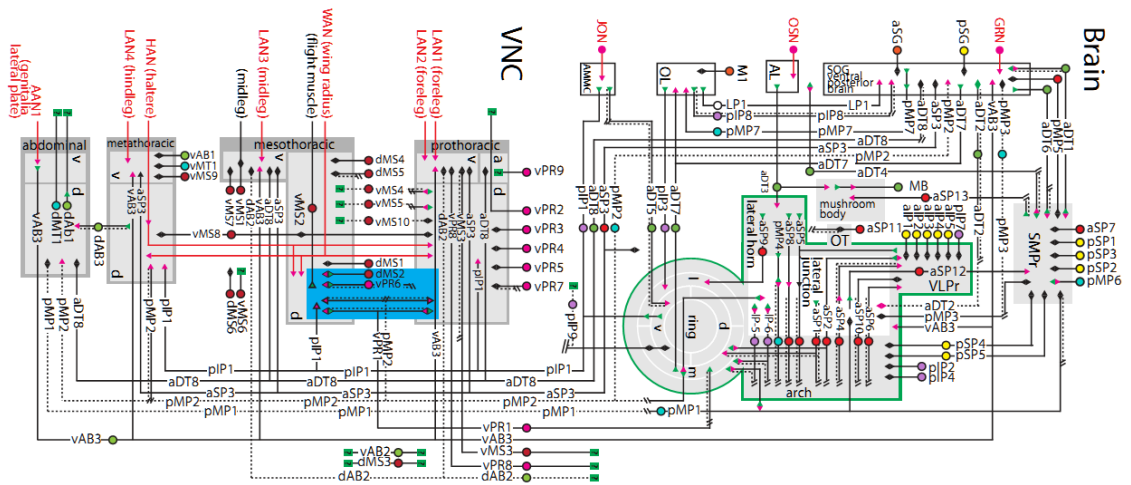
# Introduction

A major goal in circuit neuroscience is to discover how behavior is mediated through information processing in complex neuronal circuits of the brain. Our clients at the Institute of Molecular Pathology (IMP) are studying the brain of the *Drosophila Melanogaster*, also known as the common fruit fly, in order to find out how the function of neural circuits drives the fly's courtship behavior [78]. The fruit fly is a popular model organism in the neuroscience community, because it allows studies of complex processes on a simple example. By applying molecular-genetic techniques the scientists are able to highlight single neurons and produce three-dimensional images of the fly's brain using confocal microscopy. These images together with the corresponding meta-information, like the classification of the visible entity, the test subject's sex, etc. are stored in a relational database for further study and analysis.

## 1.1 Problem Statement

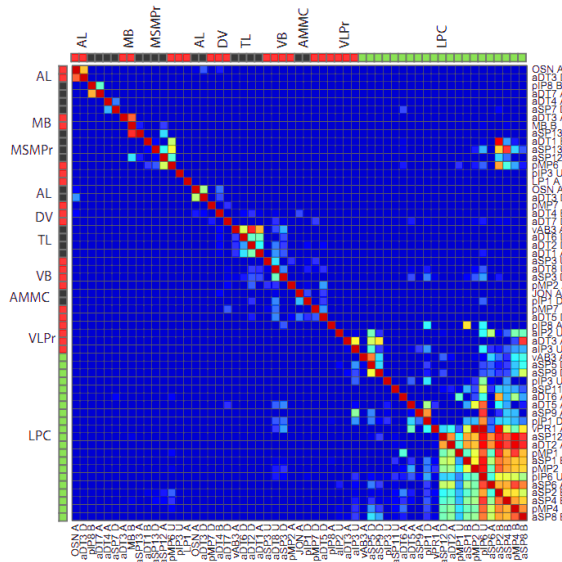
Knowledge about neuron connectivity is essential for the understanding of how information is processed and transmitted within the brain. Thus, one of the tasks of the scientists is the discovery of connections between neurons in the fruit fly's brain. A necessary but not sufficient condition for the existence of a connection is an overlap between arborizations (treelike terminal branching of nerve fibers) of two neurons. Visualization of these potential connections would, on the one hand, support the formation of hypotheses about neuronal connections, and on the other hand, offer visual verification of already confirmed research results.

In the scope of several joint research projects, a complete software infrastructure of data management, data mining, and visualization has been developed by VRVis and IMP. BrainGazer handles the 3D visualization of the collected images and segmented



**Figure 1.1:** The original drawing from Yu's publication [78].

structures, but their exploration becomes cumbersome. While it is possible to judge overlaps of two or three objects in three-dimensional space, it becomes cumbersome when there are more objects involved, since overlapping objects occlude each other. Also, for the analysis of overlaps anatomical accuracy and exact spatial positioning of the visualized entities are not as important as the ability to display large amounts of data in a clearly structured overview. A more abstract form of representation was therefore



**Figure 1.2:** The heatmap associated with the diagram in Figure 1.1 [78].

desired, which is why Jai Y. Yu used a graph representation for depicting these neuronal relationships (Figure 1.1). The graph shows which neurons have arborizations in which brain regions, while a heatmap (Figure 1.2) depicts the amount of overlap between arborizations per brain region (neuropil). Yu’s wiring diagram was created manually in Adobe Illustrator in multiple iterations over a couple of months. The positive response towards Yu’s drawing within the group of researchers and the scientific community motivated the creation of a tool that should replicate the features of the graph and further expand on them - thus neuroMap was born.

## 1.2 Requirements

neuroMap is being developed with the goal to supply the scientists at the IMP with a two-dimensional representation of their accumulated neuronal data in order to support and facilitate their research by fulfilling the following requirements:

- *Easier, more intuitive neuron connectivity hypothesis-formation:* By combining the information of heatmap and wiring diagram into a single automatically generated graph neuroMap visualizes arborizations as nodes and the potential connections between them as edges, thus letting scientists grasp all potential connections of the analyzed data at a single glance.
- *Visual exploration of the accumulated neuronal data:* Having a static graph that depicts user-specified data is helpful, but the real power of a visualization is unlocked once it offers the user a way to interact with it. Features like arborization-overlap filters and context menus for loading related or overlapping objects allow the user to extend the graph in directions of interest and thus interactively explore the neuronal database.
- *Fast generation of neural-circuit-graphics for presentation purposes:* Researchers use circuit diagrams of neural structures like in Figure 1.1 to demonstrate scientific findings in papers or presentations. Creating these diagrams manually is a laborious, time consuming process. neuroMap generates these structures automatically while offering a variety of different layout algorithms to achieve results that are meaningful and visually pleasing. Editing of the graph structure as well as export to various picture and graph formats are supported to allow further usage and manipulation of the graph.

Using two-dimensional graphs to visualize biological networks is not a new idea. A variety of software solutions already exist [29]. Nevertheless, the two fields of graph drawing and network biology are still largely disconnected, as stated by Albrecht et al. [3] who identified seven open problems in biological network visualizations. Problems relevant to our approach are: *the visualization of multiple attributes* (object type,

overlap amount, gender, neuron association), *location constraints* (assignment of nodes to specific brain regions), *visualization of flows and paths* (highlighting of interconnected entities), *exploration of hierarchical networks* (extending and reducing the graph in desired directions). The fact that existing tools tackle some of these problems but not in a combination that is desirable for our approach, as discussed in chapter three of this thesis, and the requirement to integrate the visualization into an existing framework led to the development of our proprietary solution.

The contributions of this thesis lie in the design and evaluation of a system that implements a new approach of visualizing potential neuronal connections and offers a new way of exploring neural overlap data by fulfilling the stated requirements and handling the above mentioned open problems in biological network visualization in a way that is tailored to the application-specific context and data.

## 1.3 Thesis Overview

In the next chapter more detailed background about our clients' research and the existing infrastructure is given. Chapter three discusses related work in the areas of neural and biomedical network visualization. In chapter four the basics of the techniques that went into neuroMap's design and implementation are explained. Chapters five and six describe the visual encoding and interactive functionality that are implemented in neuroMap. Chapter seven gives an overview of the system architecture and relevant implementation details; followed by an evaluation and discussion of results in chapter eight. The conclusion of the thesis and an outlook on future work is given in chapter nine.

# CHAPTER 2

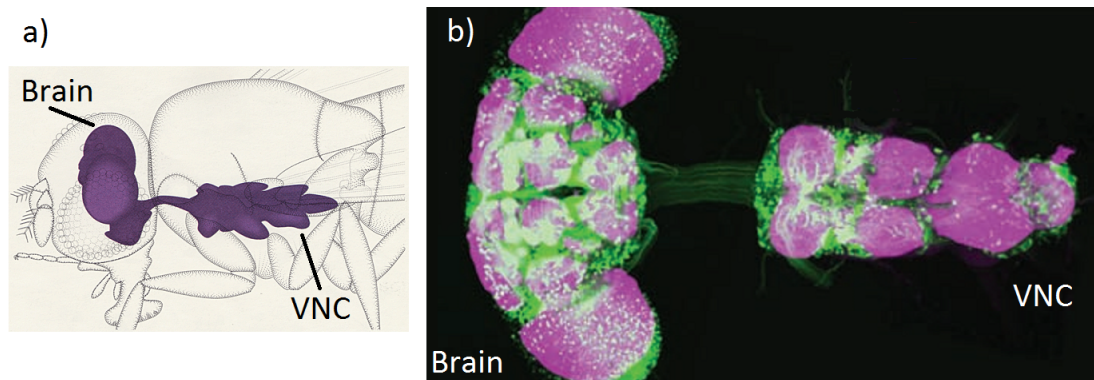
## Background

In this chapter the research area of our clients at the IMP is introduced in more detail, followed by a discussion of *Drosophila*'s characteristics and the data that forms the center of the scientists' research. Finally the software infrastructure in which neuroMap was integrated is presented.

### 2.1 Circuit Neuroscience

Circuit Neuroscience can be regarded as the understanding of the computational function of neural circuits, as linking the function with the circuit micro-structure. Yuste describes the goal of this field as the reverse-engineering of biological circuits [79]. In order to do so, knowledge about their structure and logic is needed, so that their computational algorithms can be understood. To completely solve a neural circuit would require the scientists to (i) describe a behavior whose neural circuit mechanisms they seek to understand, (ii) identify which neurons are involved, (iii) determine what drives activity in each type of neuron and how the related signals are transformed through the circuit, (iv) discover the cellular, synaptic and circuit mechanisms underlying these neural transformations, and finally (v) understand why these neural transformations are useful intermediates in producing this behavior [52]. Science is still at the beginning of this formidable task which is perhaps the biggest neuroscience breakthrough solvable within our lifetimes. What the genome project was for molecular biology, the circuit connectivity project could be for neuroscience [79].

Imaging techniques, in spite of their relative novelty, are already responsible for a significant push ahead of many areas of circuit neuroscience. The complete connectomic reconstruction at the synapse level is currently possible for small brain volumes using electron microscopy techniques, but technically not yet feasible for volumes of



**Figure 2.1:** *Drosophila*'s brain and VNC: a) schematic view in anatomical context [73], b) a confocal staining image depicting a Gal4/UAS neuron in green and stained synapses in magenta [23].

the size of a cortical column. Approaches on the human brain have mainly documented the connectivity between brain regions of living subjects. For this, MR scans are used as reference for further measurements. Then a time series of brain activity in different voxels or regions is derived for establishing functional connectivity (between brain regions, not at single cell resolution) [38].

Deciphering the human brain would be the ultimate goal but since it is the most complex of all existing organisms, researchers use *model organisms* as small scale examples in its place. Model organisms are specific types of plants or animals that can be relatively quickly and cheaply bred in large numbers and serve as scientific study objects without raising (too many) ethical concerns. The genomes of model organisms were therefore among the first to be completely sequenced. Popular examples of such species are the *Escherichia coli* bacterium, the *Caenorhabditis elegans* worm, the zebra fish or the common mouse.

The *C. elegans* was the first species to have its complete neural wiring (connectome) documented at single cell resolution by researchers in the 1970s and 1980s. They imaged extremely thin slices of this organism using serial electron microscopy, virtually reconstructed each neuron by finding all its cross-sections in the images, and found every synaptic connection between neurons [62]. The *C. elegans* is one of the most simple organisms to exhibit neural wiring structures. It has only 302 neurons and is only 1 millimeter in length. The behavioral repertoire of this species is therefore less extensive.

## 2.2 *Drosophila Melanogaster*

Compared to *C. elegans*, the fruit fly is already a much more complex organism. Measuring a length of only 2.5 mm, it possesses a nervous system that consists of a brain

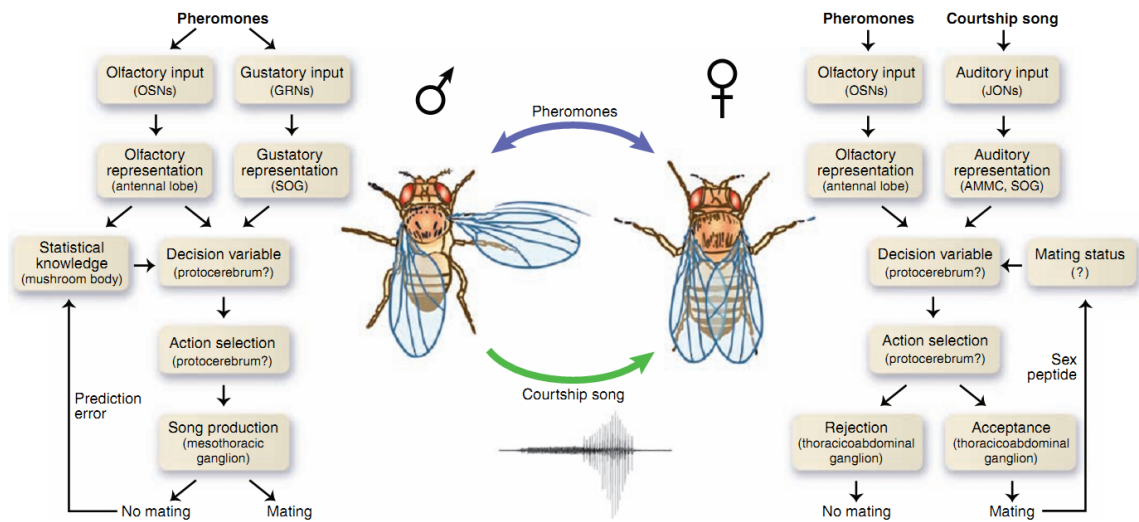


with two halves and a ventral nerve cord (VNC) with a total of approximately 250.000 neurons [64] (Figure 2.1). Compared to the human brain with its billions of neurons, this is still tiny. Nevertheless the *Drosophila* already possesses „the rudiments of consciousness“ and can perceive sound, vision, and smell. *Drosophila* exhibits a rich repertoire of complex traits, some of which have clear human homologs; e.g. circadian rhythm, sleep, drug responses, locomotion, aggressive behavior, and longevity. In addition it possesses high enough intelligence to express learning and memory abilities. All of these traits make this species a capable study object [64].

Deciphering every neural circuit in every species would be impossible and pointless [52]. A detailed comparison of several circuits in different species, however, should help reveal what features of neural circuits are fundamental and which are specializations. Many neuroscientists believe that *Drosophila* is a species worth including in this research program. One reason is the power of the *Drosophila* genetic toolbox, although mouse neurogenetic tools are beginning to rival those of the fly. Still, there are 1000-fold fewer neurons in the brain of the fruit fly which makes it a simpler test subject. Another advantage are stereotyped neurons that can be located (in theory) in every fly. Many *Drosophila* neurons are identifiable in this way. Through these properties the fly represents a useful compromise between tractability and richness [52].

The importance of *Drosophila* as a model organism in genetics has already been documented by Mackay et al. [44]. Between flies and humans there is considerable evolutionary conservation of genes and pathways affecting key biological processes, including human disease genes. More than 60% of all genes that are known to affect human disease have *Drosophila* orthologs. More than 50% of all *Drosophila* protein sequences are similar to those of mammals. Several studies validate the use of the fruit fly as powerful genetic model system for studies of human disorders like alcoholism, sleep disorders, and neurodegenerative diseases, like Alzheimer's disease, Parkinson's disease, and Huntington's disease. In each instance, genomic approaches with flies can be used to describe disease mechanisms at the genetic level. For these types of studies, large numbers of genetically identical individuals need to be reared under controlled environmental conditions. A single fly can lay up to 400 eggs and the development from egg to adult only takes 7 days under optimal conditions. Fast generation of large sample sizes is an essential factor for the construction of replicated „designer“ genotypes.

As mentioned above, the deciphering of neural circuits generally begins with an observable behavior that the neuroscientists seek to understand [52]. Dickson and his team study neural functionality on the model of the fruit fly's courtship behavior [23]. Behavior unfolds as animals select specific actions on the basis of sensory input, internal physiological states, and individual experience. Upon encountering another fly, a male may or may not choose to court. He estimates his chances of success primarily on the basis of pheromone signals and previous courtship experience. The female decides whether to accept or reject the male, depending on her perception of his pheromone and



**Figure 2.2:** *Drosophila* mating decisions. The male (left) decides whether to court based on perceived female pheromones and experience. The female decides whether to mate based on the male pheromones, courtship song and current mating status. The boxes indicate the relevant neurons or brain regions for decision making [23].

acoustic signals, as well as her own readiness to mate. Figure 2.2 shows elements of the *Drosophila* male and female mating decisions. This simple and genetically tractable system provides an excellent model to explore the neurobiology of decision making. The goal is to understand how information processing and storage in neural circuits guides such selection, and thus behavior. An important first step toward this goal is to trace the neural pathways that mediate a complex instinctive behavior, from sensory input through to motor output. Dickson therefore set out to define the neural circuitry that governs male courtship behavior in *Drosophila*. Genetic approaches in model organisms greatly facilitate the identification, characterization, and manipulation of individual circuit elements and can thereby establish causal relationships linking cellular biochemistry, circuit function, and animal behavior. These neural mechanisms are also accessible to both genetic and physiological investigation at the level of single identifiable neurons. The research is still in an early stage, but work on the fly's mating decisions has the potential to reveal fundamental mechanisms of action selection - to teach us how the brain maps sensory input, internal states, and individual experience onto moment-to-moment behavioral choices [23].

## 2.3 The Drosophila Nervous System

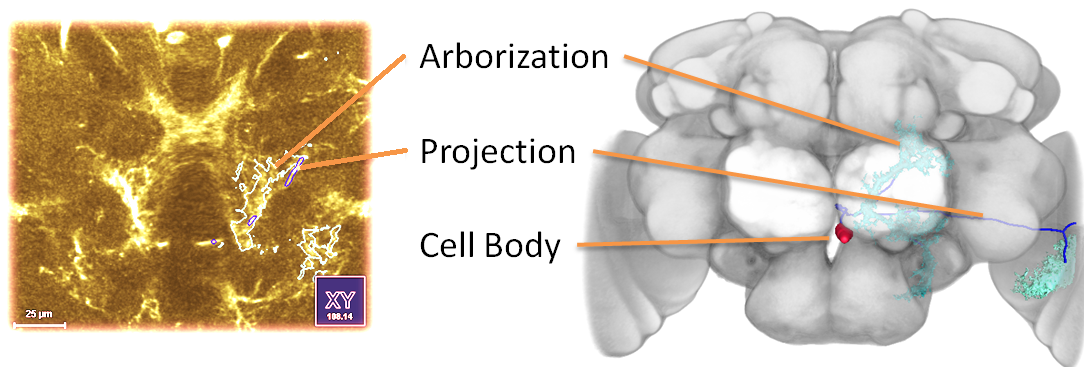
As already stated, the nervous system of *Drosophila* consists of brain and ventral nerve cord. Figure 2.1 b) shows the cervical connective between both regions as fine green fibers. The nervous system is composed of neural cells (or neurons) which in turn can be partitioned into cell body, arborizations and projections. The *cell body* contains the cell's nucleus, the control center of the cell. *Arborizations* are terminal branchings of nerve fibers that form synapses where communication with other neurons occurs. Synapses (connections) between two overlapping arborizations can only exist if one terminal is dendritic and the other is presynaptic. In vertebrates the dendrite is close to the cell body and axon terminals sit at the end of a projection. In common invertebrates like the fruit fly, the cell bodies are situated at the outer regions of the brain; all arborizations are therefore linked to their cell body by a *projection* [11]. The terminal type is therefore not definable by the existence of a projection. *Neuropils* are functional brain regions. Brain and VNC are partitioned into 60 neuropils. Neurons in a specific region are associated with certain tasks, like the optical lobes close to the eyes for example.

### Peters' Rule

The relationship between the connection of two neurons and the overlap of their arborizations can be described by Peters' rule [12]. Basically Peters' rule states that the probability of the existence of a *structural* synapse between two neurons can be estimated based on the size of their arborizations' mutual overlap. The bigger the overlap, the more structural synapses may occur, and the higher the 'strength' of the connection. Although Peter's rule makes no explicit inference about the *functional* strengths of connections, it provides a blueprint of the implied functional circuit if the synaptic strength per unit of axon-dendrite overlap (per potential synapse) is assumed to be constant on average [63]. Peters' rule can therefore be used to calculate connection strengths between different classes of neurons and to generate neuronal circuit diagrams based on anatomy alone [63] [12].

## 2.4 Data Acquisition

The neuroscientists use the Gal4/UAS system [13] to highlight specific neurons in *Drosophila*'s brain. With this approach, a targeted subset of neuronal tissue can be genetically manipulated so that it produces green fluorescent proteins (GFP). Two lines (genetically identical groups) of flies, one carrying the Gal4 protein, the other the UAS target gene, are crossed. In the child line the Gal4 protein expresses (activates) the UAS GFP gene. The resulting fluorescence effect makes the targeted neurons stand out from the remaining tissue. High resolution 3D volume images showing brain tissue in one

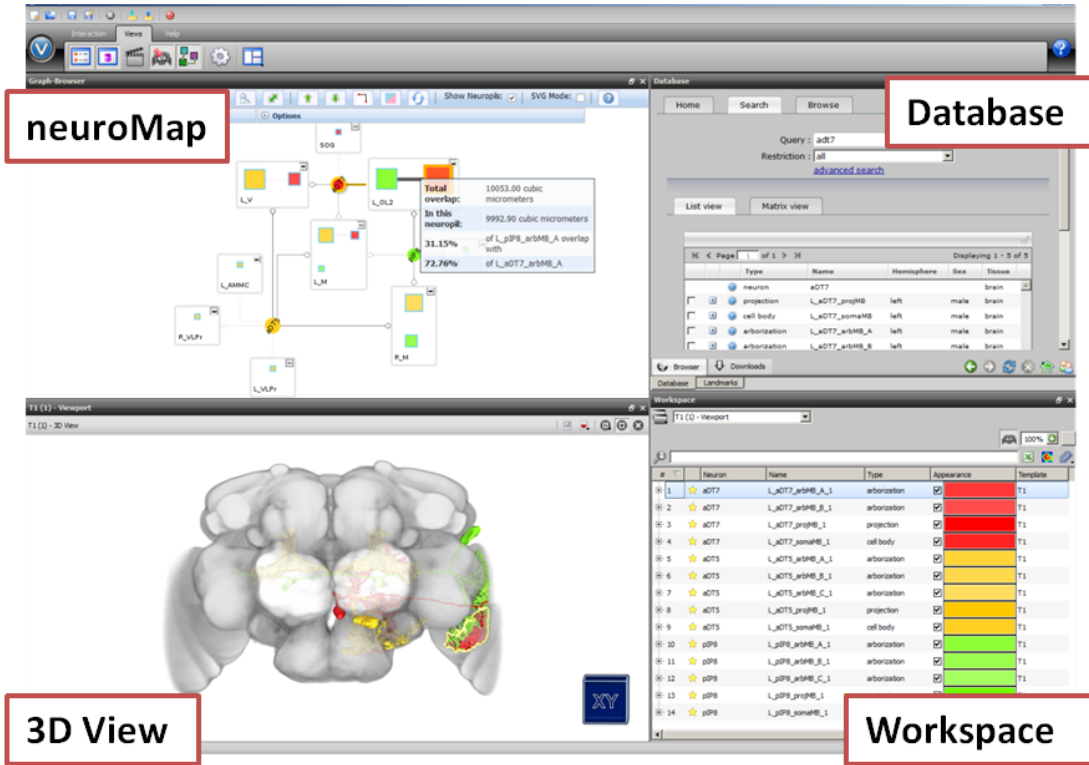


**Figure 2.3:** Example of a segmented neuron shown as outlines on a 2D section of its source image and in a 3D context together with the standard brain.

channel and the highlighted neurons in a second channel are generated with confocal microscopy. For this, the fly's brain and VNC must be carefully dissected and separated from the remaining body. The acquired scans are registered by applying a non-rigid registration method [60] to a standard brain template using the first channel. The template is an average image of a set of representative scans that were registered against a reference scan. The neurons visible on the second channel of the images consist of three elements: a single cell body, a set of arborizations and a neural projection (see Figure 2.3). Cell bodies are recognizable as small spots with high density; projections as tube-like structures that traverse the brain; and arborizations as treelike terminal branchings that can cover large areas. Neurons are classified based on the morphology or shape of these features. Neurons that share similar cell bodies, projection patterns, and arborizations, as well as expression of the same Gal4 drivers are considered to belong to the same type. Neurons that belong to the same type may perform similar functions.

After successful registration, interesting neurons are segmented semi-automatically using Amira [67]. Cell bodies, projections and arborizations are segmented separately and stored as binary masks and geometry. Each object can have multiple instances and is assigned to a single neuron. These relations, references to the object's reference image and origin, binary masks and generated surface geometry are stored in a relational database.

An indicator for communication between two neurons is the existence of an overlap between one or more arborizations of these neurons, whereas a connection can only occur between synapses of opposite polarity (dendritic and presynaptic). The overlap of an arborization or an arborization-arborization overlap with a specific synaptic neuropil gives information about the potential function of the neuron(s). The absolute amount and percentage of overlaps for arborization-arborization overlaps, arborization-neuropil

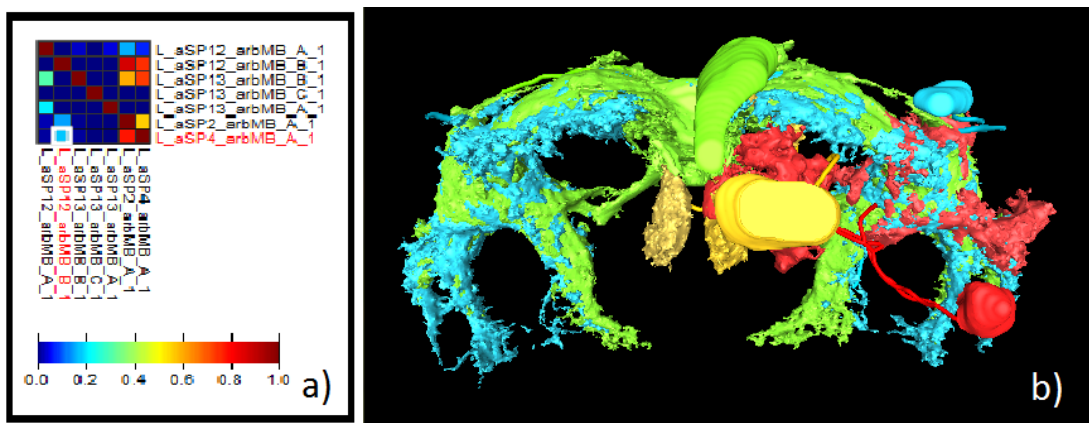


**Figure 2.4:** Screenshot of neuroMap integrated into BrainGazer. neuroMap and the BrainGazer Views share the same Workspace. Thus highlighting, editing, deletion and adding of objects is instantly propagated.

overlaps and arborization-arborization-neuropil overlaps are precomputed and stored directly in our clients' database. Since these values are precomputed they can be accessed efficiently for visualization and exploration purposes. Although light microscopy can provide much higher throughput than other methods, synapses are still smaller than the diffraction limit and cannot readily be assigned to particular presynaptic and postsynaptic cells without specialized labeling methods [76]. Synaptic information is therefore not included in the supplied dataset.

## 2.5 Existing Infrastructure

*BrainGazer*, into which neuroMap was added as an additional view in the course of this thesis, already offers means for exploring the available neuronal data. Figure 2.4 shows a screenshot of one possible configuration of *BrainGazer*'s user interface with neuroMap.



**Figure 2.5:** Details of BrainGazer views: a) the 3D viewport rendering seven arborizations together with their cell bodies and projections of four different neurons, b) a heatmap generated from the same four neurons, showing overlaps between the seven arborizations.

A textual query interface allows semantic queries on the database. Results can be further explored on the items' detailed information pages with related database entries, preview images and links to source image, surface file and segmentation mask of the respective object are accessible. These links already offer a textual way to explore the database. Additionally each info page enables the user to search for and retrieve a list of similar objects in the database. The advanced search also offers a parallel coordinate view that can be used to find images that have similar staining values in different neuropils. Also, a heatmap of selected arborization objects can be generated, visualizing the overlaps between them (Figure 2.5 a)).

Query results can be added to BrainGazer's workspace from where they can be added to or removed from the 3D view (Figure 2.5 b)). In the 3D view visual queries can be issued on the displayed objects by drawing a path on them. The resulting query then tries to find objects that are close to the specified 3D path. This allows the user to search the whole set of available data based on spatial relationships. BrainGazer therefore already offers information about object overlaps on object info pages and heatmap, as well as in the visual query results. Neuropil pages list the contained arborizations of the respective neuropil; arborization pages point to the neuropils as well as to the other arborizations that they overlap. Similarly the visual queries return all overlapping objects sorted by shortest distance/overlap. Overlap between workspace items is encoded by color in the heatmap. However, when exploring potential connectivity between multiple arborizations, each existing view has its shortcomings. The info pages just give textual information about the overlap of a single object; the 3D view offers a spatial context to the displayed overlaps but can get visually cluttered quickly when many overlapping

objects are rendered. In the heatmap, big overlaps are instantly recognizable through color coding, but all spatial context is lost and overlaps are not partitioned between brain regions (neuropils).

The inclusion of neuroMap as an additional overlap-centric way to visualize and explore potential neuronal connectivity, that offers visual abstraction as well as spatial context and direct interaction, is therefore reasonable.





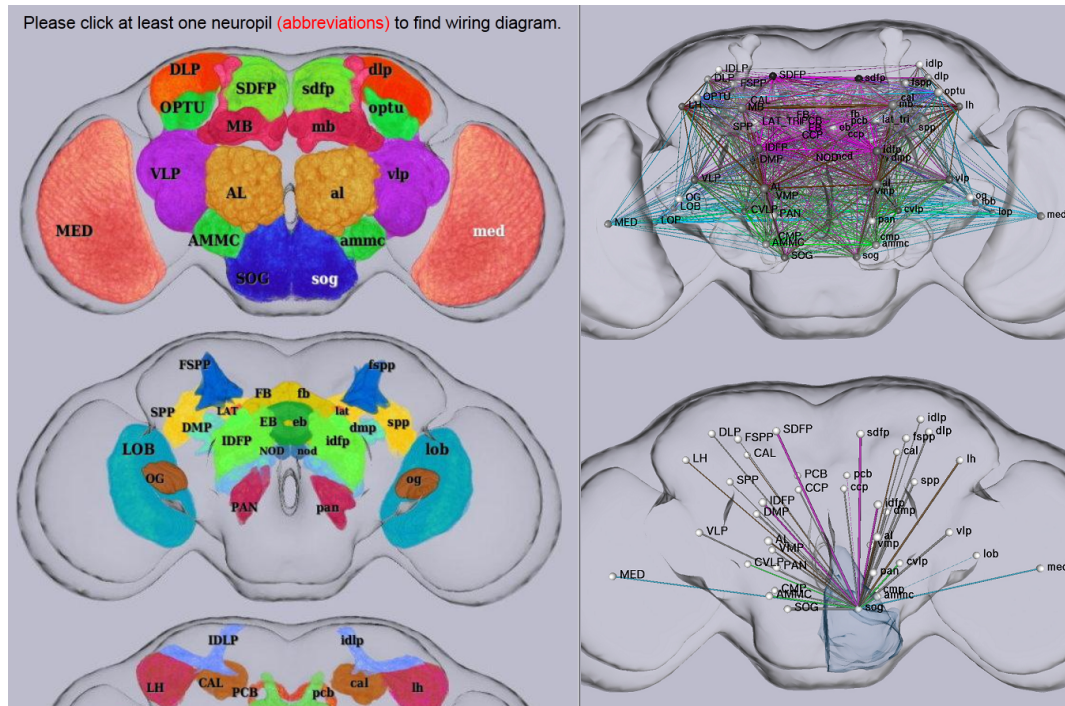
## Related Work

This chapter discusses the state-of-the-art methods that are relevant to this thesis, namely the visualization of neural networks but also approaches in biomedical network visualization. The chapter concludes with a discussion of open challenges in this active research area and a justification for the implementation of our own approach in the context of existing approaches.

### 3.1 Visualization of Neural Networks

Even though a wide range of brain atlases for the exploration of collected neuroscientific data on various species are available [4] [73] [46], the depiction and exploration of neural network structures especially at single cell resolution is not very wide spread yet. Circuit neuroscience is a relatively young field and the means for acquiring connection data at high resolution are still very limited.

The FlyBrain Project was one of the first neuronal online atlases [4]. It features Java applets for 3D and slice viewing of the brain. FlyBase is a database and web portal for genetic and genomic information on the fruit fly [73]. Neuronal connectivity is not covered. The Virtual Fly Brain web interface allows users to explore the morphological structure of the *Drosophila* brain by browsing 3D images of a brain with subregions displayed as colored overlays [46]. An integrated query mechanism allows searches of underlying anatomy, cells, expression and other data from community databases. Neu-rARt II [15] provides a 2D visualization interface to different neuroanatomical atlases. It allows the exploration, comparison and manipulation of images from different sources in a single framework. The CoCoMac-3D Viewer [9] implements a visual interface to two databases containing morphology and connectivity data of the macaque brain for analysis and quantification of connectivity data. XANAT allows graphical searches

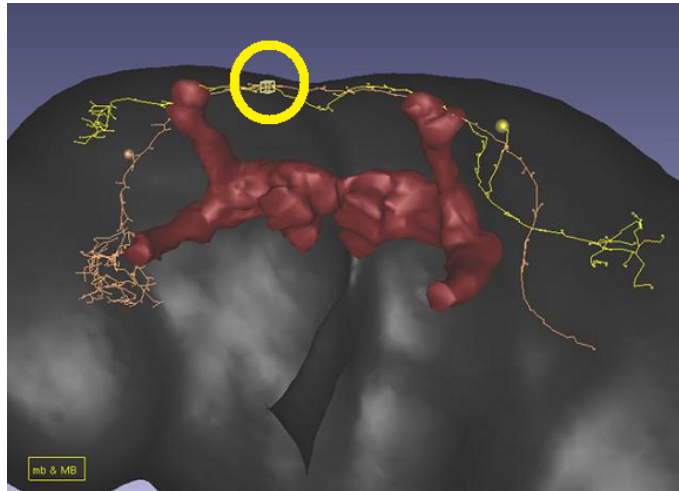


**Figure 3.1:** FlyCircuit’s wiring diagram depicting connected brain regions [16].

within neuroanatomical atlases to study, analyze, and store neuroanatomical connections [57]. Users perform searches by graphically defining regions of interest to display the associated connectivity information.

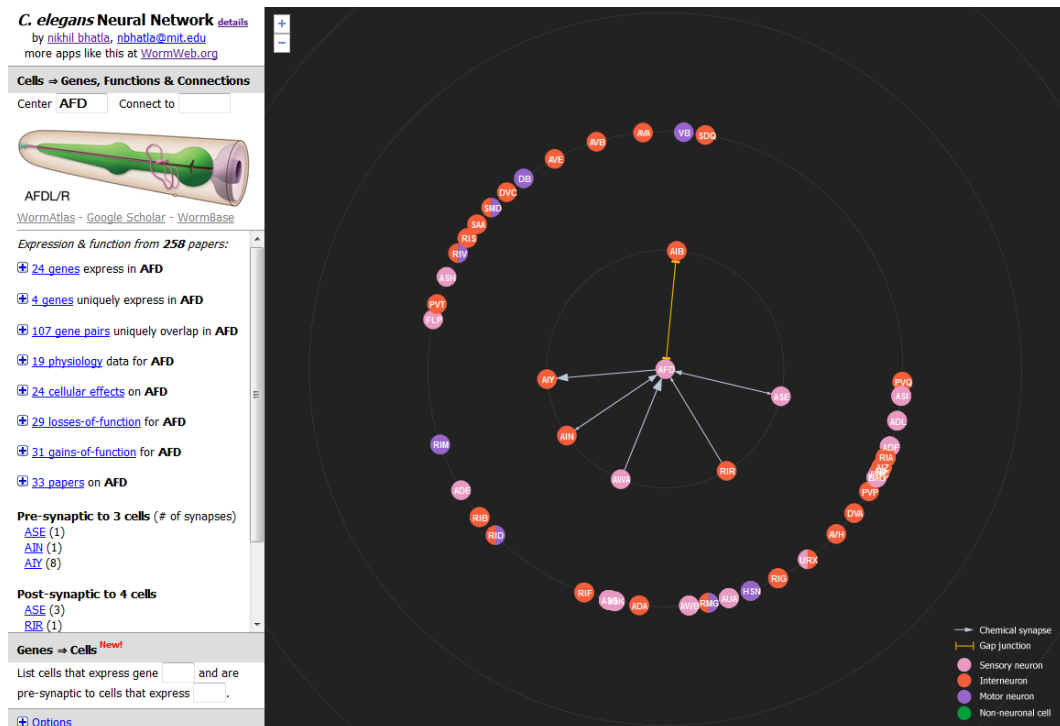
*FlyCircuit* is a web service that grants access to a public database of the fruit fly’s neurons. It has been published by a group of researchers that work on a similar goal as our clients [16]. The page offers a static wiring diagram that displays which brain regions are connected by drawing straight lines between the centers of the respective regions. The color of a line indicates the functional module. The user is offered a view of the brain in the form of three slices that picture the different neuropils highlighted by color on the left side and the complete wiring diagram on the right (Figure 3.1). Upon clicking on one neuropil, a wiring diagram that shows only the connections of the selected neuropil is added to the view below the complete diagram. A tract finding feature offers the same illustration of neuropils with the difference that here two neuropils have to be selected in order to receive a prerendered view of the neuronal tracts that connect these regions.

*Neuron Navigator* is a visual query interface to FlyCircuit’s database, focused on observing and discovering potential connections within the fruit fly’s neural network [43]. Neuron Navigator offers interactive exploration of neural connectivity in 3D as it allows users to visually query for connected entities in specified regions. Neurons are not



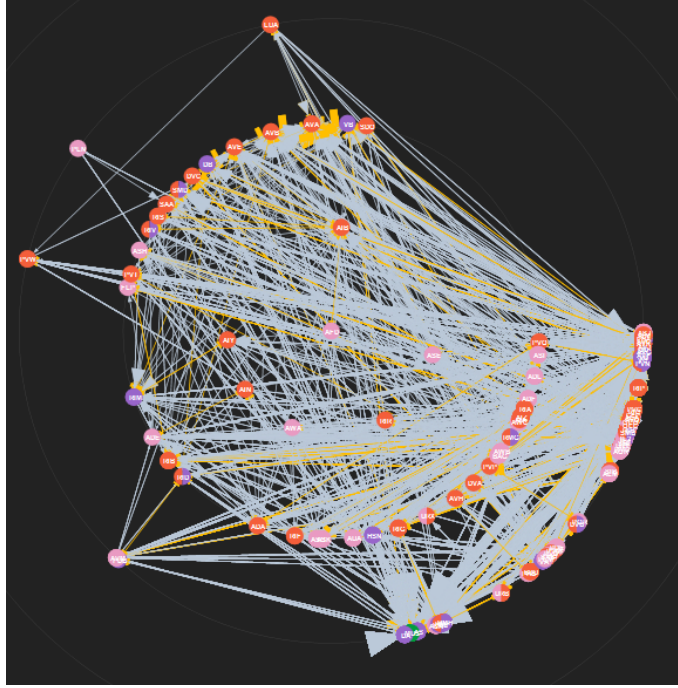
**Figure 3.2:** Neuron Navigator displaying neurons that pass through a user defined region (yellow circle) [43].

rendered as volumes but as tracing lines in a color according to their neuron-transmitter category (Figure 3.2). This avoids occlusion through overlapping volumes but at the same time confers no volumetric information. Bhatla created a web application that displays the *neural network of the C. elegans as an interactive graph* [10] in an attempt to offer scientists easy exploration and navigation of the worm's neuronal data (Figure 3.3). The graph displays neurons as nodes and the connections between them as edges. Nodes are split into sensory-, inter- and motor-neurons as well as non neuronal cells. Edges are visually represented as chemical synapses (gray arrows) or gap junctions (yellow brackets). The size of the arrows is proportional to the number of chemical synapses made between the neurons. Similarly, the size of the orange bar is proportional to the number of gap junctions made between the two neurons. The graph is displayed in a circular layout with the selected neuron in its center. The standard setting shows only the connections to direct neighbors of the selected cell. All indirect neighbors are shown on the outer layers of the circular graph; the layer level depending on each node's distance to the center node. The farther zoomed out the camera is the more layers and nodes are displayed. Optionally the representation of nodes can be switched between individual neurons and neuron groups; further all additional connections in the visible network can be rendered. Depending on the number of nodes and connections that are shown, the view can become very cluttered (Figure 3.4) and the performance decreases significantly. When the user hovers the pointer over a node, the cell name and type are displayed as a tooltip. A frame that gives detailed information about the selected neuron is situated on the left side of the webpage. Post- and presynaptic neighbors are listed as well as genes that express the respective neuron and additional information like



**Figure 3.3:** The interactive interface for the neural network of the *C. elegans* [10].

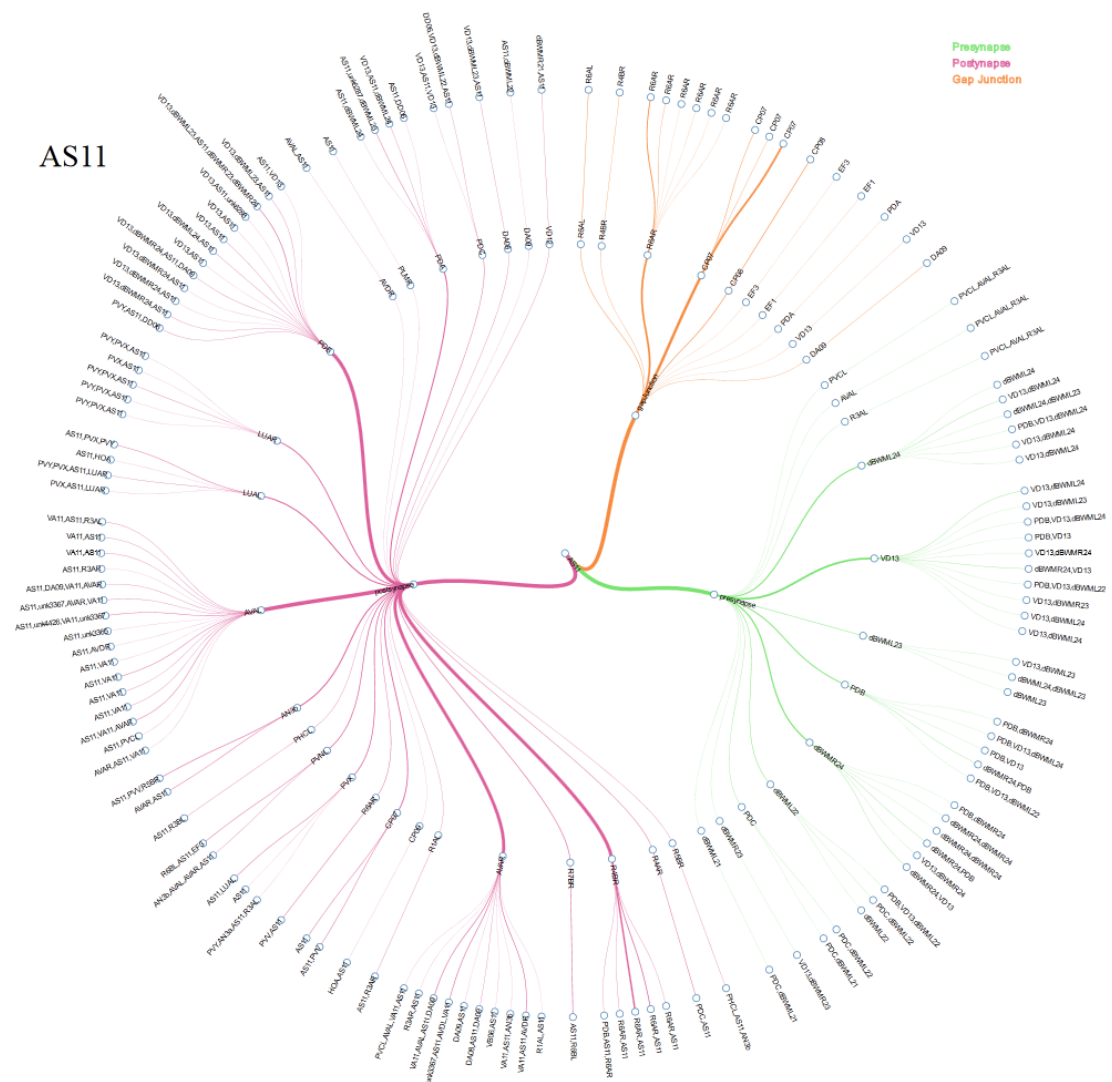
scientific papers and links to external infopages related to the neuron. A query interface lets the user specify a second neuron to display the shortest path to the center neuron. The graph shows only anatomical connections; functional information is not included. The WormWiring Project [20] is a webpage that tries to compliment the anatomical descriptions of WormAtlas and WormImage [32] with interactive tools for exploring synaptic connections. The *Partner Tree* tool (Figure 3.5) displays all partnerships for a given neuron, similarly to Bhatla’s web application. The nodes are again distributed to radial hierarchy layers around a selected neuron in the center. Here the layers have a different meaning though. The first level divides partnerships into synapse classes (presynaptic, post synaptic, and gap junction). Each class is distinguishable by its color coded edge type. The second level shows the neuron partners and the third level holds the individual synapses. The width of an edge corresponds to edge weight, i.e., the number of synapses for a given partnership. Upon mouse-over on a node a tooltip displays the edge weight as well as the synapse type and the name of the neuron partners. A new graph can be generated by typing the name of the desired neuron into a query interface on the top of the webpage. Irímia et al. developed a circular representation of human cortical networks for the exploration of the central nervous system architecture [36]. This *connectogram* (Figure 3.6), as they dubbed it, should help to classify



**Figure 3.4:** A cluttered display when rendering all connections in Bhatla’s tool [10].

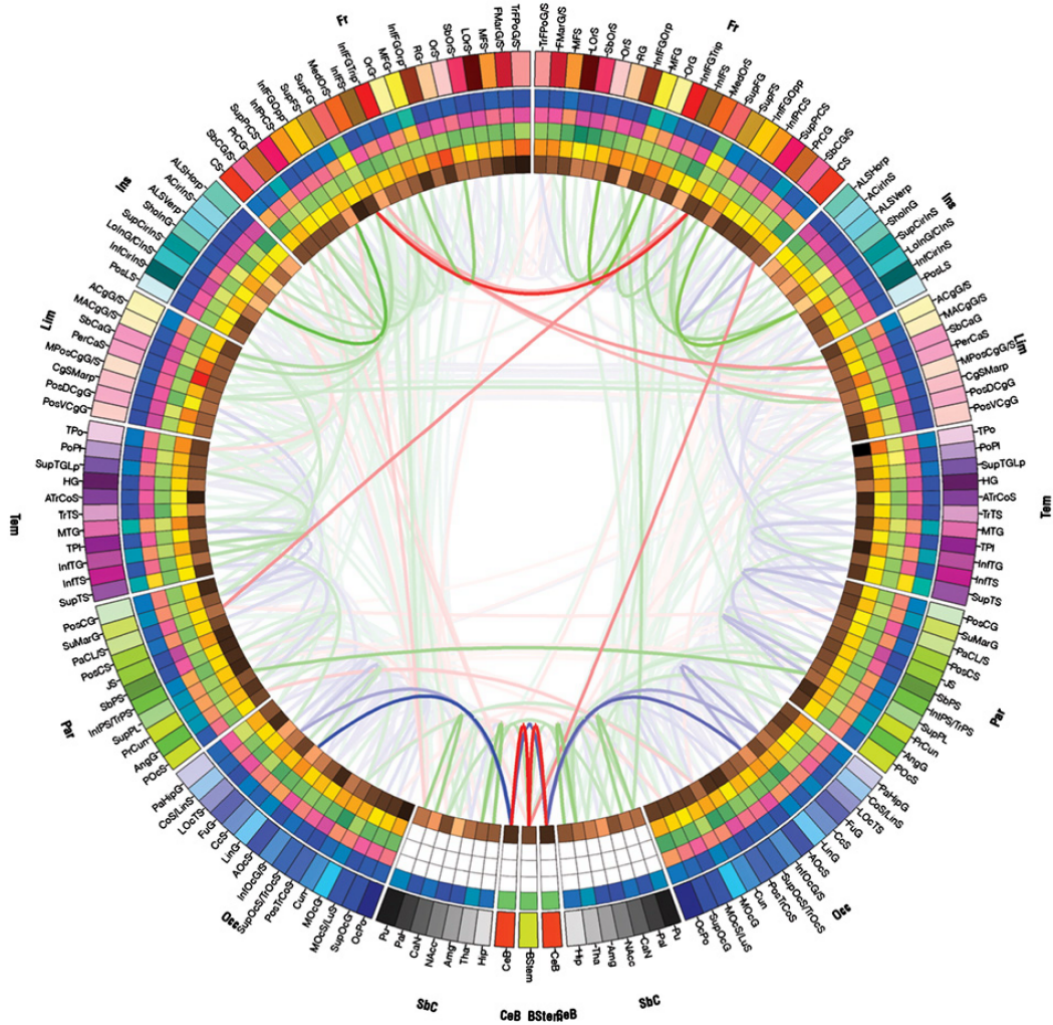
neuron connectivity relationships. The visualization is generated with Circos [41] which was originally created for visual genome comparison and analysis. The outermost ring of the connectogram shows the various brain regions as a circular array of 165 radially aligned elements arranged by lobe and further ordered anterior-to-posterior. The five inner rings are circular heat maps, each encoding a structural measure associated with the corresponding parcellation (Figure 3.7). The links represent the computed degrees of connectivity between segmented brain regions. Links shaded in blue represent tractography pathways in the lower third of the distribution, green lines in the middle third, and red lines in the top third. This allows visual comparison and analysis of the degrees of similarity and variability between the connectivity profiles of different subjects.

Jianu et al. created a tool for visualizing tractography datasets as two-dimensional paths that, in contrast to 2D point representations, preserve anatomically meaningful coordinates while possessing the advantages of abstract representations, i.e., visual clarity and easier tract-of-interest selection [37]. These *two-dimensional neural maps* should help explore and analyze connectivity in the human brain. The design of the visualization was inspired by illustrations in medical textbooks (Figure 3.8). The 2D view offers three perspectives (axial, sagittal, coronal) and is linked with a 3D view for reference. Individual tracts can be selected by brushing in each view. The number of visible tracts is dependent on the zoom level. Selected tracts can be used to retrieve additional



**Figure 3.5:** The Partner Tree displaying neuron partnerships. One of two analytic tools on the WormWiring webpage [20].

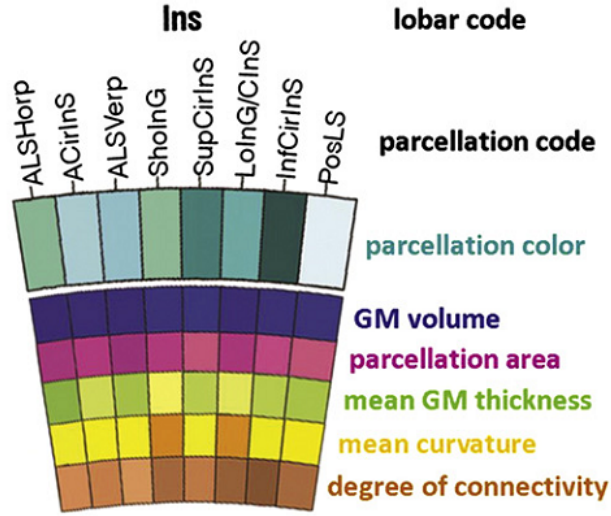




**Figure 3.6:** The connectogram depicting connectivity between brain regions [36].

information and links to external resources as well as for computation of statistical information. Selections can also be stored for future analysis and comparison.

Li et al. implemented a tool for facilitating *quantitative analysis of brain connectivity* in order to improve the understanding of how brain functions are integrated [42]. The tool relies on the identification of regions of interest (ROIs) for brain network construction (Figure 3.9 a)). The software allows neuroscientists to integrate multi-modal neuroimaging data, including structural volumes, cortical surfaces, fiber tracts and fMRI signals, for visual construction and assessment of functional ROIs. An ROI is a brain region that is functionally specialized as one unit. ROIs provide the substrates for measuring the structural and functional connectivities within individual brains and



**Figure 3.7:** Detailed legend of the connectogram's layers [36].

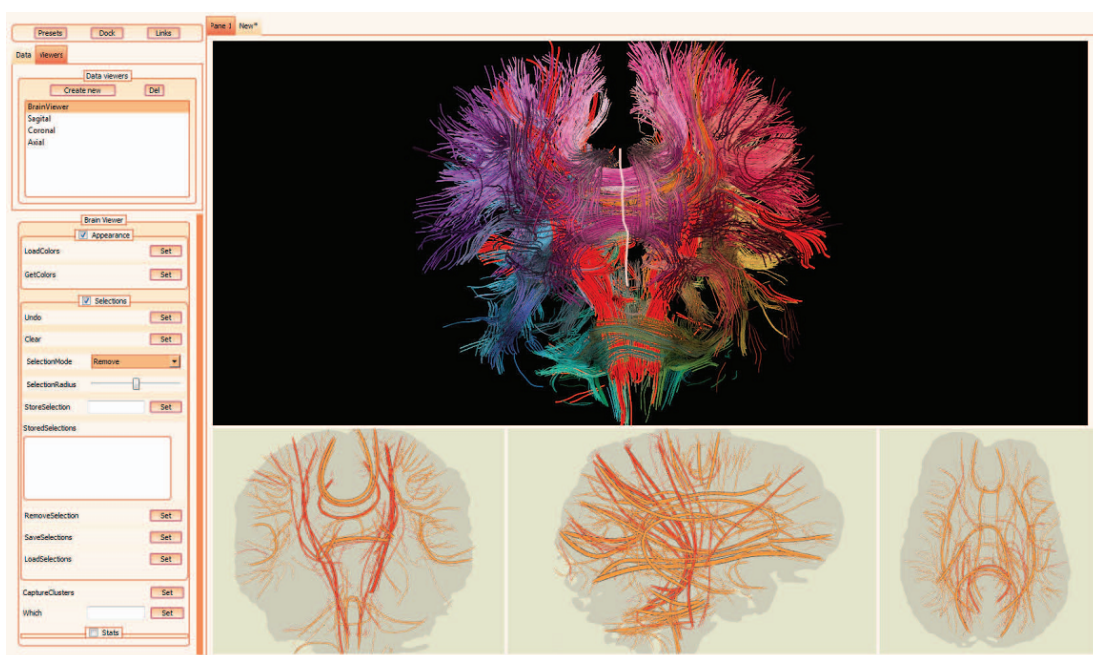
for integrating data across populations. The software implements two types of brain networks: functional connectivity networks and effective connectivity networks. In the functional network, connectivity is defined by the Pearson correlation between signals. ROIs are represented by spheres. Connectivity strength is represented by the width and the opacity of the edges (a thin and transparent edge marks a weak connection). White edges mean positive correlations and green ones signify negative correlations (Figure 3.9 b)). In the effective network connectivity is defined using the Granger causality. Arrows represent causality directions. The width and the opacity of the edges stand for connectivity strength (Figure 3.9 c)).

## 3.2 Biomedical Network Visualization

While network visualization of neural structures is still in a relatively early stage, a wide range of biomedical network visualization tools has been published in other areas as discussed in previous work [29], [54], [3]. Many of these tools are very specialized and focus on tasks like handling protein interaction [7] [50], gene expression [35] [6] or metabolic profile data [49] [47] and connect directly to associated public databases [21] [35], some allow a more general use [66] [77] [5].

Judging by its number of citations *Cytoscape* [66] seems to be the tool with the biggest userbase. Even though its initial purpose was the visualization of molecular interaction networks, it is capable of handling other types of data, as it is open-source and can be extended by a variety of plugins. These plugins provide additional functions like analysis tools and interfaces to scientific databases. Cytoscape allows the user





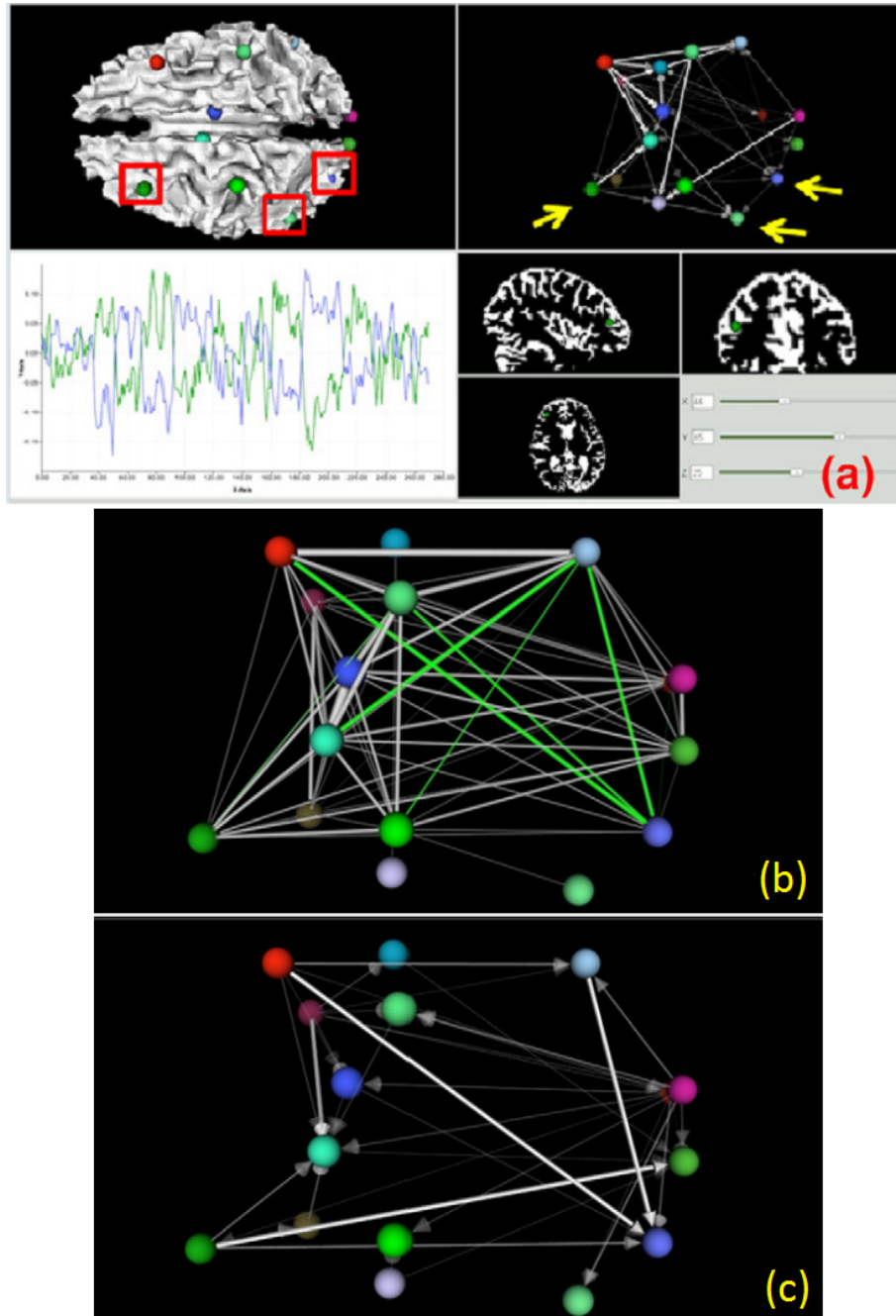
**Figure 3.8:** 3D view and two-dimensional neural maps of in vivo tractograms of the human brain [37].

to customize the graph's appearance and to choose between different graph drawing libraries for the layout of the network.

*Cerebral* [7] is a Cytoscape plugin developed for analyzing protein interactions. It allows the user to pose location constraints on the graph's structure by assigning the graph's nodes to different layers according to the value of a selected attribute. Using this approach it tries to emulate the visual style of traditional pathway diagrams (Figure 3.10). *Cerebral* allows comparison of different experimental conditions by displaying abstract overviews of the different graphs. Nodes of interest can be assigned to individual groups. Currently *Cerebral* only supports structures that follow a linear hierarchy.

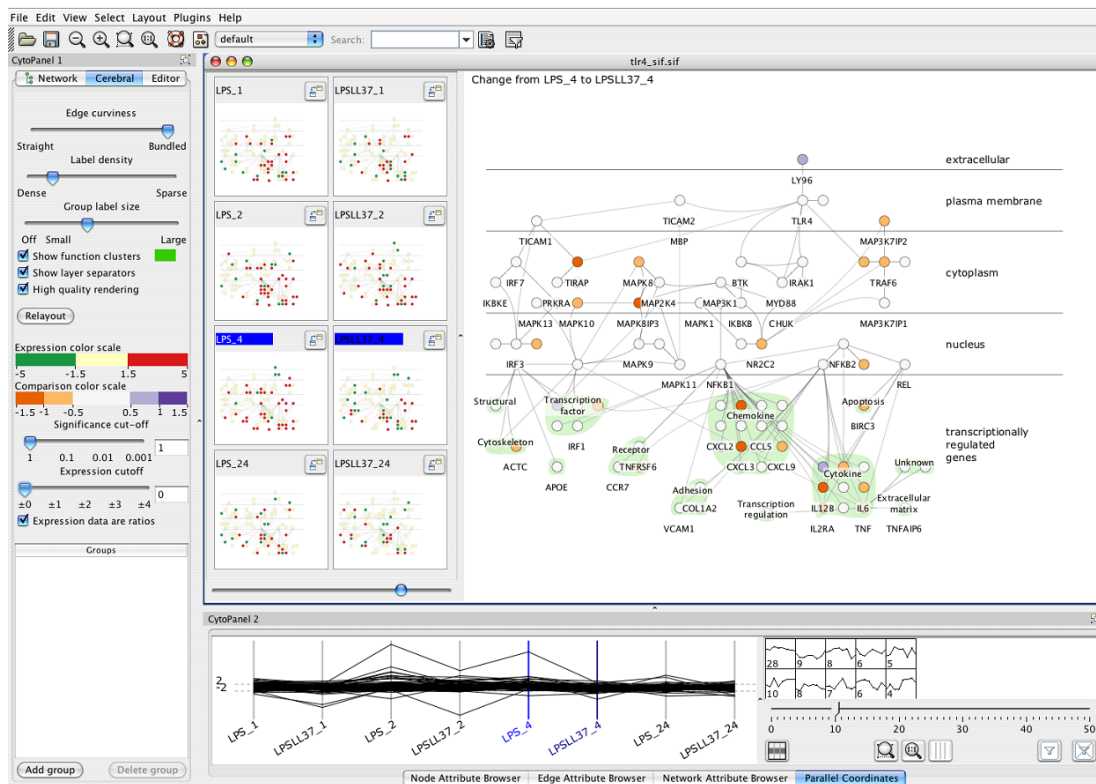
*Tulip* is an information visualization framework and graph drawing API dedicated to the analysis and visualization of relational data [5]. It is designed for handling large graphs and supports automatic cluster creation. *Tulip* is a very powerful software as it can be extended by user made plugins in order to add features like additional compatible file types, applicable layouts, visual encoding and interaction types.

Oeltze et al. published a tool built in the SimVis framework to visualize and analyze toponomics data [50]. It utilizes a graph view with a circular layout as an interaction network to supplement the 3D view of the data. Graph nodes represent affinity reagents and edge bindings between them. *VisANT* [35] is a network visualization and analysis



**Figure 3.9:** a) The workspace of Li et al.'s brain network analysis tool, b) the functional connectivity network, c) the effective connectivity network [42].

tool for gene ontology that offers data mining in standard databases like KEGG [51].



**Figure 3.10:** Cerebral within the Cytoscape workspace [7]. The large view in the middle shows the layered graph, while the small views to its left show different experimental conditions. A parallel coordinate view for data exploration is situated at the bottom of the window.

Arena3D is a tool for visualizing interaction networks in a layered three-dimensional space where each type of data is assigned to a specific layer [53]. Kojima et al. proposed a layout algorithm that provides meaningful layouts for the depiction of biological regulations in a cell by incorporating spatial constraints for the graph's elements [39].

### 3.3 Open Challenges in Biomedical Network Visualization

Network visualization in general still faces some challenges that are relevant also to the visualization of biomedical networks, e.g., displaying and interacting with large graphs. There are some challenges though that are relevant especially in the context of biomedical graph visualization. In their survey Albrecht et al. state that the visualization of biological networks does not typically apply state-of-the-art graph drawing techniques

and that graph drawing tools do not respect the drawing conventions of the life science community [3]. They note that the layout of networks should be in agreement with generally accepted or standardized biological drawing conventions and constraints that originate from recognized textbook and poster layouts. This way the user's attention should be drawn to relevant system properties that might remain hidden otherwise.

A part of these visualization and interaction challenges arises from the variety of data types that occur in biological networks. The data can be encoded in the structure of the network as well as represented by the network layout, or as graphical or textual annotations. Data can be primary (i.e., directly measured), secondary (i.e., derived, inferred, or predicted), or a mixture of both. Albert et al. list three types of biological networks (gene-regulatory, protein interaction and metabolic); neural networks are not mentioned in their survey. Still, some of the open problems they describe are also relevant in the context of neuronal network data.

## **Incorporating Spatial Constraints**

Many biological entities that can be represented in a network have properties like their position in their surroundings that are of high relevance to (or at least help to understand) their meaning in the context of the whole network, e.g., the different parts of a cell, or in this case the positions of neurons in the brain. When this is disregarded, a network's implicit information is lost. If neurons are placed without considering their location, it is not possible anymore to observe flows of information between sections of the brain. It is therefore essential for the understanding of a network's function to integrate this spatial information into the layout. Tulip supports fixed node positions and the tool of Li et al. uses the actual positions of the defined ROIs. However it is sometimes necessary to define flexible regions, that reposition themselves when the network grows, e.g., to avoid node overlaps. Cerebral allows the specification of flexible constraints along hierarchical layers, though only in one dimension.

## **Visualization of Multiple Attributes**

Analyzing data in a life-science context often requires the consideration of multiple data attributes. It is therefore often a challenge to visualize them in a way that does on one side not hide important information and on the other side not confuse the user with too much information. The simplest and most straightforward way would be to map all given data onto nodes and/or edges. Depending on the number of attributes the visualization of a node can become quickly overloaded. An alternative would be to use additional views combined with brushing and linking techniques. From the above discussed techniques most use simple node representations with just a text label (e.g., FlyCircuit) and color coding (e.g., the interactive *C. elegans* neural network). Irmia et al. encoded a variety of attributes in each slice of their circular connectogram. Li et

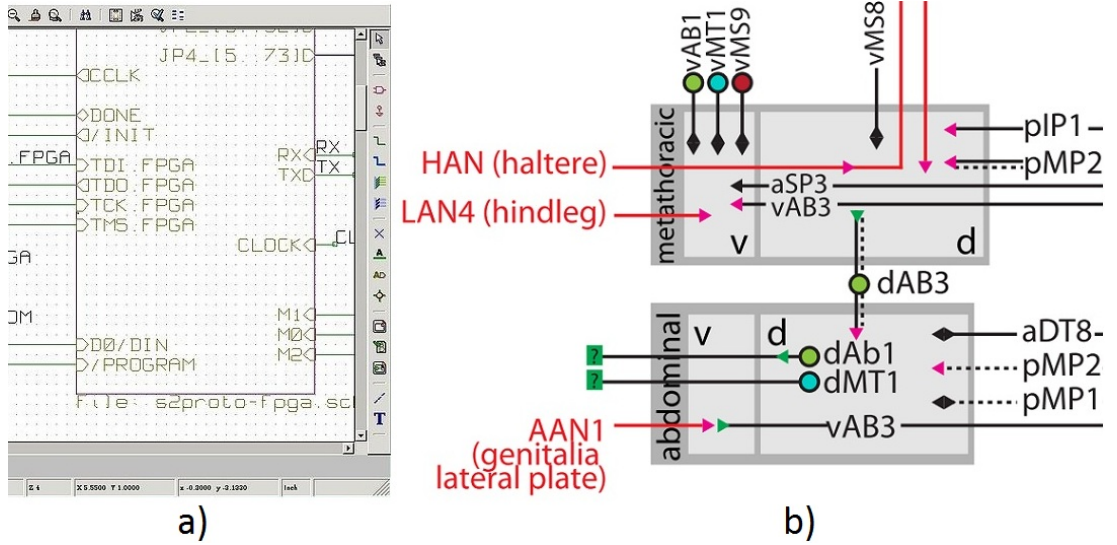
al.'s brain network analysis tool offers multiple linked views on the data, as does Tulip which additionally allows the inclusion of custom node mappings via plugin extension. However, the optimal information encoding depends heavily on the underlying data and the intended purpose of the visualization. Visual encoding is discussed in more detail in the next chapter of this thesis.

## **Visualization of Flows and Paths**

The visualization of signal flows is an inherent part of network visualization. In the context of life sciences though these flows have to be visualized with consideration of uncertainties in the data, e.g., the uncertainty of a connection depending on specific attributes. In the context of this thesis the uncertainty of a connection lies within the overlap between the two examined neurons. Users are in these cases primarily interested in the main or most certain paths through a network, i.e., paths that possess a considerable load of the overall network flow. Especially in mixed graphs standard shortest-path or nearest-neighbor highlighting is not sufficient to emphasize the connections between selected elements. Neuron Navigator possesses no means to communicate the certainty of a connection. WormWeb's tool varies the arrow size of edges according to the number of synapses between neurons. Irimia et al.'s connectogram signifies degrees of connectivity by color and transparency. In Li et al.'s brain network analysis tool the degree of connectivity is signified in the width and opacity of edges. None however implemented special highlighting techniques for emphasizing connections between selected elements.

## **Exploration of Hierarchical Networks**

The exploration of hierarchical networks mainly concerns networks that are so large that they need to be broken up into hierarchies in order to be able to reasonably interact with and visualize them. In other cases the network possesses a natural hierarchy, i.e., subgraphs that represent the inner structure of parent compartments. Albrecht et al. state that in the context of biomedical networks, biologically meaningful visualization of selected subsets and their interrelations, as well as techniques for the navigation within the network are needed. Layout changes resulting from user interaction, i.e., expand and collapse mechanisms for hierarchies, should preserve drawing conventions and the mental map of the graph as well as predefined or relative positions of graph elements. Of the discussed approaches, only Tulip offers proper support for specifying and interacting with node hierarchies.



**Figure 3.11:** Comparison of visual features of KiCad [69] (a), and Yu’s diagram (b). Yu draws the arrow tips of incoming edges on the inside of the target node; different arrow categories communicate the type of the incoming component.

### 3.4 Parallels to Circuit Design

Electronic design automation (EDA) is a category of software tools for designing electronic systems such as circuit boards. EDA tools, such as Cadence [70] or KiCad [69], assist in *designing, testing and verifying* integrated circuits. These tools aid in achieving logical correctness, efficient routing of clock signals, and maximizing circuit density. Circuit neuroscience on the other side is defined as the *understanding* of computational functions of neural circuits in general, and about *visualizing* potential neural connectivity in the case of neuroMap. While Yu’s diagram borrows some visual aesthetics from EDA tools (as can be seen in Figure 3.11), the tasks of the target users and the semantics of the data in this field are quite different. EDA software solutions were therefore not considered for the implementation of neuroMap.

### 3.5 Conclusion

The presented tools and frameworks are either geared towards specific domains or serve general purpose network visualization. Domain specific tools often do not apply state-of-the-art methods in layout and interaction. General purpose network visualization frameworks like Tulip on the other side offer a broad range of state-of-the-art features. An effective tool though needs to be tailored to the respective data and task.

Neuron Navigator [43] is able to query for neurons and connections in user defined regions of the *Drosophila*'s brain. Though due to the absence of overlap information (visually as well as in the database) the result only returns objects that are in the same defined region. There is also no way to filter or query for specific overlaps or distances between objects.

Bhatla's interactive neural network of the *C. elegans* [10] shares a lot of common ground with neuroMap, in that the focus of the application lies on the visualization and exploration of neuronal connections. In this species though all neurons and their connections are already identified and annotated. The interaction is therefore geared towards browsing the database for neurons of interest and retrieving associated information, and not towards discovering new potential connections. Similarly the Partner Tree [20] gives an overview of neuron partnerships in *C. elegans* but offers only a textual query interface to interact with the underlying data.

The connectogram [36] is a static visualization that depicts the connectivity between brain regions, e.g., not at single cell resolution. Because the generated image is static, further exploration of the underlying data is not possible.

Jianu et al.'s two-dimensional tractography visualization [37] does not display a network of nodes and links but rather paths that connect brain regions. The layout is fixed since the paths represent an abstraction of actual neural tracts. At the time of writing the amount of visible data is only adjustable by changing the zoom level which can be problematic when displaying many closely located tracts.

Li et al.'s brain network analysis tool [42] renders the ROI nodes at their actual three-dimensional spatial positions. This has the advantage of giving a direct reference to the linked 3D brain model view but since the graph occludes itself, the whole network is never displayed at once and can only be comprehended by rotating the view accordingly.

Of the discussed systems only the work of Li et al., Neuron Navigator (both in 3D), and Jianu et al.'s neural maps consider the spatial attributes of the displayed data in abstract or accurate form. Only WormWeb's tool, the Partner Tree and Neuron Navigator use single cell resolution data. All of them offer limited means of interacting with or exploring the displayed data.

The integration of neuroMap into Cytoscape [66] or Tulip [5] as a plugin would have been theoretically possible. However, one of the design requirements for our tool was a tight integration into the existing framework that our clients are using during their research. Integration into a third party tool was therefore not considered the optimal solution.

The approach that we took with neuroMap in the context of single cell resolution neural networks and graph visualization/interaction is therefore completely new, as will be documented in the following chapters of this thesis.





# CHAPTER 4

## Methods

In this chapter the basics of the techniques that went into neuroMap’s design and implementation are explained. The main focus lies on graph drawing and the related information visualization aspects like encoding and interaction. The terminology discussed in this chapter builds the foundation for the understanding of the following sections.

### 4.1 Graph Drawing

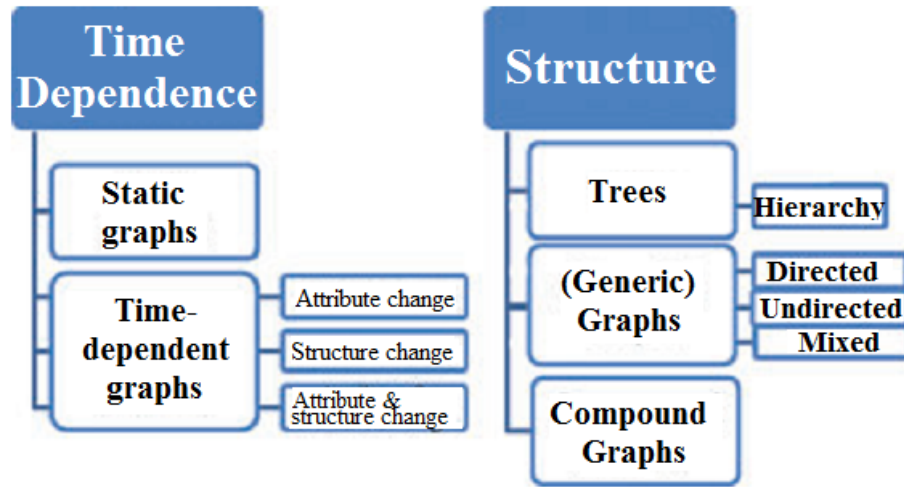
#### Definition

A graph in general describes relationships between entities [59]. These entities are represented by nodes (vertices), the relationship between two nodes is represented by an edge. A graph is thus defined as a pair  $G = (V, E)$ ;  $E \subseteq [V^2]$ , where elements of  $V$  are nodes and elements of  $E$  are edges. When these elements hold additional information like type or weight (depending on the application) the graph is also called a *network*. Since in the context of this thesis both terms apply, we use the terms graph and network interchangeably.

#### Classification

Depending on the character of the relationships between nodes, graphs can either be classified as *directed* or *undirected* [22]. For a directed graph the edge vertices  $e = (v_1, v_2)$  are ordered, for an undirected graph respectively unordered. When a graph contains both directed and undirected edges, it is classified as *mixed*.

A sequence of connected nodes is called a *path* of length  $s$  and is defined as  $path_G(v_1, v_s) = v_1, v_2, \dots, v_s$  where  $v_i \in V$  and  $(v_i, v_{i+1}) \in E$ . If a graph  $G$  contains a closed path (or cycle) with  $a_1 = a_s$ , it is called *cyclic*. If no such paths exist within



**Figure 4.1:** Graph classification by structure and time-dependence [74].

$G$ , it is called *acyclic*. A *tree* is defined as a connected undirected graph without cycles. A tree  $T$  is called *rooted* when one node  $r$  is specified as *root node*:  $T = (V, E, r)$ . A rooted tree can represent a *hierarchy* where the length of the path from node to root node specifies the node's level in the hierarchy. A hierarchy is therefore a directed acyclic graph.

A *compound graph* is a graph where nodes are subdivided into different groups [59]. It is defined as  $C = (G, T)$  with graph  $G = (V, E_G)$  and rooted tree  $T = (V, E_T, r)$  that share the same set of nodes.  $T$  describes the relationships between nodes, as two nodes that share a common parent in the tree belong to the same group. Compound graphs can be created from regular graphs by *aggregating* nodes, i.e., new *super-* or *group-nodes* are created and nodes of the original graph are added as their children. The attributes of these group-nodes can be calculated from the attributes of the aggregated nodes and edges.

Graphs can also be categorized into *static* and *dynamic* (time-dependent) [74]. For dynamic graphs, structure and attributes may change over time, while for static graphs they are constant. These changes can be visualized via animation or time-series for example. Figure 4.1 shows a classification of graphs according to structure and time-dependence.

Additionally certain graphs have elements with *geographic reference*, which means that nodes and/or edges have an inherent geographic location that needs to be considered when generating a layout [74]. These positions can pose more or less strict constraints on a layout algorithm (depending on the visualized data), like city locations on a world map for example.

## Topological Properties

Graphs can also be classified by their topological properties like the number of nodes, the density of the graph and the connectivity [59]. When choosing the optimal visualization method (layout) for a graph of a certain composition these properties are taken into account.

The graph size ( $|V|$ ) is represented by the *number of nodes*, while the *density* is the actual number of edges relative to the maximum potential number of edges  $D = \frac{2|E|}{|V|(|V|-1)}$ . A graph is denoted as sparse when its number of edges is in the range of  $O(|V|) < |E| \ll O(|V|^2)$ , while dense graphs have density values close to one. If the density of a graph is one (which means that the maximum number of edges is reached), the graph is called *complete*. The *degree* of a node is the number of connected edges, respectively neighbors (in- and out-degree for in and out going edges if the graph is directed). When there is a path from every node to every other node in the graph, it is called *connected*. If just a subset of a graph is connected, it is called a connected component. A graph is called *biconnected* if it has no node that would disconnect the graph upon removal. A spanning tree of a connected graph  $G$  can be defined as a maximal set of edges of  $G$  that contains no cycle or as minimal set of edges that connect all nodes.

The drawing of a graph is *planar* if no edges intersect; the graph itself is planar if it permits a planar drawing. According to Euler a graph with  $n$  nodes can have no more than  $3n - 6$  edges to remain planar.

Although there is no standardized definition, graphs are often referred to as large, medium or small; depending mainly on the graph size ( $|V|$ ) but also on density and connectivity. Large graphs in general provide many challenges: when visualizing them (cluttered displays), when processing them (computational times and memory footprint) and also when interacting with them (e.g., selection of overlapped elements).

A classification of the graphs generated by neuroMap is given in section 5.2, based on the abstraction of the neural data to graph elements and on the composition of these elements.

## Graph Visualization

The proper visualization of a graph requires the appropriate type of visual graph representation, efficient placement of graph elements according to the representation type and efficient visual attribute mapping (i.e., the design of the graph's elements) [74].

### Drawing Concepts

Battista et al. introduce three graph drawing concepts to describe the requirements for a „nice“ drawing: *drawing conventions*, *constraints*, and *aesthetics* [22].

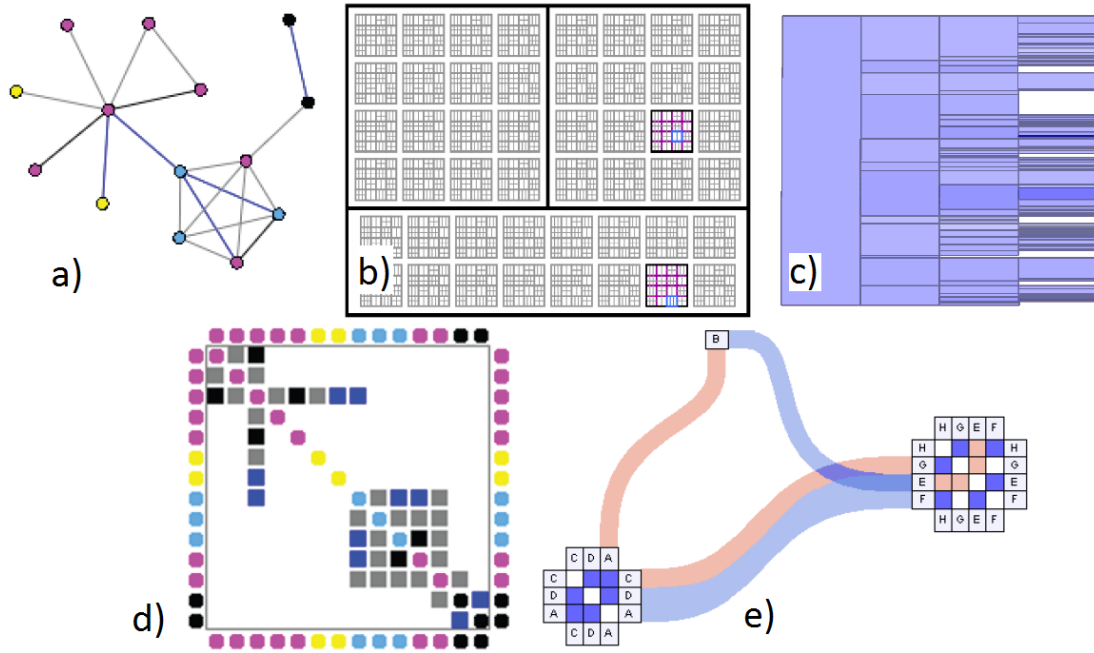
*Drawing conventions* specify basic rules that a drawing must satisfy to be acceptable. Examples for such conventions would be straight-line drawing (each edge is drawn as a straight line segment), orthogonal drawing (each edge consists solely of horizontal and vertical segments), or planar drawing (no two edges are allowed to cross).

In contrast to drawing conventions, *constraints* do not refer to the entire graph but rather to specific elements or subregions. For example they place certain nodes at the center/boundary of the drawing area, they place a given subset of nodes close together (clustering) or draw it in a predefined shape.

When visualizing a graph there are certain *aesthetic criteria* that an algorithm aims to adhere as much as possible in order to achieve readability. Aesthetics are usually optimization problems that are implemented as objective functions. These problems are often computationally hard; therefore many approximation strategies and heuristics have been devised [22]. Standard criteria comprise edge-crossing minimization, symmetry maximization and minimizing the total drawing area. Additionally each type of representation comes with its own aesthetic criteria [58]. A single layout algorithm cannot optimize all of these criteria equally, since some criteria like minimizing the drawing area while minimizing the number of edge crossings contradict each other. Trade-offs are therefore necessary. The optimal layout for a graph thus depends on the application data and the graph's properties. The class of the input graph is also an essential parameter since several graph drawing algorithms work only or work better on certain graph classes. The class, application data and properties of neuroMap's graphs and the resulting advantages and disadvantages of the available layouts are discussed in section 5.4.

## Representation Types

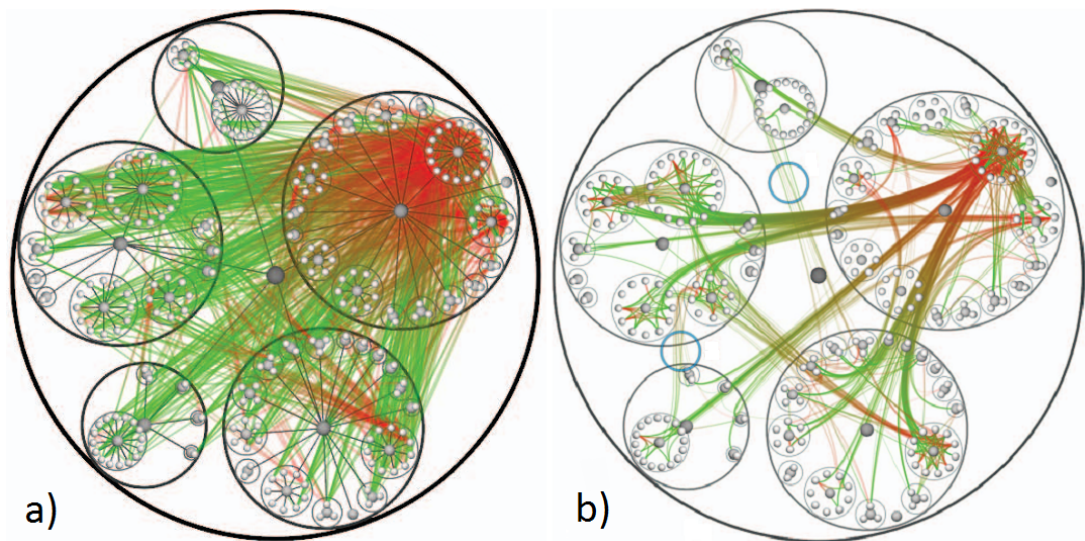
Von Landesberger et al. divide graphs into the three categories (trees, (un-)directed graphs and compound graphs) that have already been described above; each can be represented with multiple visualization techniques [74]. The **node-link technique** is the most relevant in the context of this thesis, since it is the representation form that was chosen for the design of neuroMap. It is also the technique that can be applied to all three graph types. As the name suggests, relationships between nodes are depicted by links in the form of straight or bent lines (Figure 4.2 a)). For this type of representation there exist a variety of layout algorithms that try to place nodes and edges while adhering to the above mentioned drawing concepts. The most important algorithms will be discussed in the next subsection. According to Pohl et al. [56] the main advantages of node-link diagrams are their intuitiveness, compactness and suitability for path following tasks. Additionally most node-link layout algorithms have linear complexity in time and memory for trees and are therefore computationally scalable. The drawbacks are that by design node-link representations leave significant background space empty which leads to scalability problems when applied to larger graphs; they also suffer from increased edge-crossings in large dense graphs. Node-link techniques are therefore bet-



**Figure 4.2:** Graph representation types: a) node-link, b) space-filling (enclosure) [80], c) space-filling (adjacency) [71], d) matrix, e) combined [33].

ter suited for smaller and sparse graphs. In case of strong overplotting of edges in large dense graphs, edge bundling can be used to improve the readability of the display (Figure 4.3) [34].

Another technique applied to trees are **space-filling techniques**. Here the maximal area of the display space is used to visualize the tree hierarchy. Relationships between nodes are not represented by links but via enclosure (for example Treemaps [65], Figure 4.2 b)) or adjacency (Figure 4.2 c)) of child nodes. An advantage is the efficient use of the available display space. Disadvantageous for the enclosure approach is a more difficult distinction of hierarchy structures through overlapping parent nodes. This does not apply for the adjacency approach although at the same time it is not as dense. Additionally there are *hybrid approaches* of node-link and space-filling techniques that present a certain part of the hierarchy as a Treemap and the rest as a node-link diagram, combining the space efficiency and flexibility of the respective approaches. **Matrix-based representation techniques** visualize the adjacency matrix of a given graph as an  $N$  by  $N$  grid where  $N$  is the number of nodes. Position  $(i, j)$  represents the existence of a link between nodes  $i$  and  $j$  (Figure 4.2 d)). If the graph is undirected, this technique results in a symmetric matrix. Additionally there exist different reordering strategies for the rows and columns of the matrix to reveal structures like clusters within the graph [8]. The advantage of matrix representations is that they overcome node overlaps as well as



**Figure 4.3:** A dense graph a) without and b) with edge bundling [34].

edge-crossings, one of the biggest problems in node-link representations, and therefore offer better readability for denser graphs. On the other hand this method does not scale well to large graphs (with thousands of nodes) since each additional node increments the  $N$  by  $N$  grid. Also paths are harder to follow compared to other representation types and the matrix might prove unintuitive to read to the unaccustomed user. For this method hybrid approaches with node-link techniques exist as well (Figure 4.2 e)).

## Layouts

In their survey von Landesberger et al. classify layout techniques for node-link representations according to the type of node placement in the respective algorithm [74]:

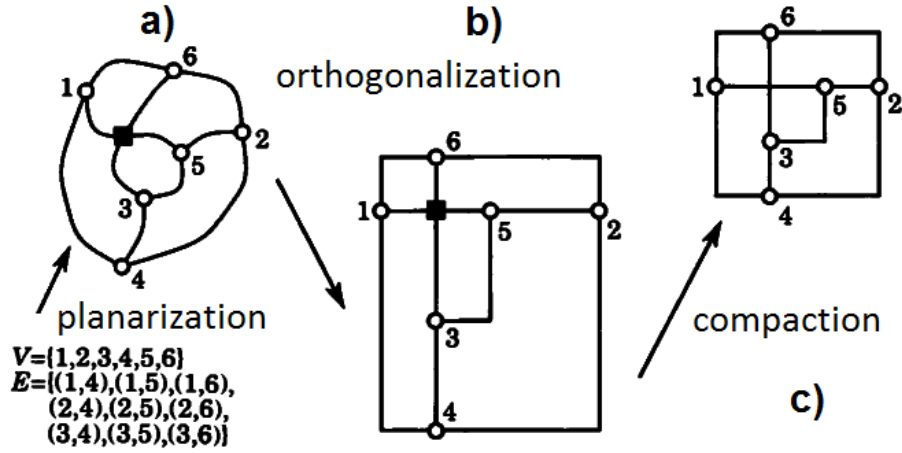
- **Force-based layouts** guide node placement by simulating mechanical laws. Repulsive forces push nodes away from each other while attraction forces act between the end-points of edges. Node positions are iteratively refined until the overall energy or stress of the system is minimized. The quality of the layout can be computed based on the sum of the magnitude of forces on a given configuration [75]. The seminal work by Eades models node repulsion by an electric force between charged particles and edge attraction by spring forces between edge endpoints [24]. Fruchterman and Reingold further improved the node distribution by adapting force models [27]. Force-directed approaches give highly symmetric drawings as well as evenly distributed nodes. Due to its complexity though, this type of layout does not scale well to large graphs of thousands of nodes. To overcome these shortcomings, an efficient GPU implementation [30]

or the deployment of heuristics [26] were proposed. According to Pohl et al. [56] force-directed layouts outperform orthogonal and layered layouts on various user tasks.

- **Constraint-based layouts** extend the force directed approach by allowing constraints on node positions, like the horizontal and vertical alignment of nodes or the specification of edge directions. Orthogonal layouts fall into this category. Constraint-based layouts greatly improve the power of expression but have a slightly higher execution time than force-directed layouts.
- **Multiscale approaches** partition a graph into simpler nested subgraphs. At first a coarse graph is layouted and then more nodes are included with each detail level. Different techniques can be used for creating the node hierarchy as well as for the layout of the resulting layers. An advantage of these methods is that they are typically faster than traditional force-directed methods.
- **Layered/hierarchical layouts** place nodes on parallel horizontal layers and are typically based on the approach by Sugiyama et al. [68]. The approach consists of four steps: 1) cycle removal, 2) assignment of nodes to layers, 3) reduction of edge crossings and 4) assignment of coordinates to nodes. They are described in more detail below. The standard implementation is already quite fast in practice and scales well to graphs of several thousand nodes.
- **Non-standard layouts** combine existing techniques or use new approaches to graph layouting. LGL [1] for example simplifies the graph by computing a spanning tree and then iteratively computes the layout in depth order with a force-directed algorithm. Due to the initial decomposition it scales well to very large graphs with billions of vertices.

In their book [22], Battista et al. specify graph drawing methodologies that are either used by specific layout types or are general purpose techniques that can be applied in various graph drawing algorithms. The most relevant in the context of this thesis are described below (the force directed approach has already been covered above).

The **Topology-Shape-Metrics Approach** belongs to the constraint-based layout family and produces orthogonal grid drawings while allowing homogeneous treatment of a wide range of aesthetics and constraints. In this method the orthogonal drawing is characterized by the three name-giving properties topology, shape and metrics that are defined as equivalence classes between orthogonal drawings of the same graph. Two orthogonal drawings have the same *topology* if one can be obtained from the other by continuous deformation without altering the sequence of edges that contour the faces of a drawing. The drawings share the same *shape* if they have the same topology and one can be transformed into the other just by modifying the lengths of the orthogonal



**Figure 4.4:** The three steps of the topology-shape-metrics approach: a) planarization, b) orthogonalization, and c) compaction [22].

edges. They have the same *metrics* if they are congruent up to a translation and/or rotation. Equivalent to these three properties there are three steps that incrementally refine the drawing: the *planarization step* (Figure 4.4 a)) determines the topology of the graph by minimizing the number of edge-crossings, for example by extracting a planar subgraph of the existing graph, followed by successive reinsertion of non-planar edges. The so formed crossings are represented by dummy nodes to keep the final topology planar. The *orthogonalization step* (Figure 4.4 b)) determines the shape of the drawing by transforming the planar representation into an orthogonal representation of the graph. Here nodes do not have coordinates, instead each edge is assigned a list of angles that describes the bends of the orthogonal edge representation. The *compaction step* (Figure 4.4 c)) finally tries to find the coordinates of nodes and edge bends that produce a drawing with the smallest possible area. Through the sequence of these steps this approach favors topological constraints and aesthetics like node positions and crossing minimization before shape constraints and aesthetics like specific bend sequences and bend minimization, and lastly metrics constraints and aesthetics like vertex coordinates and total drawing area.

**The hierarchical approach:** If the graph contains cycles, the first step removes these cycles by temporarily reversing a subset of the edges. In the *layer assignment step* a layered directed graph (digraph) is formed by assigning all nodes to a layer  $L_h$  so that if  $(u, v)$  is an edge with  $u \in L_i$  and  $v \in L_j$ , then  $i > j$ . Next a proper layered digraph is formed by inserting dummy-nodes along the edges that span more than two layers such that  $i = j + 1$  holds true. In the *crossing reduction step* the vertices on each layer are ordered in such a way that the number of edge-crossings is minimized. The *x-coordinate assignment step* assigns the final x-coordinate to each node while



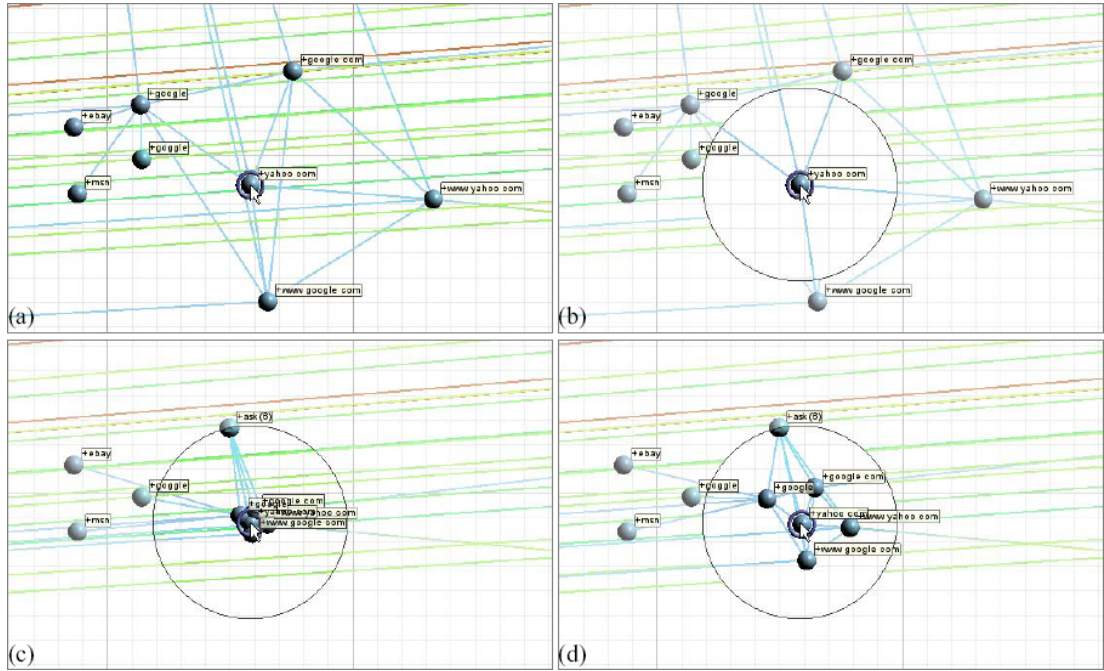
preserving the ordering computed in the previous step. Each edge is then represented with a straight-line segment and finally long edges are represented as polygonal lines by removing the dummy-nodes. Through the fixed step order this approach as well implicitly establishes an ordering among aesthetics. During the x-coordinate assignment step aesthetics like bend minimization can be achieved through alignment of dummy-nodes. Further nodes can be positioned accordingly to emphasize symmetries or to reduce the drawing area. Also constraints like the vertical alignment of vertices are supported during this step.

## Operations on Graphs

Von Landesberger et al. categorize graph interaction into view-, visual abstraction- and data-interaction [74], depending on whether the user action affects the data, the visual display of the data or the view. Preprocessing is listed as a non-interactive graph operation in this context. Von Landesberger et al. further distinguish between *topology-based tasks* like finding adjacent nodes or determining connections between nodes, and *attribute-based tasks* like searching for nodes or edges of specific values or types. Interaction in general helps users solve tasks that are connected to the exploration of a graph. Especially interactions with immediate feedback support exploration for cognitive reasons: exploration requires several hypotheses to be maintained in short term memory. Planning complex operations without feedback or using textual syntax already occupies the short-term memory and therefore significantly hinders exploration [2]. It is also efficient to help users preserve their mental map of the structure. The *mental map* is the image that the user keeps in his mind. It plays an important role in applications where the structure of the graph often changes, either due to exploration or due to the graph's dynamic nature. In an ever changing graph the mental map helps the observer to mentally carry over an image of the previous state into the next state. This enables him or her to easier spot changes between developing states.

### Preprocessing

Graph preprocessing is used to simplify the graph by removing elements while maintaining the overall structure, or to highlight certain parts of a graph upon creation by analyzing the element's properties (for example through grouping or color coding of elements with the same attribute). The reduction of the graph structure can be achieved either by filtering or by aggregation. Graph filtering can either be stochastic or deterministic (based on node/edge attributes). Graph aggregation does not remove elements but instead merges them into single nodes and edges before displaying them, thus rendering a compound graph. Through the merging the graph size is reduced and at the same time the relationships between groups of nodes are revealed.



**Figure 4.5:** Examples of different lens types: a) view without a lens, b) local edge lens, c) neighbor-clustering lens, and d) composite lens with integrates [72].

## View Interaction

View interactions allow the user to incrementally build a mental model of the scene's objects and their interrelations. They comprise standard interaction techniques like *panning* and *zooming* but also specialized approaches like *focus+context techniques* and *guided panning*. Panning and zooming is necessary when the entire data set cannot be presented at a desired resolution. Focus+context techniques like magic lenses (or fish-eye views) show more detail of or allocate more display space to the elements in the focused area of the lens while maintaining the context of the overall graph structure (Figure 4.5). Guided panning allows automatic fly-throughs along the edges of a graph.

## Visual Abstraction Interaction

Visual abstraction interaction changes the type of the visual presentation and its parameters. These parameters are often adjusted indirectly via sliders, check boxes and radio buttons. Direct adjustment, although more intuitive, is only implemented in few systems [74]. The change of visual parameters comprises techniques like *highlighting*, *brushing and linking*, and *semantic zooming*. Changes of the visual scheme consist of *layout change* and *change of the visual representation type*. Highlighting emphasizes

interesting graph elements and their relationship upon selection. Brushing and linking transfers changes in the visualization (like highlighting for example) from one view to all other coordinated perspectives on the data. Semantic zooming decreases or increases the level-of-detail (LoD) of the displayed data depending on the zoom-level. Layout changes do not only comprise the change of the layout type but also the manual adjustment of the layout and the adjustment of layout parameters. Changing the visual representation type can mean adjustment of visualization parameters but also switching to a different mapping that reveals features of the data that were hidden or less obvious with other mappings.

## Data Interaction

Interaction on the data level affects which part of the data will be displayed (*data filtering*) as well as how *values* and structure are changed. *Filtering* can be approached from three directions. *Top down* starts with the complete data and then removes the elements of the graph that do not correspond to selected criteria. This has the advantage of first giving an overview of the whole data from where more interesting parts of the graph can be explored while uninteresting parts can be discarded. *Bottom up* filtering starts from one selected node and then successively shows more elements on demand - either based on the graph structure or on a degree-of-interest function. This way only the most relevant part of the data is visualized but it is difficult to find a good starting point without knowing the complete data. The *middle-out* approach starts with a coarse graph that can be either increased or reduced in the desired direction.

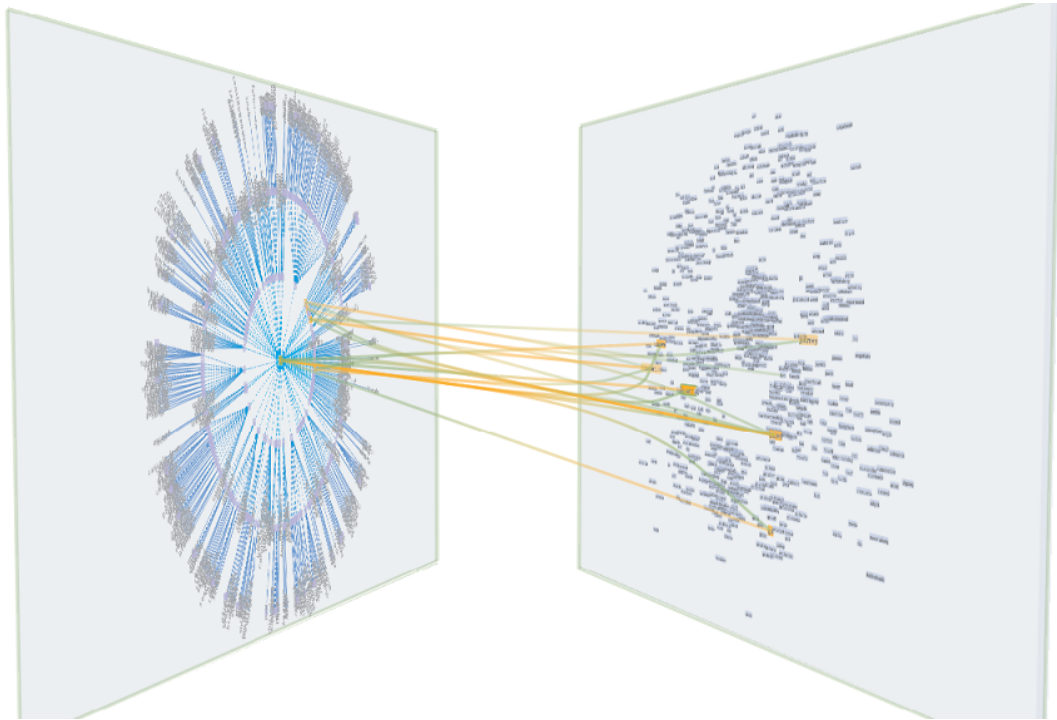
The change of *values* comprises graph editing (deleting and adding elements to the graph) and interactive graph aggregation (merging of nodes as described above) but also selective hiding. *Selective hiding* allows the user to temporarily remove or reduce structures of the graph that are currently of no interest, thus providing more space and attention to more interesting regions.

## Graph Analysis

Graph analysis comprises algorithms for analyzing a graph's structure as well as for automatically comparing two or more graphs.

### Analysis of Graph Structure

The analysis of a graph's structure not only examines the global structure but also the relationships between the graph's single elements. *Important nodes* owe their specific role to the properties of their metrics within a network (e.g., centrality). These metrics can be highlighted within the graph or displayed in additional linked views in order to explore a network by filtering or highlighting specific nodes. The *connections be-*

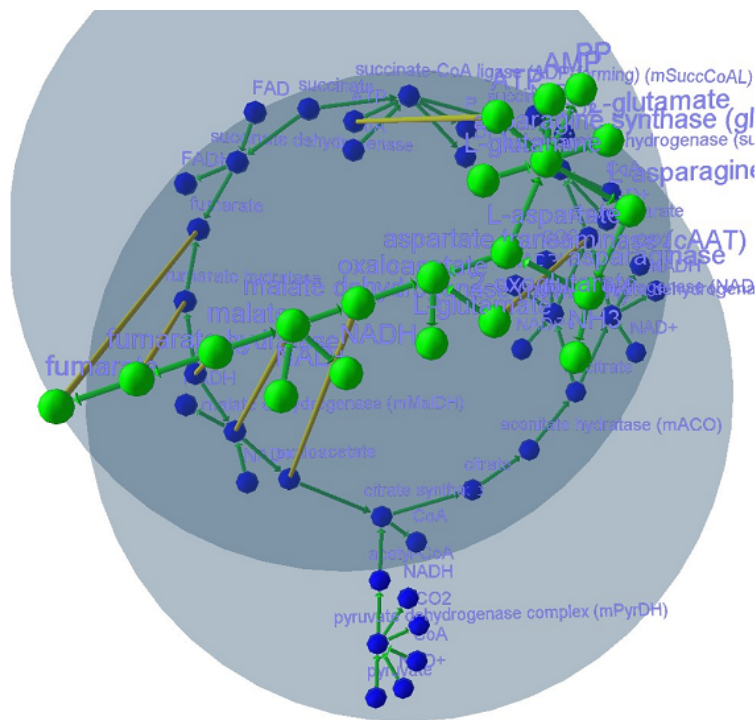


**Figure 4.6:** Graph comparisons: one-on-one node matching between two graphs with VisLink [19].

*tween two nodes* are typically analyzed by calculating and highlighting the shortest path between them. *Substructure analysis* tries to uncover a graphs’s structural design principles by searching for specific types of substructures (so called motifs [45]). Distinct types of networks (like genetic networks or social networks) have a different configuration of motifs. These networks can therefore be defined and distinguished by their composition of motifs. The significance of a certain motif in a network is determined by its frequency of occurrence. The frequency of occurrence of different motifs around a node is called the motif fingerprint of that node. In some scenarios also the identification of the *impact of graph changes on structural properties* can be of interest to the user.

### Analysis of Neural Networks

In neuroscience topological measures have been developed to characterize connectivity in brain networks on a global, regional, and local level [38]. Examples for *global measures* are characteristic path length, degree distribution, clustering coefficient, and modularity. Modularity for example refers to functional segregation in the brain. Other measures characterize functional integration, which is the ability to rapidly combine



4.7) [28]. *Structural differences* between *multiple* graphs can be calculated by comparing their topological properties like size, density and connectedness. According to these properties the analyzed graphs can for example be assigned to different groups.

## 4.2 Abstraction and Visual Encoding

The advantage of images over text is that they permit several pieces of information to be communicated simultaneously, while textual or numerical information has to be read and processed sequentially [8]. A successful visualization conveys the desired information efficiently and accurately to the targeted audience while bearing in mind the task or purpose of the visualization. To efficiently encode information into visual signs, it is necessary to know how observers can read (or decode) the given graphical information [40]. Certain visual encodings can be understood almost instantly (due to elementary perceptual tasks), while others create visual puzzles due to being too complex, too confusing or too distorted. Even aesthetics can influence the effectiveness of a visualization as a visually unappealing picture may reduce the audience's willingness to look at it. Therefore abstraction (mapping data components to graphical entities) and visual encoding (mapping data properties to graphical properties) have to be chosen carefully. Especially since the capacity of human perception concerning how many different stimuli and variations of stimuli we can accurately perceive and measure is limited and dependent on the type and combination of stimuli. When designing a visualization it is therefore important to consider these limitations in order to avoid situations where pictures are ambiguous, misleading or difficult to interpret.

### Visual Alphabet for Networks

Krempel defines a visual alphabet for networks that consists of shapes (and lines), size and color [40]. These overlap with the eight visual variables defined by Ward et al. [75] that additionally include position, orientation, texture and motion.

- *Shapes & Lines:* To distinguish between different classes of nodes, they can be rendered using shapes, icons or symbols. Symbols or icons cannot only serve the discrimination of different node classes but can also convey additional meaning, although they are sometimes limited to specific cultural domains. It is important to consider how well the chosen symbols can be differentiated from each other.

Lines can be encoded by width and line type (for example solid or dotted). Width can lead to overlapping edges in dense graphs, which can be compensated to a certain degree by importance driven edge-sorting.

- *Size:* Size determines how small or large a shape will be drawn. It easily maps to interval and continuous attributes because it supports gradual increments over

some range [75]. The relationship between the magnitude of a physical stimulus and its perceived intensity is described by a power law that depends on the type of stimulus. Steven's Law states that concerning size the human perception of length is almost linear with an exponent between 0.9 and 1.1. For the perception of area size the exponent lies between 0.6 and 0.9; and for volume features between 0.5 and 0.8. This shows that as the dimensionality of an attribute increases, so does the degree with which we underestimate it.

- *Color*: Colors can be used for encoding both interval and continuous data variables. Additionally colors can convey aesthetic impressions, cultural meanings or physiological reactions [40]. The human perception of color can be separated into three dimensions: hue, lightness and saturation. This provides three layers to communicate information by color. The Munsell color system is a color system that encodes these three channels perceptually uniform and equidistant. Uniform means that the change of one parameter does not affect the other two parameters. Equidistant means that a human observer perceives the distance between neighboring colors as equal. Color systems with these properties are able to communicate ordered and quantitative information. When choosing colors for graph elements also the background needs to be taken into account as the impressions of a color scheme vary greatly in contrast to the surrounding area. Color blindness should as well be considered as it affects a considerable number of people. Generally color should be used with caution since it can add visual appeal to a visualization but can also decrease the effectiveness of the communication process.
- *Position*: The position of elements allows to *map* their properties (e.g., category) but also to *reveal* properties of the underlying data (e.g., certain structures or clusters). In the context of graph drawing, the position is seldom a customizable variable since it is usually automatically calculated by the layout algorithm. Exceptions are layout algorithms that allow constraints on node positions.
- *Orientation*: Orientation or direction describes how a symbol is rotated according to its associated attribute. Orientation is directly tied to preattentive vision and best used on symbols with a natural single axis (e.g., arrows).
- *Texture*: Texture is a combination of other visual variables, like the symbols in texture elements, the color of a texture or the orientation of a pattern.
- *Motion*: Motion encodes information in the way one or more of the above mentioned variables change over time; for example by varying the speed at which a change in position or opacity is occurring.

With these elementary graphic attributes multiple sets of information can be communicated independently of each other at the same time. Research in human perception

showed that the different variables are perceived with individual error rates. The ranking (with increasing error) by Cleveland [18] is ordered as follows: position along a scale, length, angle, area, volume, color (hue, saturation). Attributes should be prioritized accordingly. Care should also be taken with the variations of a single graphical attribute, since the human perception only allows accurate judgment of about 7 values. One solution to overcome this limitation is to use more than one stimulus simultaneously. In their book [75] Ward et al. suggest to use redundant mappings (e.g., mapping one attribute to both color and size) when possible to improve the chance of the information being communicated accurately. Further they advise to design a visualization in a way that relies on relative judgment rather than absolute judgment (detection of differences rather than extraction of numeric values), since it is less prone to errors. The variables should also be chosen according to the purpose of the visualization - according to how the user should perceive the information (e.g., ordinal, nominal, quantitative). Of the listed visual variables, neuroMap uses shapes and lines, size, color, and position. Which graph and data attributes are mapped to these variables will be discussed in chapter 5.

## Mapping Data Properties

To create the most effective visualization for a particular application, it is essential to consider the semantics of the data and the user's domain-specific mental model. One of the most common and intuitive mappings in visualization is the mapping of spatial attributes to screen position. Other mappings become intuitive when their particular context is considered, like mapping temperature to color for example. Often color already has specific interpretations in different fields (e.g., cartography, geology). Therefore the application domain of the visualization already may dictate a logical use for the color attribute. In general when selecting a mapping it is important to consider the compatibility of the scale of the data field and the graphical entity or attribute. For ordered attributes like temperature it would not be intuitive to select a non ordered graphical attribute like shape. Respectively, unordered attributes like sex should not be represented by ordered attributes like length. Ward et al. though state that sometimes interesting results can emerge from using non-intuitive mappings [75]. They therefore conclude that default mappings should be based on the most intuitive selection but for exploratory tasks user customization should also be permitted.

## Mapping Graph Properties

Additionally to the attributes of network elements (like type or weight), also the inherent properties of a network can be visualized by encoding them into graphical variables like the node size for example. Encoding the *node degree* makes nodes with more relationships stand out. When *closeness* is encoded, the node size signifies which nodes



can reach many other nodes by short paths. Encoded *betweenness* informs the user which nodes control a larger number of the shortest linkages to adjacent networks. This mapping of graph properties allows the observer to intuitively discover patterns and therefore provides orientation within a network.

## Labeling

Proper labeling in a visualization is essential to allow the viewer to understand what is being shown. A plot without some sort of legend for defining its elements would be difficult to read. In graph drawing the problem of labeling can be more complex, not only because of the potential for many nodes but also because it might be necessary to label edges as well. If there is only a small number of distinct labels that show the type of a link or a node, non-textual labels like color or shape are sufficient. However, if a graph contains more than a small number of distinct elements (e.g., five or six), it becomes necessary to include textual information besides the above described attribute mapping for proper interpretation. This information is displayed in the form of node- and edge-labels. For small graphs it is common to put the labels within the nodes, using rectangular or round node shapes. To avoid distorted node sizes, the size of the nodes should be dictated by the length of the longest label. Similarly for edges the labels can be placed near the center of the edge. To avoid confusion the placement should be consistent, for example always left or on top of the edge depending on its orientation. In larger graphs, displaying all labels quickly becomes ineffective, either due to label overlaps with nodes, edges or other labels, or due to decreased readability when zooming out. Several strategies exist to cope with this problem [75]. One solution is to only show labels in a small region of the graph, for example near the cursor position or on mouse-over. If the density is too high, fish-eye views can be additionally deployed. Another approach is to alternately show random subsets of the labels. Also semantic viewing can be deployed to de-/activate or resize node labels at certain zoom levels.



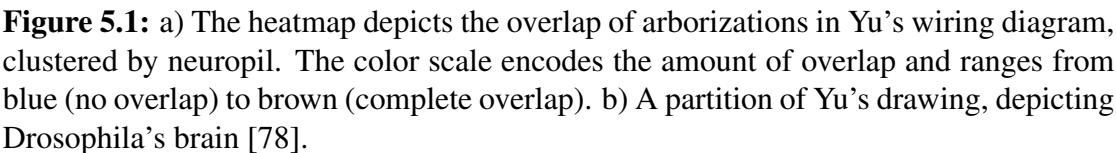
## Visual Encoding

In this chapter the decisions that went into the visual design of neuroMap are discussed. Starting with a more detailed presentation of Yu's wiring diagram, section 5.2 explains which parts of the neuroanatomical information are visualized and how they are abstracted into graph elements. Section 5.3 then discusses how these elements are visually designed to encode the associated information, and in section 5.4 the available layout types are presented.

### 5.1 Yu's Drawing

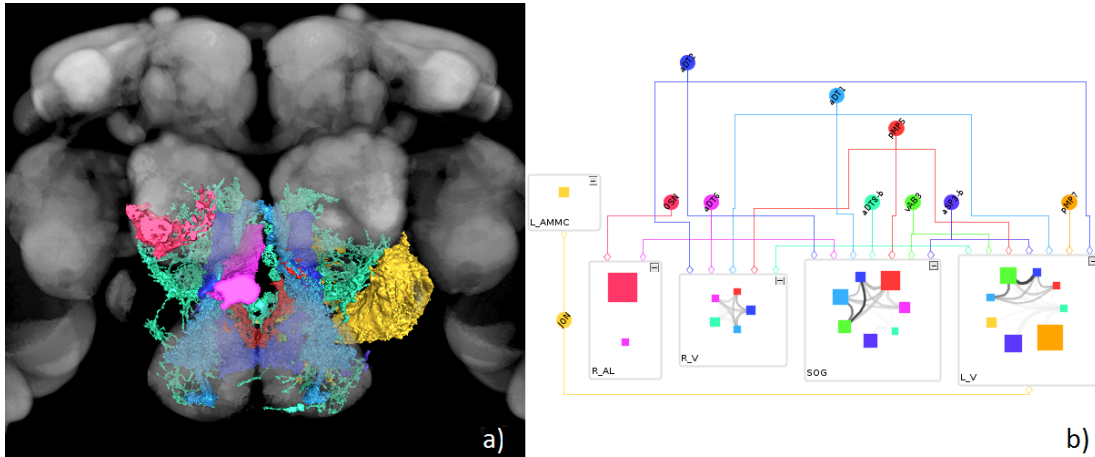
The motivation for and starting point of neuroMap's design process was Yu's wiring diagram of a courtship behavior related neural circuit (Figure 5.1 b)) [78]. The drawing depicts neural pathways of a certain group of neurons that extend from sensory input to motor output. The diagram was used to present the findings that were discussed in the paper to the scientific community and to inspire hypothesis formation about potential functional neural connections.

The visual elements of the graph are cell bodies, projection edges and neuropils. *Cell bodies* are depicted as colored circles with a label of the neuron name. The color depends on the neuronal type. From each cell body *projection edges* lead to each neuropil where the neuron has a synaptic terminal (innervation). The edges are represented by solid or dotted black lines, for neurite or contralateral neurite projections. Sensory afferent neurons are visually distinguished from the other neurons by having pink cell bodies and projection edges. The arrow tip of the projection edge gives information about the type of *terminal*. Presynaptic terminals are represented by pink triangles, dendritic terminals by green ones, and unresolved terminals by a black diamond shape. *Neuropils* are white or gray boxes that contain the terminal symbols. A connection between two



## Semantics of the Displayed Data

50



**Figure 5.2:** 12 arborizations from Yu’s diagram visualized in the volume renderer (a), and in neuroMap (b). Even with only 12 arborizations displayed, it is difficult to judge overlaps in 3D.

drawback of Yu’s abstract representation format, since it is laborious and unintuitive to search the heatmap for the specific grid cell that contains the overlap value. neuroMap therefore includes this information directly within the visualization, thus striving to adhere to the semantics of the data. The focus of the visualization is therefore to convey potential connectivity while distracting as little as possible from it.









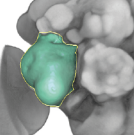


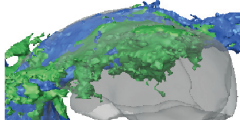


## 5.2 Abstraction to Graph Elements

Similarly to Yu’s graph, a neuron’s abstract representation is partitioned into a single **cell body** and one or more **projections** that lead to the attached **arborizations**. The **overlap** between arborizations that was visualized in a separate heatmap in [78] is represented directly within the graph in neuroMap. The fly’s brain and VNC are represented by the 60 **neuropils** that they are composed of. An arborization can overlap and therefore lie in one or more neuropils. neuroMap offers a **Simple View** on the data where neuropils are omitted and arborizations are not split up, and a **Standard View** that displays neuropils and partitions arborizations between them accordingly. Figure 5.3 displays neuroMap’s graph elements in comparison to Yu’s diagram and their anatomical counterparts.

Like in Yu’s graph, *cell bodies* are represented by nodes. *Projections* are drawn as edges that denote the relation between cell body and arborization.

In contrast to Yu’s drawing, *arborizations* are represented as actual nodes. By giving the arborization its own representation the multitude of associated information (like size, neuropil overlap, or the fly’s sex) can be directly encoded within the visualization. An *overlap* between two arborizations is represented in the form of an edge that connects

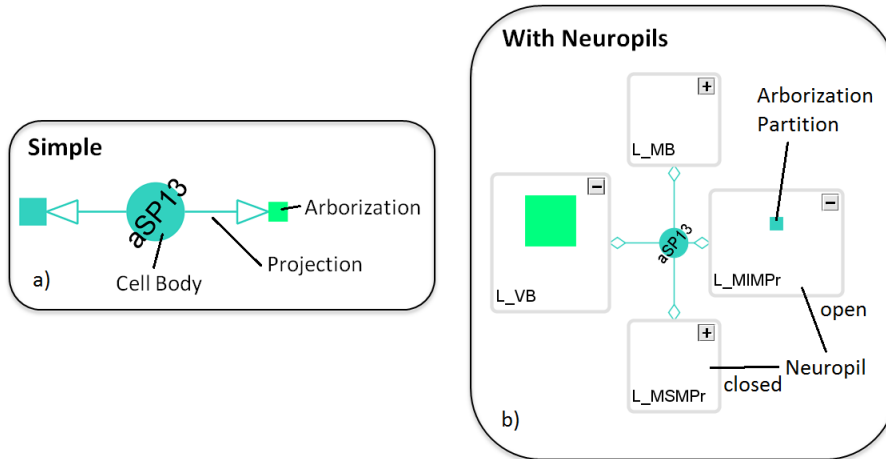
these two arborizations. *Neuropils* are represented as super nodes that accommodate all innervating arborizations.

	Anatomical	Yu's Diagram	neuroMap
Cell body		aSP4 	
Arborization		n/a	 aSP4_arbM
Projection			
Neuropil		 AL	 AL
Overlap			

**Figure 5.3:** Direct comparison of neuroMap's elements with their anatomically accurate counterparts and representation in Yu's graph.

## Simple View

The purpose of the *Simple View* is to reduce the displayed information to a minimum and to set the focus on the overlaps between arborizations. This is achieved through the omission of neuropils. Since the brain is not partitioned into neuropils, arborizations do not have to be partitioned as well, and are rendered as single nodes (Figure 5.4 a)). The overlap between two arborizations is therefore represented in a single edge. Projection edges lead directly from cell body to arborization node. The number of projection edges that originate from a cell body node therefore only depends on the number of displayed arborizations that belong to the neuron. This results in a graph with a reduced number of nodes and edges in comparison to the Standard View, and a simpler, more direct overview of potential neuronal connectivity. However, the tradeoff of the simpler view, is the loss of all locational and semantic information that is encoded through neuropils.



**Figure 5.4:** Simple View (a), and Standard View (b): Abstraction of a single neuron with and without innervations into specific neuropils.

## Standard View

The *Standard View* should give the user the whole range of available information. The neuropils that overlap with the loaded arborizations are therefore included. In contrast to arborizations, the positions of neuropils across test subjects does not vary (as they are part of the standard brain template). Neuropil nodes can therefore be assigned to fixed compartments and give locational context to the displayed data. Also, neuropils are often associated with a certain type of functionality and can therefore give implicit information about the neurons that overlap with them and about the potential connections within them.

Since arborizations can overlap with multiple neuropils, in this view a single arborization is represented by multiple nodes, one in each overlapping neuropil (Figure 5.4 b)). Overlap edges are therefore split up between the neuropils that contain the corresponding arborizations, as well. Projection edges lead from the cell body to the neuropils that encapsulate these arborizations. The number of projection edges that originate from a cell body node therefore depends not only on the number of displayed arborizations but also on the number of neuropils with which these arborizations overlap. Through the inclusion of additional nodes and edges, the resulting graph is more complex than its equivalent in Simple View but at the same time gives more detailed information about the displayed data.

## Classification

Summarized, the basic elements of neuroMap are cell body and arborization nodes, projection and overlap edges, and neuropil super nodes. Using the terminology from chapter 4, the graphs rendered by neuroMap can be classified as static mixed compound graphs. *Static*, since the graph's content and structure does not change over time. *Mixed*, because the graphs contain both directed edges: projections lead from cell body to arborization or neuropil; and undirected edges: overlap edges connect two arborizations but do not indicate source or target. The classification as *compound* graph stems from the neuropil super nodes that contain arborization nodes and therefore subdivide the graph structure into different groups. neuroMap's graphs are generally sparse with a density  $D < 0.1$ . When regarding the subgraphs within neuropils separately though, the density can be higher if many arborizations overlap each other. Due to the low density the overall graph structure is planar. Subgraphs though can be non-planar due to their potential high density.

## 5.3 Visual Encoding

In general the visual design of graph elements has been geared to adhere to the semantics of our clients' data. Elements that are more likely to be part of a connection between neurons have therefore more visual presence than elements that are less likely to be part of a connection. The assessment of this likeliness follows Peters' rule (introduced in Chapter 2.3). Following this rule, bigger overlaps between objects are deemed more important than smaller ones. How this is manifested in the design of the individual graph elements is described in the following.

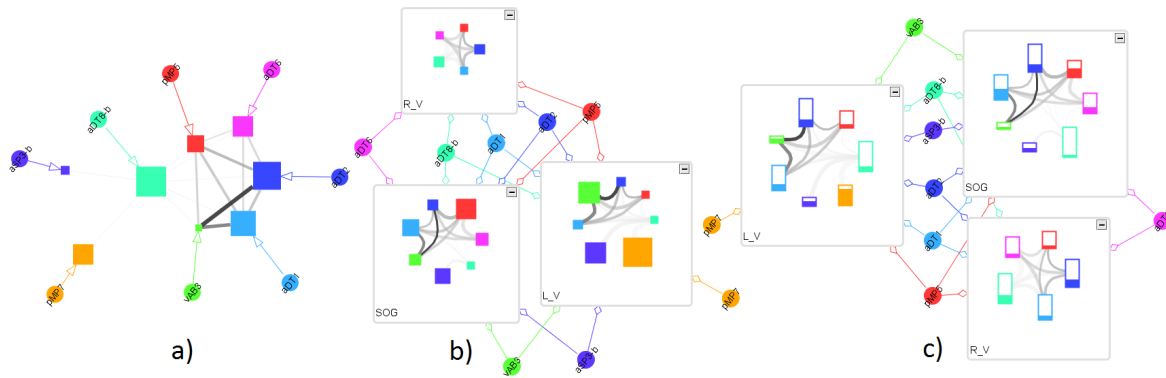
### Cell Bodies

Like in Yu's drawing, cell body nodes are depicted as circles. In neuroMap however, the node size is not uniform but scales with the number of connected projection edges to indicate neurons with a bigger number of innervations. The scaling is capped to avoid nodes that are too large or too small. Nodes are large enough for labels to be drawn across instead of beside them. This saves space on the canvas and avoids overlaps with other graph elements.

### Arborizations

The visual design of arborization nodes depends on the selected viewing mode. In *Simple View*, arborizations are represented as squares. Since in this mode each node represents an entire arborization, the node size is scaled by the arborization's actual volume. Because arborization volumes differ quite dramatically in size, with the smallest





**Figure 5.5:** Comparison of all three arborization encoding types: Simple View (a), Standard View (b), and detailed encoding (c) on the same dataset. In Simple View and detailed encoding the node size represents the total volume of an arborization, while in Standard View the size represents the overlap percentage with the respective neuropil. The overlap percentage is still present in detailed encoding within the node filling.

one measuring only 202 cubic microns, and the largest one 679722 cubic microns, the applied scale is logarithmic. This way the differences in size are still visually judgeable while the smallest and largest node size are kept within a reasonable area.

In the *Standard View*, arborizations are split between their overlapping neuropils. To let the user easily grasp the distribution of an arborization over its overlapping neuropils, the node size of each partition is scaled according to the arborization's overlap with the respective neuropil. The more percent of an arborization overlap with a neuropil, the bigger the innervation node is drawn within the neuropil node. A drawback of this encoding method is that the node size does not convey information about the actual volume of the partition in each neuropil. A small partition of a very large arborization will therefore appear smaller than a large partition of a very small arborization, even though the former is a few times larger than the latter. This can be misleading in some scenarios (see Figure 5.5 a) and b)). Even though the arborization of the green colored neuron is much smaller than the arborization of the turquoise colored one, it appears larger in neuropils SOG and L\_V because its overlap with these neuropils is bigger.

To eliminate this potential confusion, *detailed encoding* was added to the Standard View (Figure 5.5 c)). Here nodes are represented as vertically oriented rectangles. The rectangles are scaled vertically according to the total arborization volume, and filled according to the arborization-neuropil overlap volume. The view thus simultaneously encodes the information of the Simple View (total arborization volume) and of the Standard View (arborization-neuropil overlap). The same scaling as in Simple View is applied. Here only the height of the nodes is scaled, since (as described in chapter 4.2) variations in one dimension are better judgeable than variations in two dimensions, es-

pecially when multiple attributes are visualized simultaneously. Similarly to cell body nodes, arborization node size is capped to avoid too large or pixel-sized nodes.

If objects from male and female subjects are visualized simultaneously, arborization nodes are framed by a sex specific color to visually distinguish them from each other. Additional information about an arborization is made available through semantic zooming and tooltips, both of which will be discussed in the next chapter.

## Projections

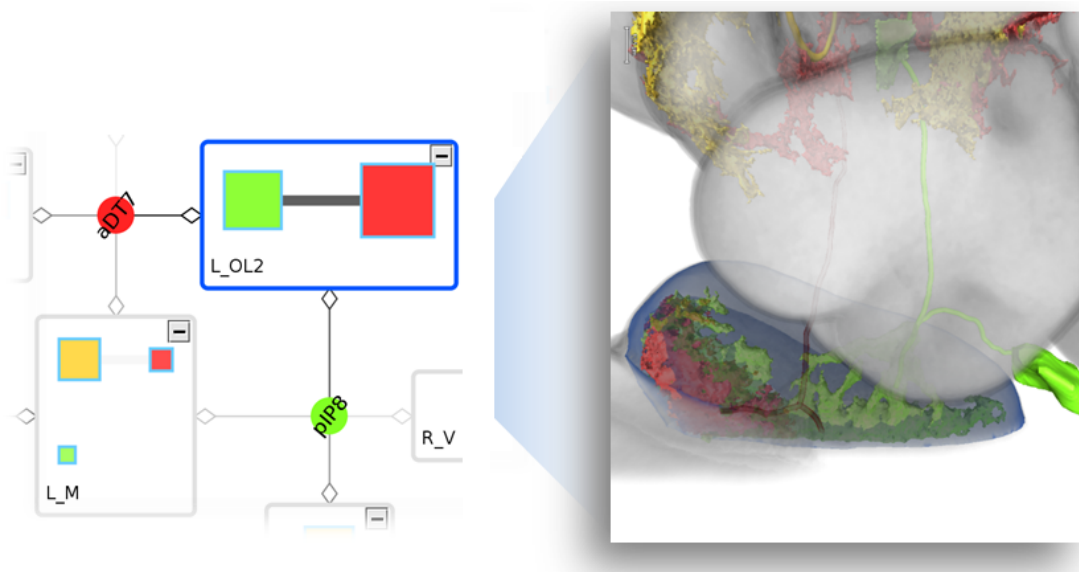
Projection edges tie a cell body and its associated arborizations together. In Simple View projection edges are colored using the cell body color (Figure 5.5). For a graph with neuropils there are two selectable encoding types. The first uses the same encoding as the Simple View. This visually links a path on the canvas to its associated cell body and therefore makes orientation within the graph easier. The second encoding mode is designed to visually enforce the arborization's overlap with the neuropil by applying a grayscale encoding: the more percent of an arborization overlap with a certain neuropil, the more saturated the line will appear. Small arborization partitions will therefore always be related to less saturated projection edges. Through the scaling between black and gray, large overlaps that are more likely to form a connection will rather catch the viewer's attention than smaller overlaps that blend in with the background. In a graph with a lot of arborizations this helps to direct the user's attention towards more probable signal flows. The grayscale encoding is pictured in Figure 5.6.

In the initial version of neuroMap projection edges connected directly to their respective arborization node within a neuropil. However, in combination with the overlap edges between arborization nodes, the interior of neuropil super nodes appeared cluttered and confusing. In favor of a clean and visually appealing image, projection edges were set to terminate at the border of the respective neuropil super node.

Like in Yu's drawing, the end point of a projection edge can convey the terminal type. However, since the database does not yet actually include synaptic information, all arrow tips are uniformly represented by a white diamond shape as placeholder for the actual terminal information. Nevertheless, neuroMap is built with synaptic terminals in mind. So the appearance of the graph can be adapted as soon as the necessary information is available.

## Overlap Edges

The overlap of two arborizations is depicted as a connecting edge between them (Figure 5.6). The overlap amount can range from 100% to 0.01% and is encoded as grayscale value between black and light gray and as transparency between zero and 90%. An overlap of 100% results in a solid black line, while an overlap of 10% will be rendered in a highly transparent light gray. The scaling between black and gray will direct the



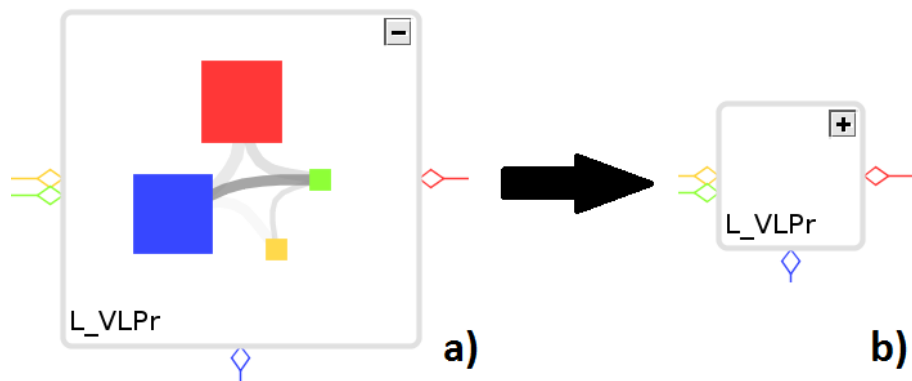
**Figure 5.6:** The encoding of different types of overlaps and their actual anatomical appearance. The *arborization-neuropil overlap* is encoded in the size of the arborization nodes and in the projection edge's shade of gray. The *arborization-arborization overlap* is encoded in the connecting edge between two arborization nodes.

viewer's attention towards bigger overlaps that are more likely to form a connection according to Peters' rule. The amount of an overlap is bidirectional, as the overlapping volume holds a certain percentage of each arborization's total volume. For a more streamlined view and less visual clutter, only the larger of both overlaps is directly encoded in the graph since it is a better indicator for the plausability of a connection.

As an overlap between two arborizations can lie within multiple neuropils, each of these neuropils holds a certain percentage of the arborizations' total overlap volume. In *Simple View* the overlap is not split up over neuropils and is therefore represented by a single edge. In *Standard View* the distribution of the overlap across neuropils is encoded in the thickness of an overlap edge. A thick line indicates that a big percentage of the overlap lies in a certain neuropil, while a less significant portion of the total overlap is indicated by a thin line. This makes it possible to quickly spot the areas of an overlap where a connection is more likely to occur. By focusing on the bigger of both overlaps and using grayscale, transparency and line thickness, the attention of the user is directed towards relevant overlaps, regardless of the view mode, while still keeping the context to the whole range of overlap information.

## Neuropils

The visual design of neuropil super nodes is kept simple to distract from their content as little as possible. The nodes are represented as rounded rectangles with white background and a colored frame. A label with the abbreviation of the neuropil name is displayed in the bottom left corner and a state icon in the top right corner of each node. The full name of a neuropil can be obtained from its tooltip window. A neuropil node has two states, opened and closed. When opened, the neuropil node's size scales automatically to accommodate the size of its content. In closed state the node's content is hidden and its size is set to a value that depends on the number of contained arborizations. The values for the size of closed neuropil nodes was chosen so that they will always be smaller than in opened state in order to take less space on the canvas. With this, users can hide unwanted details if they are not interested in the contents of a certain neuropil, while neuropil size and incoming projection edges still unobtrusively encode information about its content. Neuropils in both states can be seen in Figure 5.7. Depending on the selected layout, the position of a neuropil can convey information about its location in the brain. Neuropils that do not overlap with the displayed data are omitted from the visualization.



**Figure 5.7:** Neuropil node containing the same data in opened state (a), and closed state (b) with adapted node size.

## Node Color

The color of all nodes is derived from their corresponding workspace items which in turn are colored according to the selected color scheme and can be recolored individually. The neuron based color scheme for example gives all items that belong to the same neuron a uniform color. If no cell body objects were used for graph creation, the cell body node receives the same color as the first arborization to which it is connected. If a

neuropil is not loaded in BrainGazer's workspace, its frame is painted in light gray by default. Color itself does not convey any information and is only used to visually link associated elements and distinguish non related elements.

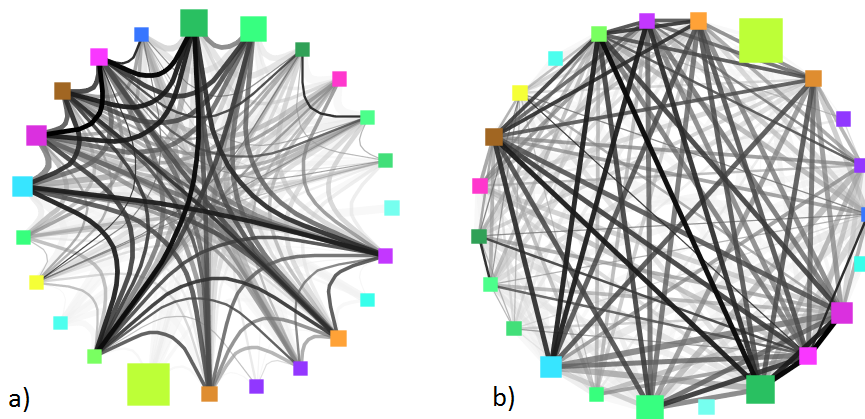
## 5.4 Layouts

### Subgraph Layout

Visual clutter in neuropil group nodes has been a challenge, as neuropils should stay as compact as possible while their content quickly increases when additional arborizations are included in the graph. The content of a neuropil node, is therefore laid out with a *circular layouter* which ensures compact node placement. Since the resulting layout always forms a circle, this results in a more uniform look for all neuropil nodes.

Still, when there were many overlaps within a neuropil the overlap edges would quickly occlude each other due to the circular positioning of arborization nodes and it would become difficult to single out a specific overlap. To solve this occlusion problem, a quadcurve edge router was deployed with bends in the middle of the circle for each edge. The resulting curved edges occlude each other much less than the original straight edges (Figure 5.8).

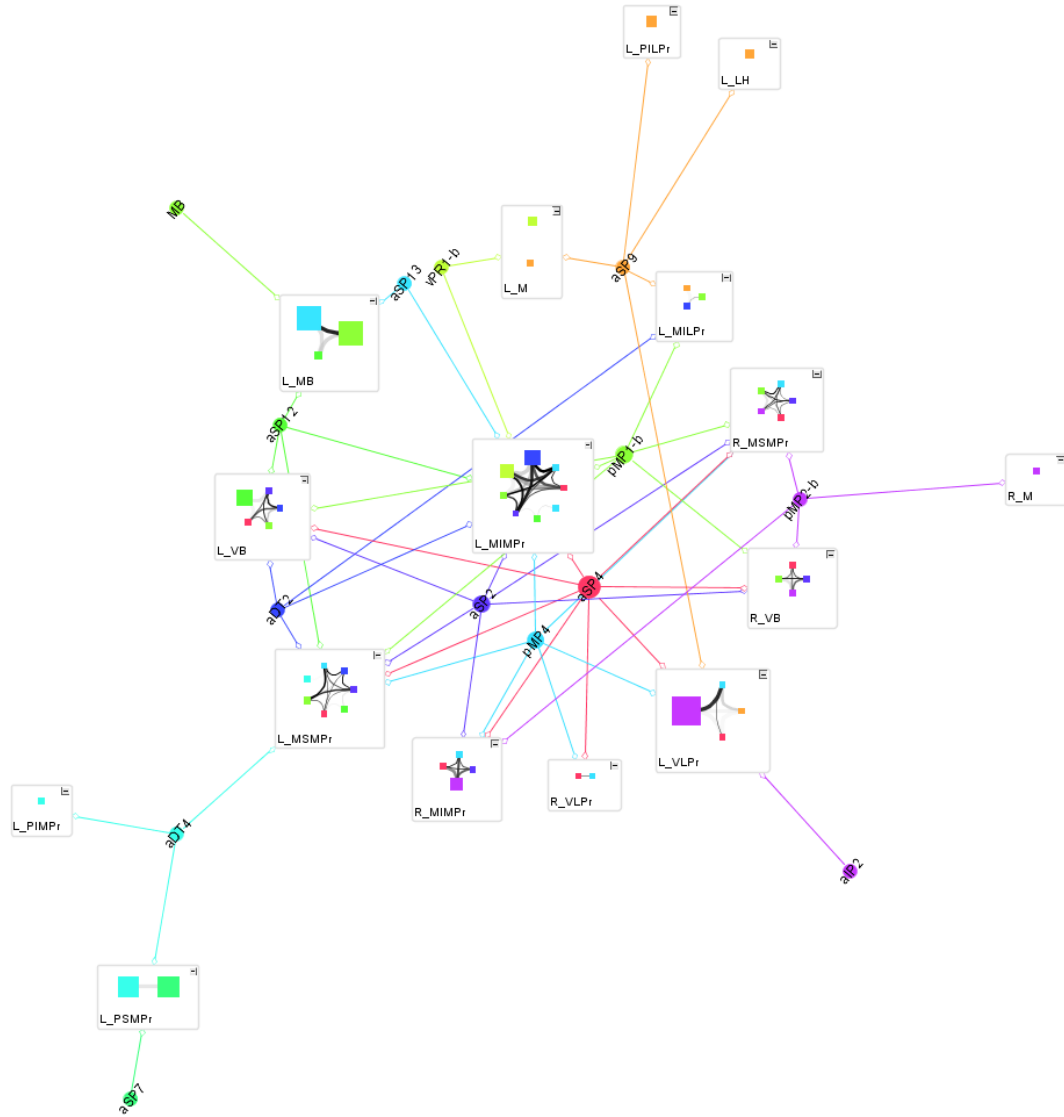
Additionally, to avoid occlusion of important overlaps, the overlap edges in each neuropil are sorted by their overlap amount. The additional use of transparency makes it easier to trace the path of partially occluded edges.



**Figure 5.8:** Comparison between circular arborization layout within neuropils with (a) and without (b) bends. By bending the edges towards the center of the circle, the layout appears more ordered than with the crisscross look without bends. Edges still occlude each other but more towards the center of the circle.

## Graph Layout

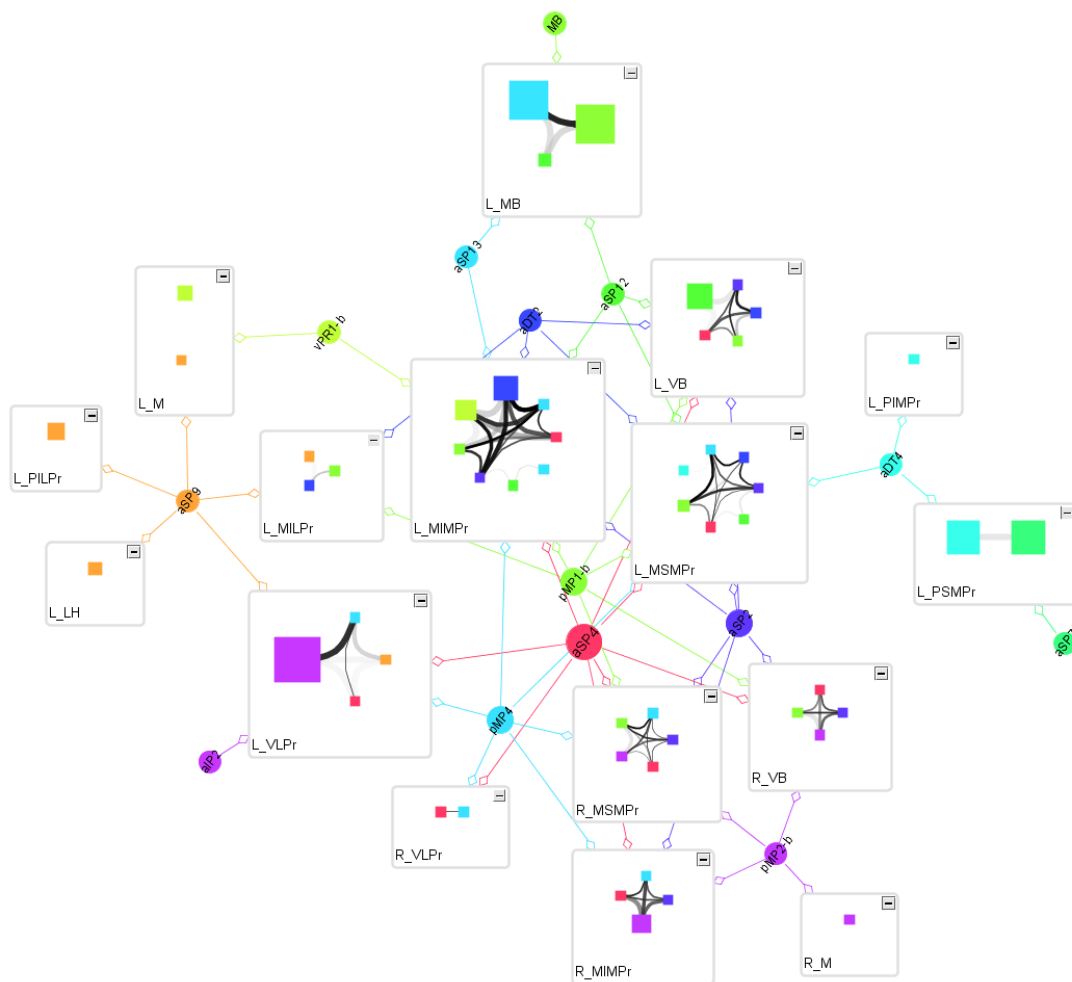
neuroMap offers five different layout modes for positioning a circuit graph's nodes and edges on the canvas: circular, organic, orthogonal, hierarchic and anatomical.



**Figure 5.9:** neuroMap's circular layout assigns biconnected elements to the same circular partition.

## Circular Layout

The circular layouter produces layouts that emphasize group and tree structures within a network. It creates node partitions by analyzing the connectivity structure of the network, and arranges the partitions as separate circles. Each partition represents a biconnected component of the graph. Nodes that belong to more than one biconnected component are assigned exclusively to one partition. Concerning the semantics of the data, this layout visually separates entities that have little or no interaction with each other, as can be seen in Figure 5.9. In this example the main partition circle consists only of interconnected neuropils from the brain’s left hemisphere while the remaining smaller partitions are part of other brain regions.



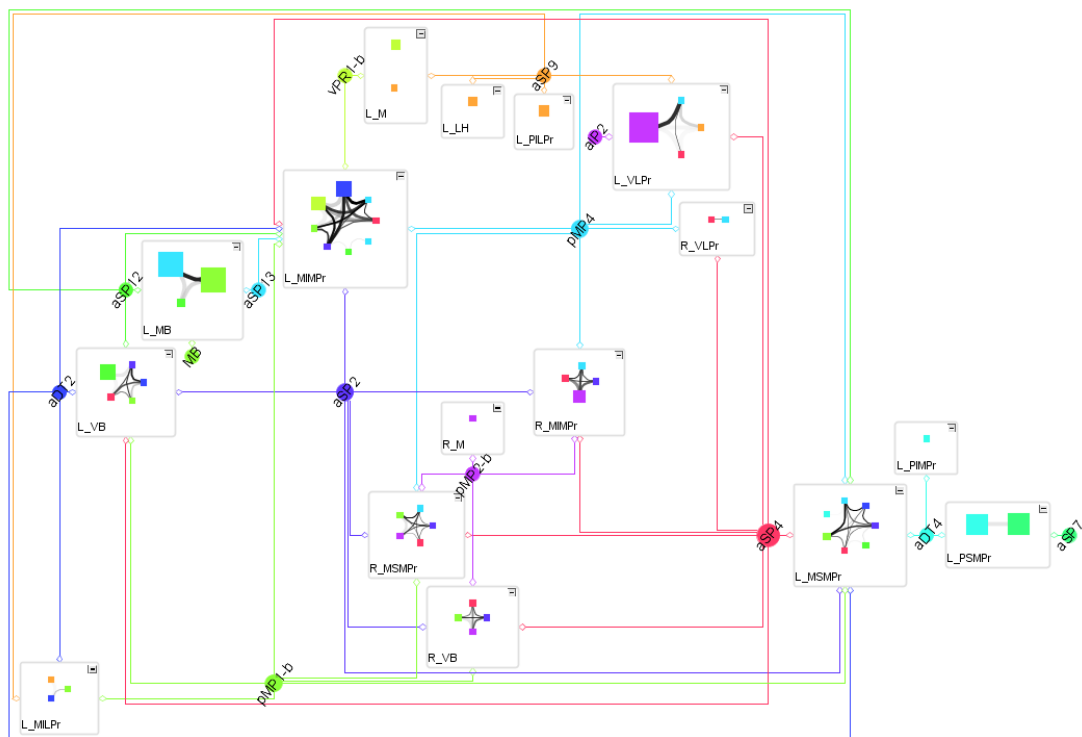
**Figure 5.10:** neuroMap’s organic layout is a force based layout approach.

## Organic Layout

The organic layouter is based on the force-directed layout paradigm. Resulting layouts often expose the inherent symmetric and clustered structure of a graph, show a well-balanced distribution of nodes and have few edge crossings. Similar to the circular layouter, tightly interconnected elements tend to form clusters while components with fewer connections are more isolated, as can be seen in Figure 5.10. Due to its heuristic optimization the organic layout produces slightly different results each time it is executed.

## Orthogonal Layout

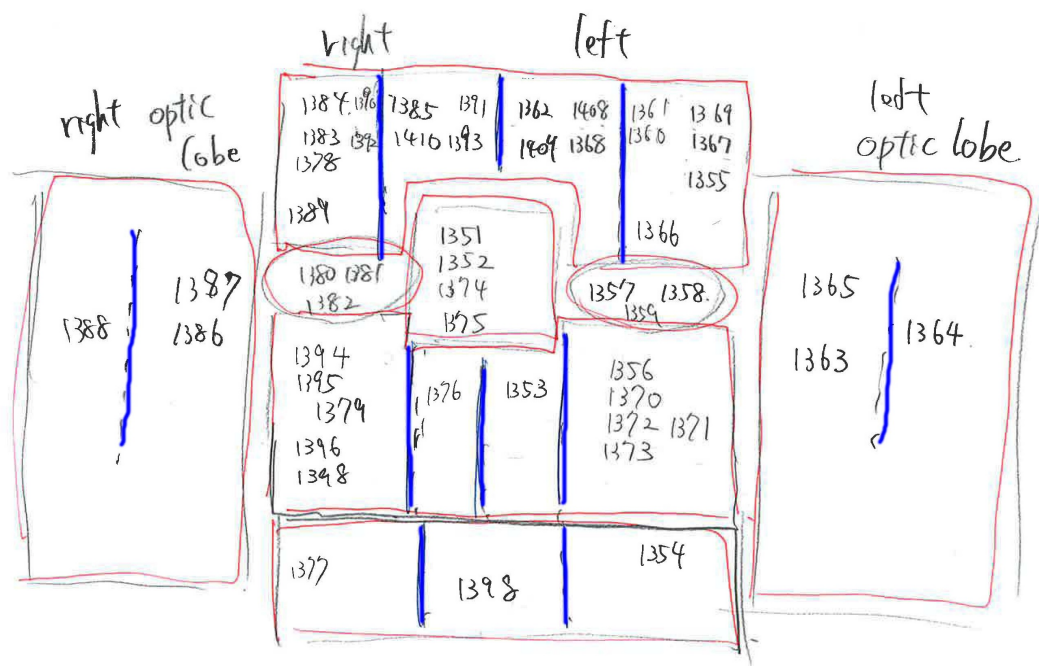
The orthogonal layout algorithm is based on the topology-shape-metrics approach. It is well suited for medium-sized sparse graphs and produces compact drawings that have a circuit diagram look through their orthogonally routed edges (Figure 5.11). However, because the orthogonal layouter strives to produce planar graphs, the distance from one node to another one does not convey any intrinsic information, like in the circular or organic layout.



**Figure 5.11:** neuroMap’s orthogonal layout tries to produce planar graphs.



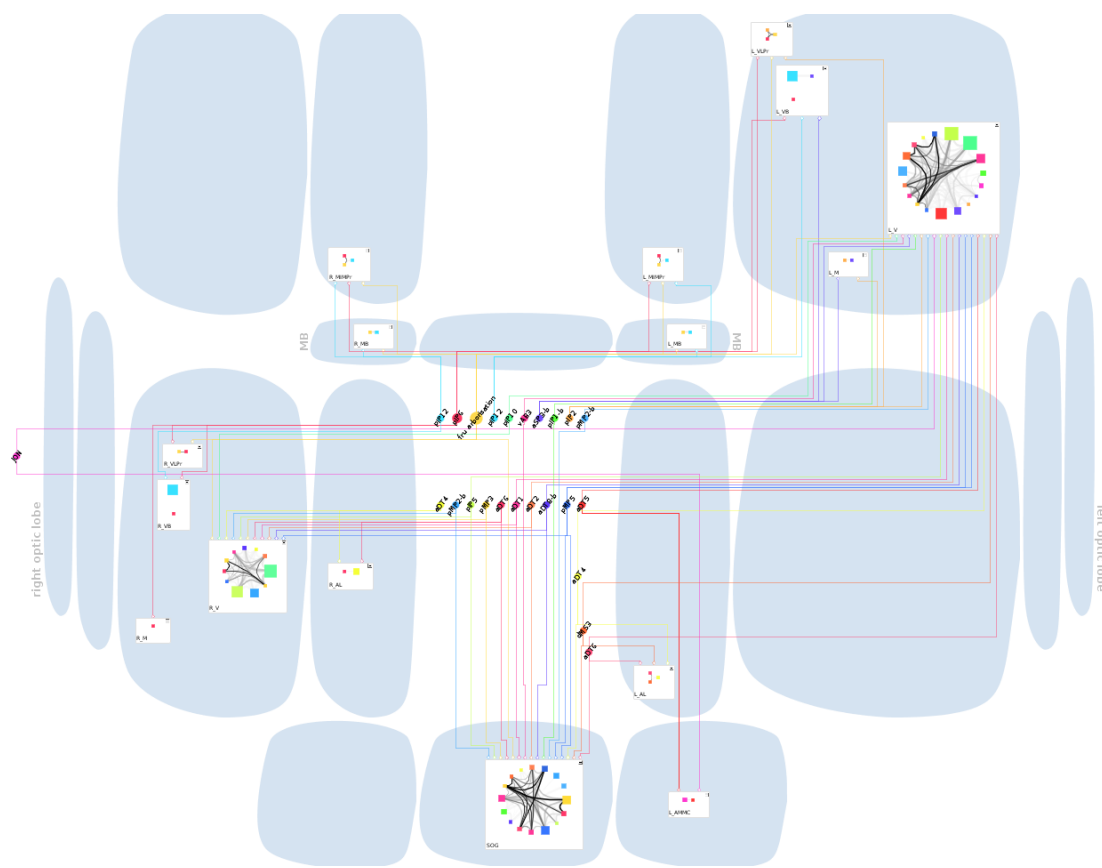




**Figure 5.13:** The sketch of brain compartments that served as blueprint for the anatomical layout.

of the graph. This is not the case in the other available layouts since node positions can change with each new configuration of the graph. The anatomical layout is only applicable to graphs with neuropils, since arborizations sometimes overlap with multiple neuropils in different brain regions. Arborization nodes can therefore not always be assigned to a single brain region.

A compartment in neuroMap is represented by a blue area that surrounds the associated neuropils. Some compartments like the left and right optical lobe are labeled to facilitate the users's orientation. The left and right brain halves are switched to match the scientists accustomed view on the data. If the number or size of the displayed neuropils within a compartment increases, the compartments scale automatically to accommodate their content. All cell body nodes except those of sensory afferent neurons are placed in the center of the graph. This has two advantages: on the one hand additional enlargement of compartments is avoided, on the other hand it results in a much clearer view of the signal flow as all projection edges originate from the center of the graph. Although there is no way to distinguish sensory afferent neurons from other neurons in the database, neuroMap flags some known sensory afferent neurons internally. Like in Yu's graph these cell body nodes are then placed separately outside the graph, to the left of the assembled brain compartments. This placement visually suggests the information flow from outside into the brain and distinguishes these neurons from non-sensory af-



**Figure 5.14:** neuroMap’s anatomical layout emulates an abstract view of the fly’s brain.

ferent ones. The underlying algorithm of the anatomical layout is a hierarchic layouter that helps to highlight the main direction of the flow within a directed graph. To reduce the amount of edge clutter in this layout, all projection edges of a neuron are bundled at the cell body node and drawn on top of each other instead of parallel to each other, until they split to connect to their respective neuropil nodes, much like in Yu’s graph. This results in an overall cleaner view.



# CHAPTER 6

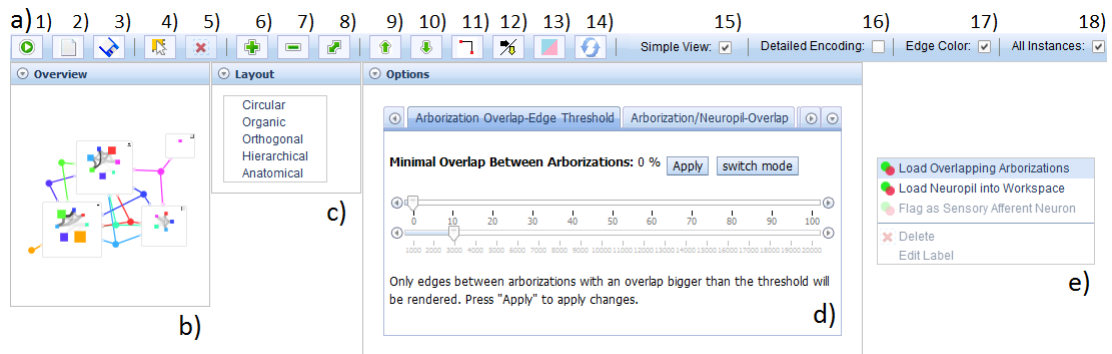
## Interaction

In this chapter neuroMap's user interface and the various interactions that it enables are discussed. neuroMap's design is aimed to make the user's tasks of graph creation, manipulation and exploration as easy and intuitive as possible in consideration of the semantics of the data. The design is therefore central to the aspect of visualizing and exploring potential connectivity between neurons (e.g., arborizations and their overlaps).

Section 6.1 introduces the elements of the interface, section 6.2 describes how graphs can be created in neuroMap, section 6.3 then explains how data and the visible structure of an existing graph can be manipulated, and section 6.4 describes how the structure of a graph and its underlying data can be explored.


### 6.1 Interface Overview

The neuroMap user interface consists of a toolbar, three retractable windows on the canvas, and context menus that popup upon right click on certain graph elements (Figure 6.1). The three retractable windows contain an overview of the entire graph, the selectable layout modes, arborization filter options and general settings. The overview helps the user to keep the orientation by displaying the entire graph together with a rectangle that indicates the zoomed-in area. The options menu consists of five switchable tabs. The first two contain sliders to adjust the arborization-arborization and the arborization-neuropil overlap thresholds as well as buttons to toggle between absolute and relative threshold. The remaining tabs let the user adjust the zoom factor, hit-test sensitivity and graph export format. The context menus are for editing node labels and deleting graph elements, and also offer node type specific functionality that will be elaborated later in this chapter.



**Figure 6.1:** The neuroMap user interface: a) toolbar, b) overview, c) layouts, d) filters and options, and e) context menu. The toolbar: 1) generate, 2) clear, 3) export, 4) toggle Edit Mode, 5) delete, 6)7) zoom in/out, 8) fit to screen, 9)10) open/close all neuropils, 11) reroute, 12) toggle layout orientation, 13) toggle sex filtering, 14) re-layout, 15) toggle Simple View, 16) toggle detailed encoding, 17) toggle edge encoding, 18) show all instances.

## 6.2 Graph Creation


To generate a graph, users have two options: the whole content of the workspace can be directly imported into neuroMap by clicking the *generate button* . If users want to create a graph from just a subset of the workspace's content, they can select the desired items with the mouse and drag and drop them onto neuroMap's canvas. By dragging and dropping additional items, an existing graph will be extended.

Graph generation is arborization centric, since potential connections between neurons are of main interest to the neuroscientists. This means, that the arborization forms the basic building block of a graph. Depending on what type of item is used as input for graph generation, the associated arborizations will be queried and used as building blocks. The basic building block, as generated from a single arborization, will consist of the cell body node, the arborization node(s) and all neuropil nodes in which the arborizations reside. When more than one arborization is loaded, the overlaps between all of them are calculated and visualized as overlap-edges. When a neuropil is used as input for graph creation, all arborizations that reside within the specified neuropil will be used as building blocks. This means that also all cell bodies and other overlapping neuropils of the contained arborizations will be loaded and displayed. When a cell body is used as input, the graph is constructed from all arborizations belonging to the cell body's neuron.

## 6.3 Graph Manipulation

### Filtering

As the content of the database steadily increases, there will be more and more objects that overlap each other. Using a single neuropil that overlaps with lots of arborizations as input would already result in a graph of substantial size. It is therefore necessary to offer filtering mechanisms for decreasing the amount of less relevant information in the view. To accommodate this, *filters for both types of overlaps* in the data were implemented: arborization-neuropil overlaps and arborization-arborization overlaps (Figure 6.1 d)). By increasing the thresholds of these filters, arborization nodes and overlap edges with smaller values than the specified thresholds will be omitted from the visualization. Each threshold can be adjusted in two different modes, either by overlap percentage or by overlap volume (measured in cubic microns). Both modes of threshold adjustment have been implemented because the arborizations in the fruit fly's brain can differ drastically in size (see section 5.3). The percentage threshold allows relative filtering of object to object relations, independent of the actual volume. The volumetric threshold allows absolute filtering of arborization partitions and overlaps.

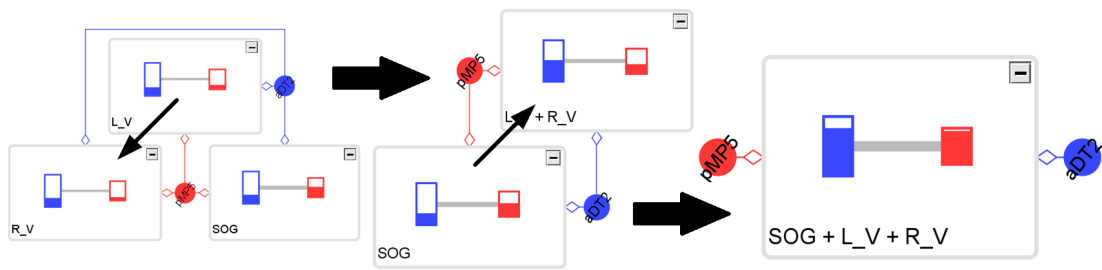
Filtering of female or male arborizations or deactivating sex-specific filtering is also possible (through the *toggle sex filtering button* ). This lets users for example quickly switch between the view on the female and male brain without the need to assemble a new graph from scratch.

Deactivating the „*Show All Instances*“ (Figure 6.1 18)) checkbox will filter all secondary instances from a graph. *Primary instances* are the designated representative entities if multiple entries of the same object exist in the database. This feature allows the quick comparison of related instances without the need of having to manually add and remove them from the visualization.

### Simplification

The complexity of the graph can be reduced by closing single neuropil nodes and hiding their contents with a click on the state icon. neuroMap offers the user also the possibility to open or close all neuropil nodes simultaneously. As mentioned in the visual encoding section, the relative size of a closed neuropil node still gives information about its number of overlapping arborizations. So even a graph with only closed neuropil nodes shows, which neurons could be connected in which neuropil, although not to what extent.

Another feature for simplifying an existing graph is the *merging of neuropil nodes*. The feature was requested since some groups of neuropils form functional units. If scientists are more interested in analyzing the potential connections within such a func-




**Figure 6.2:** Neuropil merging: arborization nodes and overlap edges are combined in the merged neuropil.


tional unit, rather than within multiple single neuropils, the merging feature enables them to do so.


Merging cannot only be useful for building functional units from single neuropils, though. If a partition of a graph does not necessarily need to be displayed in full detail, the part of the circuit can be simplified with the merging feature.

By dragging and dropping one neuropil node on another in Edit Mode, both are merged into a single new one that combines arborization nodes and overlap-edges from the original nodes. If an arborization or overlap was present in both nodes, the visual attributes are also combined, e.g., the thickness of the edges, the size or filling of the nodes. Since a neuropil also stores information about arborizations that are not displayed because their overlaps fall below the filter threshold, a merging operation can yield additional arborization nodes, that were not present in the original neuropil nodes, when their combined values rise over the specified threshold. Figure 6.2 shows the result of a merging operation.

## Layout Manipulation

Basic operations for changing the visual appearance of a graph, are the application of different layouts or changing the color of elements in BrainGazer’s workspace. The layouts can be selected directly from the „*Layout*“ dropdown window (Figure 6.1 c)). The layout orientation of anatomical and hierarchical layout can be changed from top-to-bottom to left-to-right with the *toggle layout orientation button* . In some scenarios this results in a better usage of screen space or a visually more appealing graph.

neuroMap’s internal list of sensory afferent neurons can be extended by selecting  **Flag as Sensory Afferent Neuron** from the right-click context menu on a cell body node. These cell bodies are then placed outside of the graph in the anatomical layout to indicate external neural stimuli, like described in section 5.4.

The *reroute button*  gives the user the option to additionally reroute all projection edges orthogonally, regardless of the selected layout. In contrast to the selectable layout



algorithms, this will only affect edges and not node positions. Orthogonal edge-routing produces some interesting results, in combination with the organic layout for example, since a circuit look is achieved while nodes are still clustered by the force directed algorithm (as opposed to the orthogonal layout where node positions are chosen to achieve minimal edge crossings). Another application for this button would be the rerouting of edges after the manipulation of the graph in Edit Mode.

neuroMap's *Edit Mode* provides several ways to fine tune the appearance of the graph manually, e.g., for presentation purposes. The user can alter the size and position of nodes and edit the text of node labels.

*Deletion* of graph elements is also only possible in Edit Mode and is context sensitive, i.e., is differently handled based on the object's type. When the user deletes an arborization node, all associated nodes that the arborization is composed of will be deleted. If the removal of this arborization leaves the cell body node without child nodes, the cell body will be removed as well. If a neuropil node is left empty by the removal of an arborization node, it is also deleted. Single arborization nodes can be removed by deleting their associated projection edge. If the user deletes a cell body node, all its associated arborizations are removed. If a neuropil node is deleted, only the node and its content will be removed from the graph. Since the workflow is arborization centric, it is assumed that neuropils will be removed by the user for lack of interest in the specific neuropil rather than the intent to delete all arborizations that overlap with the neuropil.

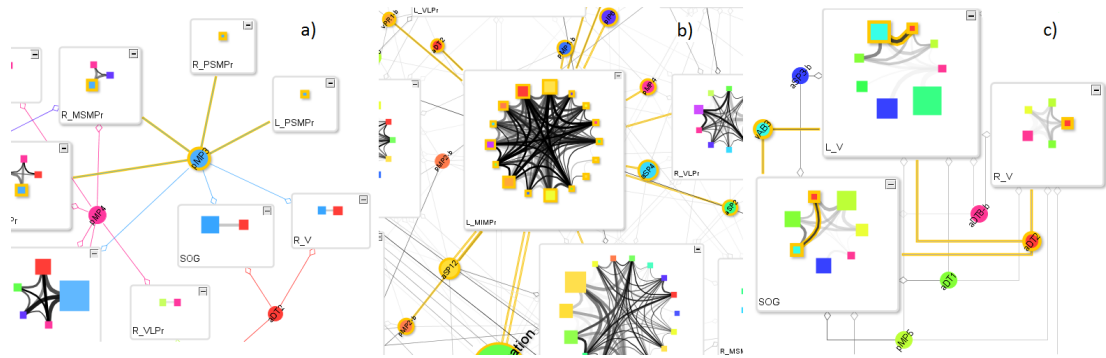
## Exporting the Graph

When the user is satisfied with the graph's structure and content, he or she can *export the graph* to a selected format. Currently GraphML, GIF, JPG, PNG, BMP and SVG formats are provided.

## 6.4 Exploration

### Highlighting

To emphasize relationships between elements that could be difficult to grasp in a large graph, a *context-sensitive highlighting* feature was implemented. A highlighted cell body or arborization node is rendered with a thick orange frame and casts a shadow on the canvas that gives it the impression of being elevated from its background. Both, overlap edges and projection edges, are rendered with thick orange borders when highlighted. If a neuropil node or one of its contained elements is highlighted, it will cast a shadow on the canvas and change its fill color to orange in closed state or highlight its content in opened state.



**Figure 6.3:** Context sensitive highlighting of graph elements: a) double-clicking an arborization node highlights all partitions of an arborization, b) double-clicking a neuropil node highlights all connected cell bodies, c) double-clicking an overlap edge highlights all partitions of this overlap as well as the involved arborization partitions.

Depending on the origin of the highlighting request, different relationships are accentuated. A double-click on an *arborization* node will highlight all nodes that represent the specific arborization, as well as the neuron’s cell body node and all associated projection edges (Figure 6.3 a)). In a large graph with many neuropils, this gives users a clear view of the partitioning of a single arborization and lets them grasp at a single glance where exactly these partitions lie (regardless of neuropil node states). Similarly, double-clicking on a *cell body* node will give the user an overview of all the neuron’s arborizations and their partitions over the graph. When a *neuropil* node is selected for highlighting, directly connected projection edges and cell body nodes as well as the neuropil itself or its content (depending on state) are highlighted (Figure 6.3 b)). This emphasizes, which neurons could be potentially connected through the selected neuropil. When a *projection* edge is selected, only the cell body at its source and the arborization node or neuropil at its target are highlighted. Highlighting an *overlap* edge will show the user, in which other neuropils this overlap exists, and will also highlight both arborizations in each neuropil, regardless if there is an overlap between them (Figure 6.3 c)). The initial implementation of overlap-edge-highlighting only highlighted arborization nodes when they were actually connected by an overlap edge. Since the neuroscientists were also interested in knowing where the two participating arborizations of an overlap edge do not connect, this information was also included. Through this object-type dependent highlighting, users can issue consecutive highlighting requests to explore the graph’s connectivity.


## Linking

BrainGazer's *linked view feature* displays the selection state of objects, synchronized over workspace and volume renderer. By including neuroMap into the list of linked views, an additional application-wide way of highlighting graph elements is provided, that further strengthens the integration of neuroMap into BrainGazer's infrastructure and opens new ways for application-wide interactions with the neuronal data. Linking works for multiple items simultaneously and in all directions, which means it can be initiated from each view.


Selecting an item in the workspace will highlight it in the graph by framing all corresponding arborization nodes with an orange border, and also in the volume renderer by giving the object a surrounding shimmer. This helps users to spot selected workspace items quickly and without effort even in large graphs. Figure 2.4) shows the linked selection state of an arborization across BrainGazer's views.

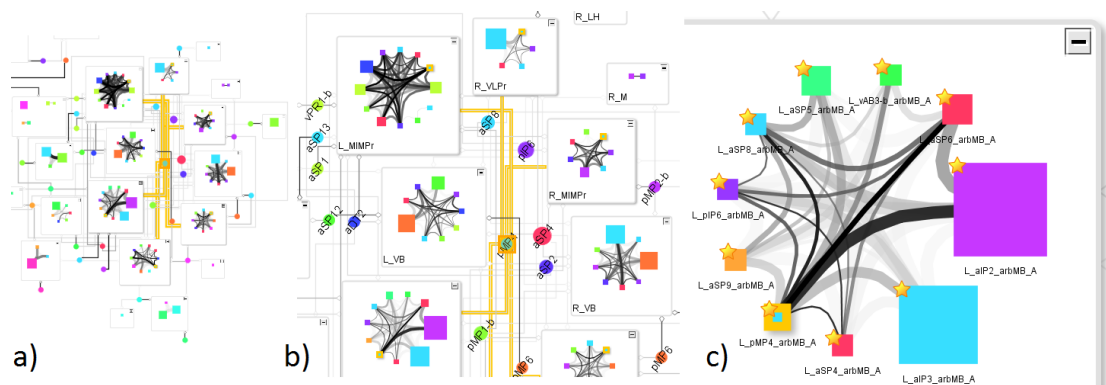
Similarly, when an item is selected in another view, it will be instantly focused and selected in BrainGazer's workspace. This speeds up the workflow by eliminating the need to skim the entire list of loaded workspace items for the corresponding graph element. By selecting an overlap edge, for example, the user can instantly review an anatomically exact version of the graph's abstract representation of this overlap and initiate a spatial query on the region if desired. The result can then be added to the workspace and included in neuroMap's 2D visualization. The linking therefore enables all views to supplement each other's features for exploration and analysis of the supplied data. neuroMap currently supports selection state linking of all graph items except projection edges, since these are generated with each arborization node and not from actual database information.

## Extending the Graph Structure

For exploring the database directly from within neuroMap, the right-click context menu offers the option to *query* for overlapping elements and load the results into the graph as well as into the workspace. The  **Load Overlapping Arborizations** command in the context menu has different effects for each type of node. When issuing the command from a cell body, all additional arborizations that belong to this neuron are loaded into the graph and workspace. When starting from a neuropil all arborizations that overlap with this neuropil are loaded. Similarly, when selecting the option from an arborization node, all arborizations that overlap with this arborization are loaded. The query on the database for loading the overlapping arborizations uses the threshold values from both types of overlap filters. For a neuropil only arborizations with an overlap higher than the arborization/neuropil overlap threshold will be loaded. For an arborization, only arborizations with an overlap higher than the overlap edge threshold will be loaded. The combination of the „Load Overlapping Arborizations“ command with both filters gives the

user more control over the expansion of the graph.

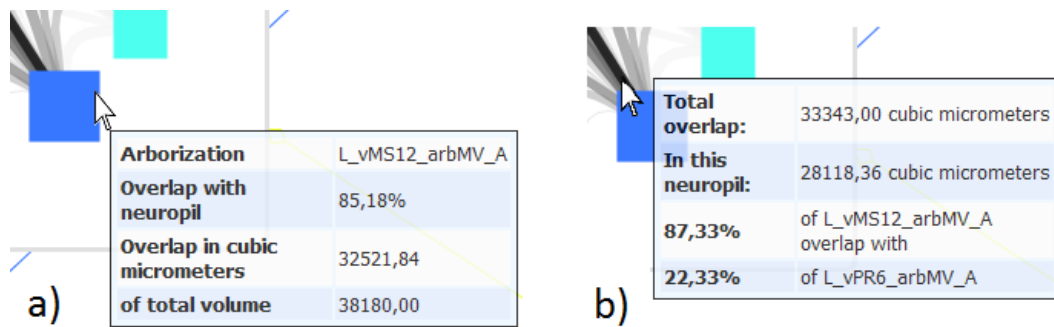
The self explanatory  **Load Neuropil into Workspace** command in the context menu saves the user the hassle to query for a neuropil that is already loaded in the graph. Adding a neuropil to the workspace enables the scientists to access its info page for direct comparison between database information and visual information; or to load the neuropil into the render view where it can be used as reference frame to the displayed items. Using this feature takes only two mouse-clicks, while querying for the desired neuropil in the database view takes considerably more time and effort.



**Figure 6.4:** Levels of Detail (LoD): a) lowest LoD with thick edges and omitted labels, b) second lowest LoD with enlarged labels, c) highest LoD with arborization labels and primary instance icons.

## Semantic Zooming

As a graph grows in size with the loading of additional elements, it becomes necessary to zoom out and gain an overview of its entire structure and to zoom in to focus on specific details. To supply the user with the most essential types of information for each zoom level, *zoom level dependent Levels of Detail (LoD)* were added to neuroMap. The lowest LoD is applied when the graph's zoom level is smaller than 0.3 (Figure 6.4 a)). Here the overall structure of the graph is more important than small details. All node labels are therefore hidden, since enlarging them to readable sizes would occlude the fine structures of the graph when zoomed out this far. Projection edges are enforced by multiplying their line width with a factor of 5. Since important overlap edges are significantly thicker than projection edges, their line width is only multiplied with a factor of 2. The same principle is applied to the highlighted versions of both types of edges. The selective enforcement of edges reassures their visibility at this zoom level. When grayscale encoding is selected, especially darker and therefore possibly more important edges are still clearly visible and distinguishable. The next level of detail



**Figure 6.5:** Tooltip windows of a) an arborization node, and b) an overlap edge.

is activated at a zoom level between 0.3 and 0.6 (Figure 6.4 b)). Here cell body and neuropil labels, as well as the region descriptions in anatomical layout are enlarged, as they would be hard to read in their original size. The graph is already zoomed in enough so that enlarged labels do not occlude other elements. Overlap edges are clearly visible at their original size and therefore not enforced anymore. The line width of projection edges is still multiplied by a factor of 2 to ensure their visibility. The next LoD is applied for a zoom level between 0.6 and 1.5. Here the graph is rendered in its original state. For the close-up view in zoom levels above 1.5 the names of arborizations are displayed beneath their nodes and a star icon that identifies primary instances is rendered in the top left node corner of the respective arborizations.

## Tooltips

Tooltips give additional information about each graph element that is not directly encoded within the visualization. They are accessed by hovering over the respective element with the mouse pointer. The use of tooltips avoids cluttering the visualization by encoding the most relevant information directly and offering details on demand. Figure 6.5 shows examples of an arborization tooltip and an overlap edge tooltip.

In the following the tooltip information available for each type of graph element is listed. Cell body: neuron name. Projection edge: names of corresponding arborization and neuropil. Neuropil: detailed name. Arborization: name, overlap percentage with neuropil, overlap volume in cubic microns, total arborization volume in cubic microns. Overlap edge: total overlap volume in cubic microns, overlap volume in the current neuropil, percentage of overlap volume of first and second arborization's volume.



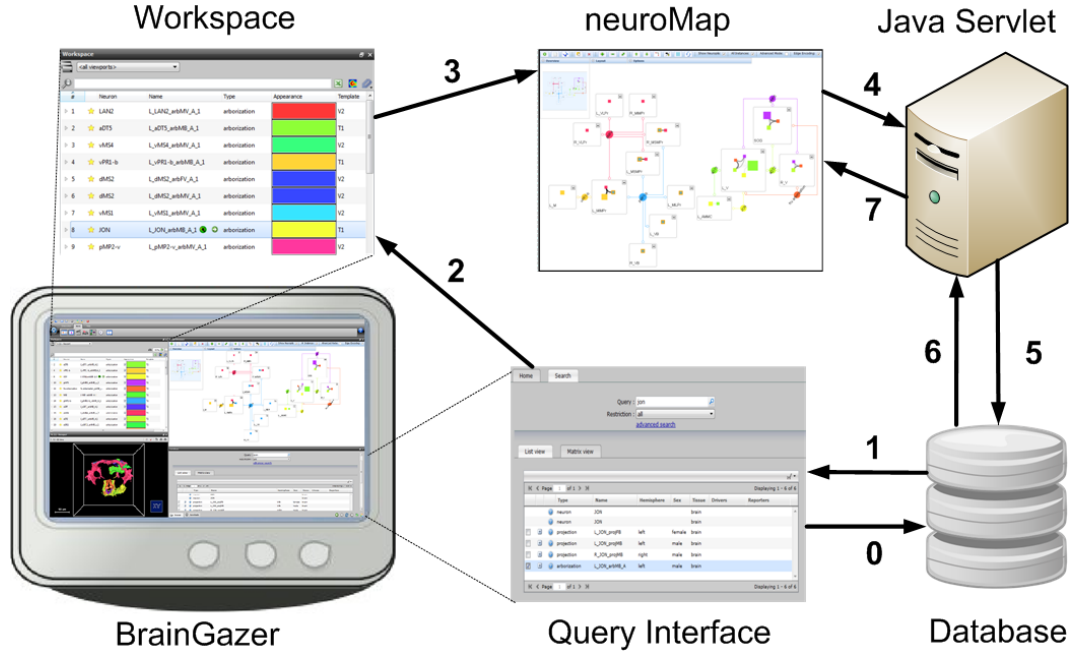
# Implementation

In this chapter the software architecture and relevant implementation details behind neuroMap are presented. Section 7.1 gives an overview of the system architecture. Section 7.2 introduces yFiles AJAX, the graph drawing library and client-server-framework that powers neuroMap, and section 7.3 presents the relevant aspects of BrainGazer's underlying database and how this data is used as input for neuroMap's visualization.

## 7.1 System Overview

neuroMap is built with the *yFiles AJAX* toolkit [77] as an additional web-based view in BrainGazer. yFiles AJAX offers a JavaScript Dojo widget for graph display and interaction on the client side and a Java Servlet interface to the *yFiles for Java* graph drawing library on the server side. The yFiles for Java graph drawing library handles creation, storage, manipulation, and rendering of the graph structure. Therefore most user interactions are sent to and handled on the server side. The server queries the database for information to construct the graph structure, modify the appearance of the graph's elements and annotate the elements with additional information. After the graph is completely generated, it is laid out with the currently selected layout algorithm. Finally the view of the graph is rendered and the tiles are sent to and displayed in the client browser.

BrainGazer is implemented in C++ , the user interface was built with the Qt toolkit. neuroMap runs in Qt's webkit browser and communicates with BrainGazer through Qt's signal and slot system. By putting neuroMap's interface in a browser window, it is possible to seamlessly include it in BrainGazer's infrastructure as an additional view, while still having the option of releasing a standalone web service (which is planned for future versions).



**Figure 7.1:** neuroMap's system overview in the scenario of graph creation within BrainGazer: 0) the user issues a query for a neuron of interest, 1) the database returns the results, 2) the user adds a selection of the results to BrainGazer's workspace, 3) a selection of the workspace items are used to generate a graph, 4) neuroMap sends the item IDs to the Java Server, 5) the server queries the database with the supplied IDs, 6) the returned information is used to construct the graph, 7) the graph is rendered and the tiles are sent to neuroMap's view.

Figure 7.1 depicts a system overview of neuroMap and BrainGazer, as well as the workflow for graph creation and the associated communication between system components.

## 7.2 yFiles AJAX

For neuroMap's implementation, it was decided to use a graph drawing library rather than to extend an existing graph drawing application, since neuroMap should be integrated into BrainGazer's environment as an additional view. From the various graph drawing libraries available on the market [17] [5] [25] [31], yFiles [77] is the one that was deemed the most suited for the task.

yFiles is a powerful graph drawing library with customizable layout algorithms that offer many features essential to neuroMap's visualization style. These features include



support for nested graph structures with proprietary internal layout (i.e., for neuropil nodes and their content), location constraints for graph elements (like in the anatomical layout), and flexible orthogonal edge routing capabilities that are important for neuroMap's „circuit diagram look“.

Also, the IMP intends to release a web-based version of some of BrainGazer's features and yFiles AJAX enables neuroMap to be used as both, an integrated part of BrainGazer as well as a standalone web-application.

yFiles can be extended with proprietary layout stages, as well as custom renderers. This way, the look of a graph can be tailored to the specific needs of an application. The standard routing in the internal layout of neuropil nodes was replaced by a custom layout stage to bend edges towards the center of the circular layout. The painters of each node- and edge-type were adapted to support neuroMap's look, LoD rendering, and highlighting in neuroMap. A custom background painter was added for rendering the partitions in the anatomical layout.

## Client

For the web client the yFiles Graph Editor was extended to implement neuroMap's user interface and client-side functionality. Typically a user interaction triggers a server request that contains the request name and the corresponding item IDs.

## Server

As the server holds the actual graph, it is able to map the client item IDs to their actual node or edge object representatives. Each object can hold arbitrary additional meta data like tooltip infos or the parent neuron ID for example. The object(s) along with its meta data are then used to perform the request, e.g., highlight related elements.

## 7.3 Database

For retrieving the information from the available database to generate the circuit diagram, two queries are issued: one for the general graph structure which consists of cell body, arborization, and neuropil nodes, as well as projection edges (*main query*), and the other one solely for overlap edges (*overlap query*). The input parameter of the main query is a list of arborization instance IDs. As already mentioned, an arborization as an object can have multiple instances, one primary, the rest secondary. The database requests query for the relations of specific instances (not objects). If the user supplies other neuronal entities (cell bodies or neuropils), for each type another query is issued before the general query, in order to retrieve the associated arborization instance IDs, as described in section 6.2. The result of the query contains 20 fields. The most important

aspects are depicted in table 7.1 with an explanation of how each field is used in the graph generation process.

The overlap query uses arborization instance IDs as well, and additionally the IDs of the neuropils that overlap with these arborizations, in order to find the overlap between each pair of arborizations per neuropil. The most important fields of this query with a description of the usage of the retrieved information is given in table 7.2.

The central point of both queries is the `neuropil_instanceoverlap` table from the database that contains the percentage of the overlap of each pair of instances per neuropil. The overlap of an instance with itself therefore yields, how much of the whole instance overlaps with the respective neuropil. Since in Simple View (without neuropils) only the direct overlap between arborizations is needed, only the `object_to_object` table is used in the overlap query.

When the user employs the context menu on a node to load its overlapping arborizations into graph space and workspace, a query for each type of neuronal object (arborization, cell body, or neuropil) first determines the instance IDs of the overlapping arborizations. These IDs are then used in the main query in order to add them to the graph. Then the overlaps between the newly loaded instances from the main query are calculated, as well as the overlaps between existing and newly added instances. The same procedure is applied when an existing graph is extended with items from the workspace.

Neuron Name	Visible in cell body node labels and tooltip. Used for filtering sensory afferent neurons.
Neuron ID	Used for bundling projection edges in hierarchic layouts. Identifies cell body nodes; links arborizations and cell bodies to a neuron.
Cell Body ID	Used for color lookup from workspace items and selection state transfer for cell body nodes.
Neuropil Name	The full name of a neuropil. Only visible in the neuropil node tooltip window.
Neuropil Short Name	Visible on neuropil node labels and projection edge tooltips. Used for identifying and assigning neuropils to their respective partitions, and for layering VNC neuropils in anatomical layout.
Neuropil ID	Identifies neuropil nodes. Used for color lookup and selection state transfer.
Arborization Name	Displayed in arborization and projection edge tooltips and as node label in the highest LoD.
Arborization Instance ID	Identifies arborization partitions in combination with the respective neuropil ID. Used for color lookup and selection state transfer.
Model Instance ID	Denotes the ID of the primary arborization instance ID. Used for identifying the primary arborization instance.
Arborization-Neuropil Overlap	Describes how much percent of the arborization overlap with a neuropil. Used to calculate the arborization partition node size in Standard View, the node filling in detailed encoding, and the shade of gray of projection edges. Used to filter arborization partitions according to the threshold <i>percentage</i> .
Arborization Volume	Total volume of an arborization; used for node size scaling in Simple View and detailed encoding; visible in tooltip.
Overlap Volume	Calculated from Arborization-Neuropil Overlap and Arborization Volume. Displayed in arborization tooltips; used for filtering arborization partitions according to the <i>volume metric threshold</i> .
Arborization Sex	Used for discerning and filtering male and female arborizations.

**Table 7.1:** The most important fields of the main query result.

Arborization & Neuropil IDs	Used to identify both arborization partitions that are connected by an overlap edge.
Overlap Volume	The total overlap of both arborizations in voxels. Visible in the tooltip; used to calculate the bidirectional overlap percentage with each arborization volume; used for filtering overlap-edges according to the <i>volumetric threshold</i> .
Bidirectional Overlap	Describes, how many percent of one arborization overlap with the other one. The bigger of both overlaps is used to calculate the gray scale value and transparency of the edge, and for filtering overlap edges according to the threshold <i>percentage</i> .
Neuropil Overlap	Describes, how many percent of the total overlap volume reside within the respective neuropil. Used to encode the overlap edge thickness.

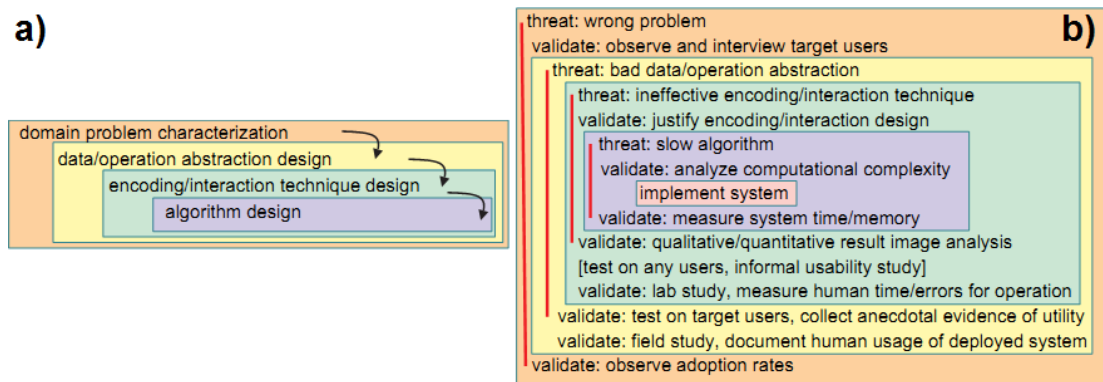
**Table 7.2:** The most important fields of the overlap query result.

## Evaluation and Results

This chapter covers the description of the evaluation process and its results. Section 8.1 introduces the applied method of evaluation, 8.2 presents the results of the discussions of neuroMap’s features with our clients. Section 8.3 gives an evaluative comparison of the visual features between Yu’s diagram and neuroMap. Section 8.4 points out how the open challenges in biomedical network visualization that were mentioned in section 3.3 were handled in neuroMap. The chapter is concluded in section 8.5 with a summary of the observed findings.

### 8.1 Evaluation Method

The evaluation of visualization is a major challenge at the moment, since traditional evaluation metrics such as task completion time are insufficient to quantify the utility of a visual analysis tool [55]. For the validation of neuroMap, the evaluation process was adapted to the *nested four layer model for visualization design and validation* that Munzner proposed [48]. A schematic of the four layers and their threats to validity is depicted in Figure 8.1. The four layers are domain problem characterization, data/operation abstraction design, encoding/interaction technique design and algorithm design. Each layer has its own threat to validity and two phases of evaluation: immediate and downstream. Immediate evaluation, as the name suggests, should happen in the design phase, before the implementation. Downstream evaluation can only be conducted after implementation, with the functional product. For a scientific paper usually only a subset of these layers is addressed. As the main focus of this thesis lies in neuroMap’s visual encoding and functionality in relation to the semantics of our clients’ data, the evaluation is based on the encoding/interaction technique design layer. The threat for this layer would be that the chosen design is not effective at communicating



**Figure 8.1:** The four nested layers in visualization creation (a), and their threats with immediate and downstream validation (b), as proposed by Munzner [48].

the desired abstraction. The immediate validation against this threat is a discussion and justification of the selected visual design. Discussion and justification have been given in chapter 5 (Visual Encoding). For the downstream evaluation that is done after the implementation phase, qualitative discussions of result images and interaction features with the neuroscientists were held.

To reassure that the development of neuroMap was always headed into the right direction, regular evaluation meetings with our clients were arranged. These discussions gave great insight into the scientists' workflow and helped better understand their mode of thought, which allowed the incremental improvement of neuroMap's features by adapting them to the scientists' specific needs. These talks were especially necessary since neuroscientists and computer scientists (or scientists of different fields in general) have their own vocabulary and thus their own way of communicating domain specific problems. The dialog between different groups can therefore be difficult and affected by misunderstandings, especially in the beginning of such collaborations. For example, the scientists were less interested in the sophisticatedly laid out graphs that are produced by the organic or orthogonal layouter, and more in the anatomical layout. The spatially relevant positions of neuropil nodes are more helpful to them than node clustering or minimized edge crossings. As a result of this, more time was spent on the improvement of the anatomical layout. The initial version only separated the brain into four crude regions; the current version features eighteen functional regions.

## 8.2 Discussion of Results

In addition to the regular evaluation meetings during the implementation phase, there were two in-depth evaluative discussions at the end of neuroMap's development cycle. These discussions were held with each scientist individually, one time with four, the

other time with three members of our collaborating group. A questionnaire served as checklist and guideline to structure the discussion. The participants, all male, consisted of a post-doc researcher, two PhD students and a master student. For three of the four participants the overlap of arborizations in the context of neuronal connectivity plays an important role in their research; these three were already actively using neuroMap. For the fourth one arborization overlaps will be of concern in future projects. He considered using neuroMap in later projects for comparing changes in overlap amount from different genetic experimental conditions.

The first session included basic questions about the subject's work in the context of neuron connectivity, a walkthrough of neuroMap's visual encoding and features, followed by a discussion of the most important ones, and a comparison between neuroMap and Yu's diagram. In the second session the new features that were partially added from the feedback of the first session were introduced and discussed.

## View Modes

The discussion of the views with and without neuropils showed no direct preference for one over the other. The prevailing opinion was that the neuropil-free graph in *Simple View* would be a good starting point for research since it shows overlaps directly without splitting them up, and therefore also offers a simpler, less cluttered view. The scaling of the node size according to volume makes the different arborizations easier distinguishable and therefore helps orientation in the absence of spatial structuring.

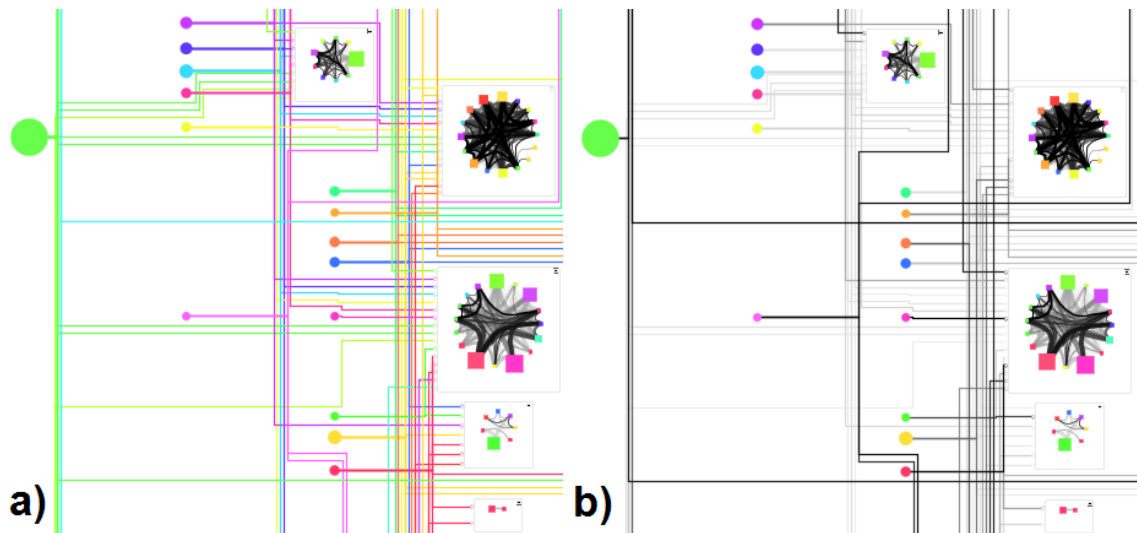
The *Standard View* on the other side offers more detailed information about overlap regions and a better structured view, and would therefore be more useful in scenarios where the overlap in certain functional brain regions is of concern. When comparing the standard against the detailed arborization node encoding, the detailed encoding was considered an improvement since it eliminates misinterpretation of the node size in Standard View. On the other side, for one of the participants the main interest lies in the actual overlaps between arborizations, so for him the arborization node size is less important in terms of exploring connectivity. He therefore favored the Standard View. The second one preferred the detailed encoding since he preferred visual clarity over simplicity. The third one could see himself using both, depending on the use case. Summarized, it was agreed that it is good to have both options available.

Since some neuropils are split arbitrarily and have no functional meaning, one scientist suggested the option to merge neuropils into single more meaningful group nodes. The idea was also well received by the other participants and has been implemented for the second evaluation session. The *merging* allows construction of combined neuropils like in Yu's graph which is useful since the full „neuropil resolution“ is not necessary in each research scenario. Since neuroMap encodes more information and is therefore visually more complex than Yu's graph, the merging is also a measure for increasing the visual simplicity, which according to one scientist is a necessary step when present-

ing findings to an audience that is not familiar with neuroMap’s graph design. For one scientist who focuses more on total arborization overlap than overlaps per neuropil, the merging feature also offers a great compromise between the Simple View and the view with neuropils. The full resolution is more important in well researched neuropils where the particular overlaps play a greater role.

## Edge Encoding

When considering the visual encoding of *overlap edges*, the scientists found the grayscale encoding of the overlap intuitive but found the additional encoding of the arborization-arborization-neuropil overlap as line thickness confusing. When asked if they would prefer the removal of the thickness encoding, they reckoned that the feature would be useful for more experienced users. One suggested a legend directly on the canvas to indicate the method of encoding for inexperienced users. Currently these details are only available on neuroMap’s help page.



**Figure 8.2:** Color (a) and gray scale (b) encoding of projection edges on the example of a large graph. The color encoding facilitates orientation within the network while the grayscale encoding emphasizes big arborization-neuropil overlaps.

In the direct comparison of both types of *projection edge* encoding (grayscale and cell body color), participants uniformly preferred the color encoding because it helps orientation especially in large graphs. Since the line has the same color as the cell body, it is easily traceable to its origin. So even when zoomed out, it is instantly recognizable, to which neuropils a cell body connects. If all lines in the graph are drawn with different shades of gray, tracing a single or a bundle of lines becomes difficult. The usage of the



cell body color also adds to the visual simplicity of the graph which seems to be the direction the scientists prefer. Also the grayscaling of projection edges was seen as redundant, since the conveyed information is already encoded in the arborization nodes. A comparison between both encodings can be seen in Figure 8.2. The option to encode the projection edge according to arborization color instead of cell body color was not well received since the differently colored lines can no longer be visually linked to their cell body. The use of the cell body color is therefore the most effective way to convey a graph element's affiliation to a neuron.

## Layouts

The layout that was uniformly considered the most intuitive for graphs with neuropils, was the *anatomical* layout, due to its localization of neuropils to partitions that resemble the anatomy of the brain. *Organic and circular* layout were also considered useful since they cluster highly interconnected elements. The *orthogonal* layout was seen as the least useful since the position of elements does not have much relevance. Concerning the *hierarchical* layout, it was mentioned that Yu's initial drawing was more along the lines of a hierarchical graph before he began to place certain elements at anatomically related positions.

For the *Simple View*, the circular layout was deemed the most efficient at conveying neuronal connectivity. It clusters highly connected graph elements, similarly to the organic layout, but at the same time is visually more pleasing through the arrangement of nodes in circles.

For the *anatomical* layout the scientists would have preferred anatomically correct positioning of cell bodies as well. Nevertheless, the necessary information is not available in the database yet, and not even all neurons have a segmented cell body object. One scientist suggested that the name of a neuron actually encodes the general position of the cell body and could therefore be used for positioning. pmP for example stands for posterial-medial-protocerebral. This new insight could be used for cell body placement in future versions of neuroMap. In the meanwhile the positioning of the cell bodies in the middle of the graph was considered as a good alternative, since they form a central point from which the flow of projection edges originates.

Concerning the edge flow direction (or the orientation of the layout) in the anatomical layout, one scientist stated that the top down layout seemed more familiar, another did not mind the direction since the projection edges do not convey any exact anatomical information. Additional naming of compartments for easier orientation was requested during the first evaluation session and has since been added to some compartments. For labeling the remaining compartments additional feedback from our clients will be needed. In general, even though the anatomical layout was the preferred one, the neuroscientists desired a look that is even more similar to the template of the brain in terms of partition placement and size. This would make the layout even more intuitive for un-

trained persons, which would be beneficial especially for presentation and publication purposes.

## Exploration

The *highlighting* of graph relations was generally well received because it facilitates orientation and exploration, especially in larger graphs. The highlighting of neural connectivity generated the most interest, but also different enhancement requests. One request was the option to select two or more cell bodies for highlighting their connections, which would be faster than searching explicitly through overlap edges. Another suggestion was to highlight all overlap edges that connect to a selected arborization instead of only highlighting the associated arborization partitions. A third suggestion was the indication of indirect connections between two neurons through a third neuron. Apparently there is room for expanding the highlighting feature that can be considered in future versions.

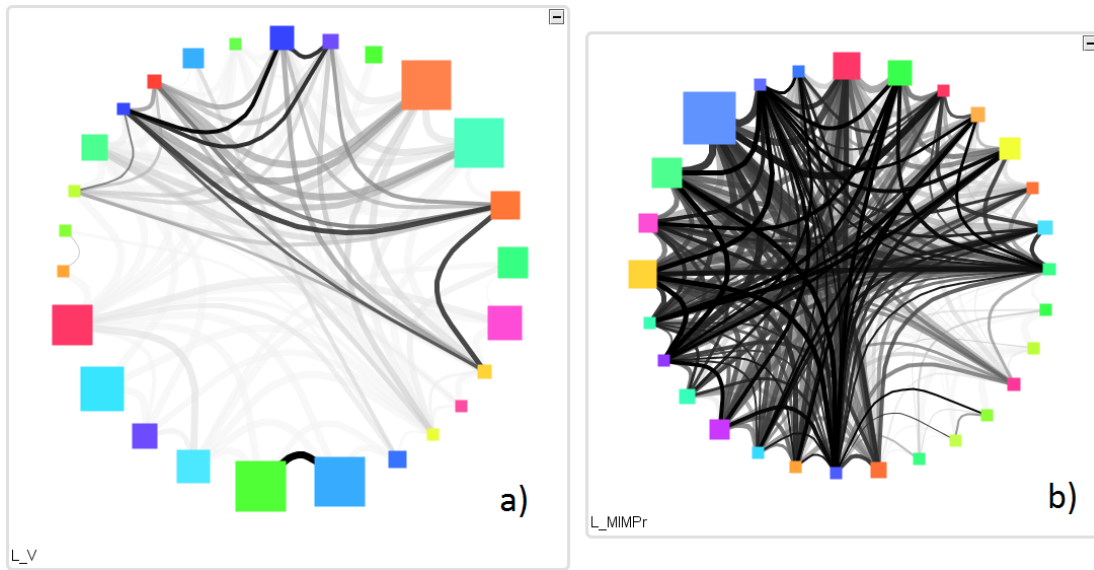
*Extending the graph* through loading additional elements using the context menu in combination with the two overlap filters was considered useful for finding potential connection candidates to neurons of interest. Although, one scientist admitted that neuroMap's linked view in combination with the 3D view's visual queries was his preferred way of searching for overlapping objects. The loading of neuropils or of the complete set of arborization of a cell body serves as a quick alternative to querying for these items in the database view.

neuroMap's *selection state highlighting* in BrainGazer's linked views was adopted seamlessly since the neuroscientists seemed to use the feature already efficiently during the second evaluation session to orient themselves within the different views.

*Filtering* was seen as a necessary means for controlling the minimal threshold for data that is added to a graph when generating a new structure or extending an existing one, thus avoiding visual clutter by omitting unwanted low resolution detail. The inclusion of both absolute and relative thresholds was deemed necessary due to the large fluctuations in arborization size. The absolute threshold is typically only used in special scenarios, as it bears the risk of filtering out entire arborizations due to their small size.

## Scalability

The typical use case in neuroMap involves only a handful of arborizations. Yu's drawing displayed all involved neurons of the paper's study, which amounted to about 80 of them. Nevertheless, a scenario where a user would want to look at all neurons in the database cannot be ruled out. To evaluate the scalability of neuroMap's graphs, a stress test with a graph containing all 213 arborization items that were available in the database at the time of writing was therefore conducted. This resulted in a graph with



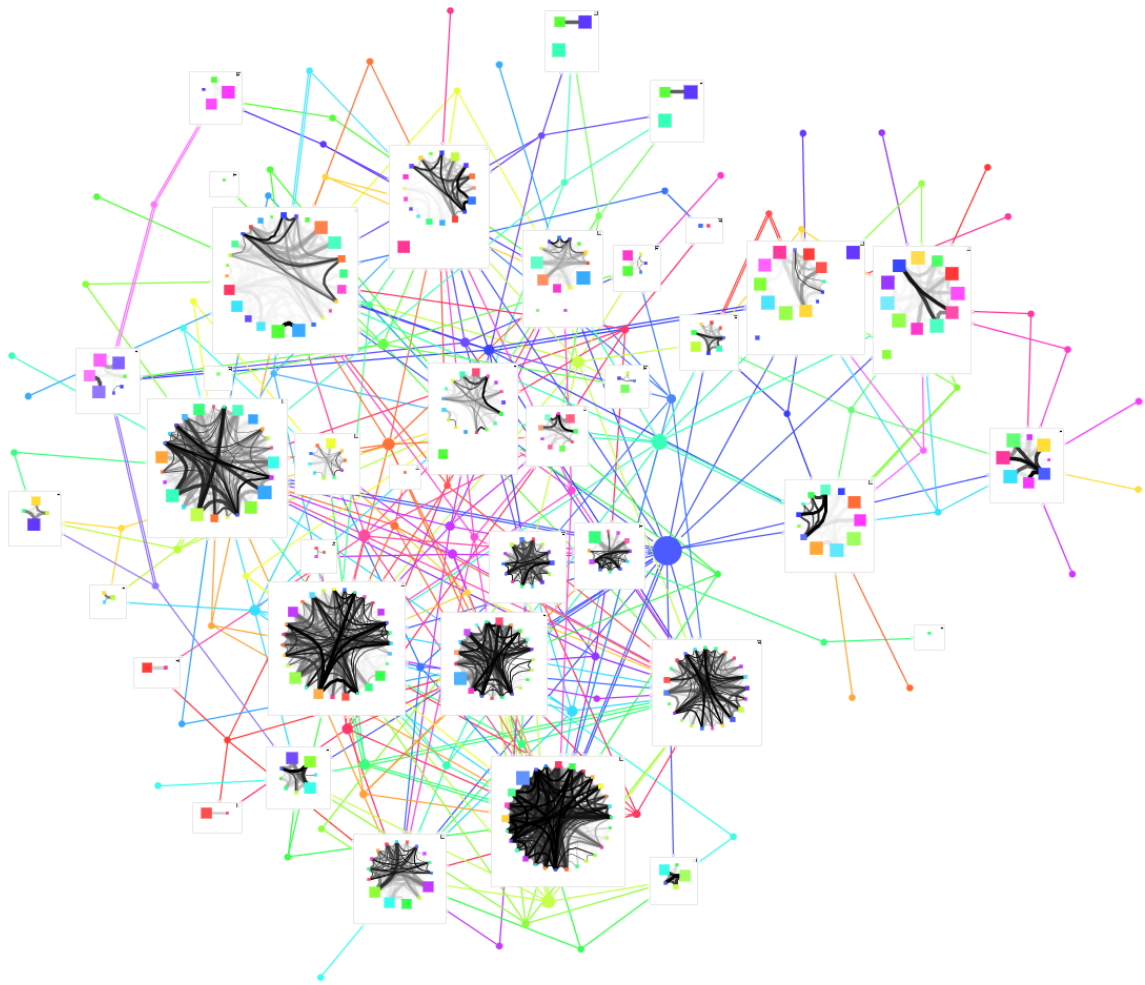
**Figure 8.3:** Two neuropils from a graph that contains over 200 arborizations: a) the overlap edges in the neuropil node are still distinguishable, b) a cluttered neuropil - it is hard to make out individual edges.

625 nodes and 3850 edges. The main concern in this scenario is that the circular layout of arborization partitions within neuropil nodes is so cluttered that overlap edges occlude each other to a degree that makes it hard to discern individual edges.

In our test case the overlaps were distributed over neuropils in a way that it was still possible to make out and select all individual edges when zooming in (Figure 8.3 a)). Only few neuropils overlap with so many arborizations simultaneously that edge occlusion was a problem (Figure 8.3 b)). Nodes themselves (of arborizations, cell bodies, or neuropils) are never occluded since they are drawn on top of edges.

The organic layout is the most suitable for this case since it uses the screen space so efficiently that even when looking at the entire graph, it is displayed at a zoom level where individual nodes and overlap edges are still discernible (Figure 8.4). The other layouts use too much screenspace to make the fine details of the graph recognizable when displaying this amount of data. This is especially the case for the orthogonal layout due to its planarization of projection edges. Nevertheless, the highlighting feature is a helpful tool for keeping track of individual relationships in these situations.

It can be concluded that neuroMap's weak point concerning scalability is the increased clutter within highly overlapped neuropil nodes. For the stress-test scenario of 213 arborizations the organic layout still performs well, i.e., the graph remains readable and connectivity information is extractable. Nevertheless, it can safely be assumed that with increasing size of the database's content, overlap edge occlusion will pose a chal-



**Figure 8.4:** A graph containing 213 arborizations, laid out with the organic layouter.

length that demands additional visualization methods, like fisheye lenses for example.

### 8.3 Comparison with Yu's Drawing

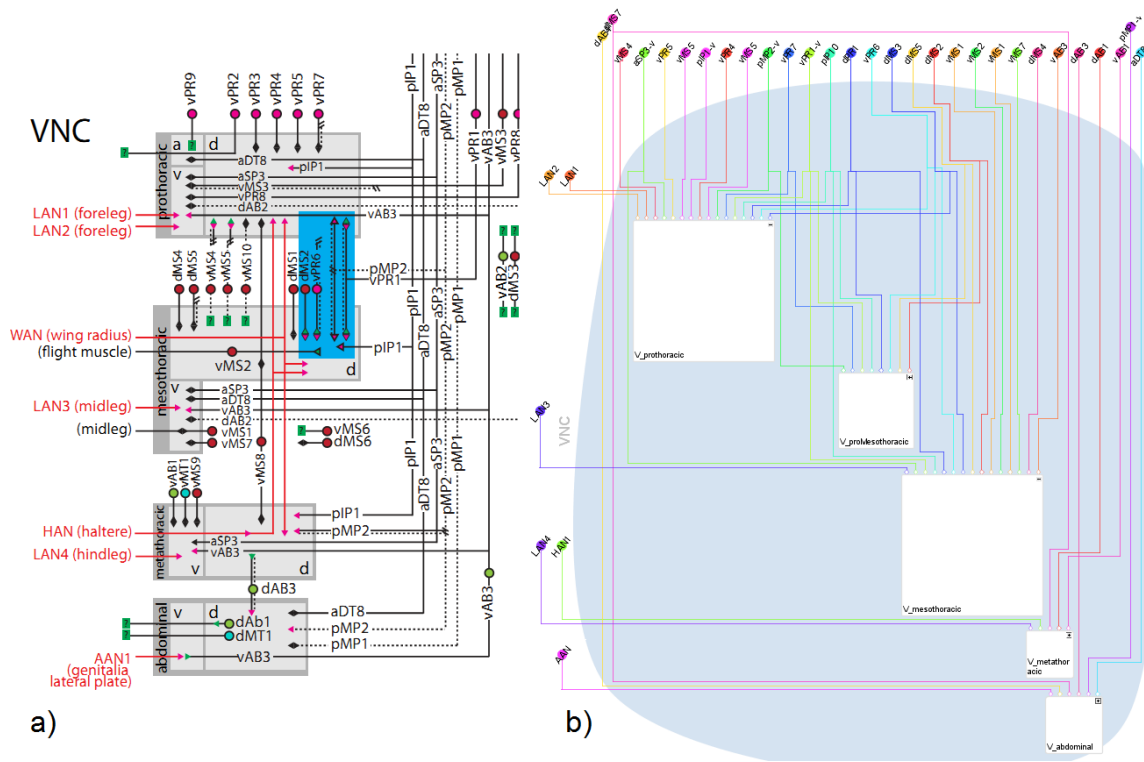
Compared to traditional 3D visualization, Yu's diagram has been regarded as an improved way of viewing the brain's wiring because it offers more information at a glance through its abstraction of the examined data. In a talk with the scientists it was explained that this form of representation is well suited for displaying bigger groups of neurons that are not very well researched yet. The abstraction therefore provides an overview that facilitates the formation of hypotheses about connections between the examined neurons.

Since Yu's diagram is laid out manually, it cannot be directly compared to the available layout algorithms in terms of graph drawing metrics, like efficient node placement or edge routing, but rather in terms of the conveyed information and intuitiveness of the resulting image. Yu's drawing is a mixed approach of anatomically motivated spatial constraints and arbitrary node/edge placement. The five *neuropils* of the VNC are placed at the bottom of the graph, layered in anatomically correct order. The placement of neuropils in the brain partitions favors convenient node placement over anatomical correctness. In neuroMap's anatomical layout the positions of neuropils are based on their actual anatomical positions which enables a more intuitive understanding of the image than in Yu's graph.

*Cell bodies* in Yu's graph are placed where convenient, i.e., have no relation to their actual positions. If the cell body placement in neuroMap would be set according to the name encoding like proposed in the previous section, the generated image would be even more meaningful in comparison.

The opposite is true for *projection edges*. In neuroMap the edges are placed and routed exclusively by the respective layout algorithm. In Yu's graph the projections are more accurate in terms of wiring, as the number of outgoing edges on a cell body node complies with the actual number of individual projections that originate from an anatomical cell body. The number of branchings on an edge complies with the actual number of branchings on the anatomical projection. An anatomical projection can for example go from cell body to an arborization and from the arborization again to another arborization. This way the projection edges in Yu's graph do not only visually connect a cell body to its attached arborizations, but also help connection prediction and support a more meaningful presentation of the wiring diagram. To automatically generate this sort of wiring in neuroMap, the necessary information is not available in the database at the time of writing. The actual position of projection edges in Yu's graph, does not convey any information though. The already mentioned absence of synaptic terminal representation in neuroMap is due to the absence of the necessary information in the database as well. To adapt at least the visual style of projection edges in Yu's graph, neuroMap bundles all outgoing edges of a cell body in the hierarchical and anatomical layouts until their first bend. Just one line leaves the node in each direction and splits up where it connects to a neuropil.

neuroMap's hierarchical layout is visually the most similar to Yu's graph because it places the cell bodies at the left side of the canvas vertically aligned, and projection edges flow from left to right. In the orthogonal layout, even though edges are rooted orthogonally, there is no main flow direction. The anatomical layout shares the abstract imitation of anatomical structure with Yu's graph. In the first evaluation session, an observed advantage of Yu's diagram over neuroMap was a better indication of sensory input through highlighting sensory afferent neurons and putting them on the left side of the diagram. The special node positioning has since been included into neuroMap's



**Figure 8.5:** A set of neurons innervating the VNC as pictured in Yu’s publication [78] (a), and a similar constellation of neurons generated with neuroMap (b).

anatomical layout. Figure 8.5 shows a side by side comparison between Yu’s graph and neuroMap.

The biggest advantages of neuroMap over Yu’s diagram that were brought up through the discussion are the inclusion of overlap information and therefore the indication of the probability of connections, as well as the automatic generation of the graph. In general, neuroMap offers more precision through arborization nodes and overlap edges, while Yu’s graph leaves much information unresolved but is also simpler and possibly clearer for the same reason. Through the node merging feature, the graphs in neuroMap now can be adjusted to a desired simplicity level in terms of arborization partitions. Additionally, closing neuropil nodes hides arborization nodes and overlap edges which brings neuroMap visually and in terms of simplicity very close to Yu’s graph, while still offering the possibility to show more detailed information where required. Filter options like the possibility to limit arborization partitions according to a specified threshold offer additional fine tuning of the displayed information.

In conclusion, the strengths of Yu’s graph lie in the clarity of its overview due to its simplicity. neuroMap can match Yu’s graph in terms of simplicity while delivering more

precision in multiple stages of abstraction that allow the tuning of the visualization to a desired level of detail. The areas where neuroMap cannot deliver the same information as Yu's drawing (synapse information and anatomically inspired projection edges) stem from the absence of the necessary information in the database.

## 8.4 Tackling the Challenges

Since network visualization in bioinformatics is a relatively new area, and bioinformatics itself a broad field, there are domain specific challenges that need special consideration when designing a visualization. The four open challenges in biomedical network visualization that were described in section 3.3 have been approached in the following ways in neuroMap:

The challenge of *incorporating spatial constraints* to represent biological structures has been handled in the anatomical layout by assigning neuropils and cell bodies to predetermined positions as described above. This way the graph complies with the mental image that the scientists have of the brain and therefore eases orientation.

The *visualization of multiple attributes* has been handled mainly in the visual encoding of nodes and edges but also information that is not encoded within the graph is accessible simultaneously through the selection state linking of BrainGazer's views. The linking makes it for example possible, to simultaneously view abstract and anatomically exact representations of the same items. The different visual encoding modes with their abstraction levels allow the addition or removal of visual attributes.

The challenge of *visualizing flows and paths* in biomedical network visualization lies in the handling of uncertainties and the accentuation of relevant structures within a network. Uncertainty, the probability of a connection, is handled in the grayscale, transparency and thickness encoding of overlap and projection edges. Relevant structures (e.g., more probable connections) are also accentuated through this encoding and can be further emphasized with the highlighting feature that takes the graph's intrinsic dependencies into consideration, as well.

For handling the challenge of *exploring hierarchical networks*, Albrecht et al. suggest a biologically meaningful visualization of the subsets of a network and their interrelations. Further they suggest to preserve the drawing conventions and the mental map of a graph as well as predefined or relative positions of graph elements after user interaction. neuroMap possesses a natural hierarchy through neuropil nodes and their content. Through considering the characteristics of the data, a biologically meaningful visualization of these substructures is achieved. The anatomical layout preserves the positions of graph elements by confining them to their assigned brain compartments. Therefore adding, removing, and merging elements does not affect the relative positions of the compartments, only their size. Neuropil merging and state change allow interactive simplification of the hierarchic structure. The LoD zooming enables the ex-

ploration of the hierarchic network at different zoom levels by adapting the detail for each type of graph element in the hierarchy without changing the layout or structure of the graph.

These solutions are of course tailored to the data that neuroMap operates on, and to the message that the generated images should convey, as there is no all-in-one solution for the presented challenges.

## 8.5 Performance

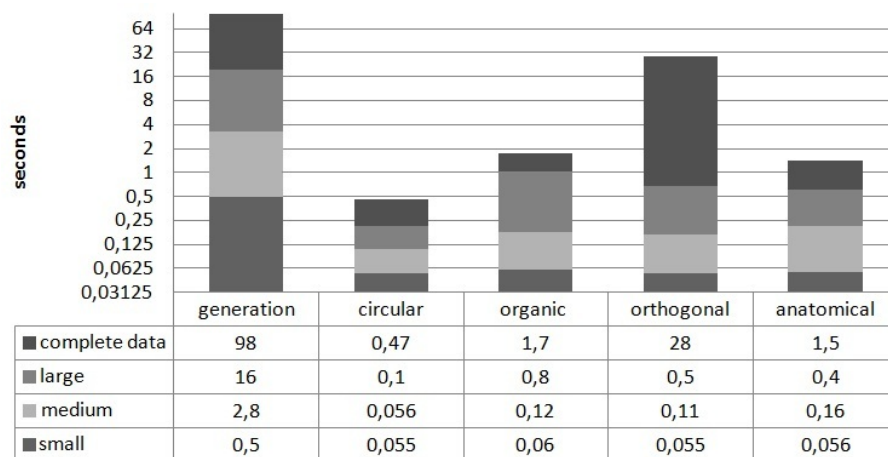
For the evaluation of neuroMap's performance a PC with an Intel Core2 Quad Q6600 processor at 2.4 GHz per core and 4 gigabytes of RAM running Windows 7 64 bit was used. The web application and the Java server were hosted on the same machine. The database was hosted on a PC with an Intel Pentium 4 processor at 3 GHz and 2 gigabytes of RAM running Ubuntu 11.04. Results may therefore vary not only with the configuration of the user's machine but also with the server, server load and internet connection. The graph in Figure 8.6 shows the performance of neuroMap with graphs of different sizes. A sample loading of overlapping arborizations by context menu (querying for, generating, and laying out of additional elements) took approximately 2 seconds and added 44 nodes and 281 edges to the graph and 17 arborizations to the workspace. The only actions that scale badly are graph generation and orthogonal layout. The application stays responsive for interactions like zooming and panning, highlighting, or neuropil node closing, even in large graphs.

## 8.6 Summary

The evaluation sessions with our clients helped clarify, which features and encodings in neuroMap are the most effective at communicating the intended abstraction, and how future versions could be even more improved. The discussions also indicated that the stated goal of providing means for easier hypothesis formation was met, as at the time of writing neuroMap has already been adapted by our clients, i.e., to make biased screenings. Biased screening means that certain candidates are dismissed from further observations because of their absence of overlap with the inspected neuron. Especially through neuropil merging and detailed encoding the scientists hope to tailor their graphs to a representation that lets them explore their interests by applying complexity where necessary and simplicity where possible.

Further, neuroMap was used for presentation purposes in meetings and will probably be used in publications in the near future. One scientist considered to use a graph in anatomical layout as a poster Figure, if the layout would be even more anatomically correct. Before, anatomical renderings of overlaps in combination with the respective





**Figure 8.6:** Performance times for a small graph of 7 arborizations, 27 nodes and 22 edges; a medium graph of 50 arborizations, 167 nodes and 251 edges; a large graph of 93 arborizations, 321 nodes and 803 edges; and a graph of all 213 arborizations in the database with 625 nodes and 3850 edges.

stained images were used to present findings. This takes a lot of time and work to produce but is possibly still more suitable in scenarios where just a handful of well known neurons are presented or discussed. In this case, a pairwise anatomical rendering would be used for example. neuroMap is especially suited for presenting findings in a circuit with lots of not so well known neurons, like in Yu's graph.

Concerning the recreation of Yu's visual style, the first approaches were naive, but improved with the progressive feedback of the scientists as far as the available information in the database allowed, until it offered the same visual simplicity but also extended detail were it was desired. Generally each meeting with our clients (not just the two evaluation sessions) brought new and interesting ideas in regard to the respective development stages of neuroMap. The high interest and enthusiasm towards neuroMap show that there is potential in its deployment that surely will be expanded through additional feedback once it is introduced to the whole group of concerned scientists.



## Conclusion and Future Work

### 9.1 Conclusion

In this thesis neuroMap was presented, a new approach for visualizing potential neuronal connections in the fruit fly's brain as an interactive circuit-style wiring-diagram that was integrated into BrainGazer as an additional view. neuroMap's creation was motivated by Yu's manually constructed wiring diagram [78]. The desirable aspects of this drawing are its two dimensional abstraction of complex volumetric data that enables a clear overview and highlights features that would be lost in a three dimensional representation. neuroMap's aim is to support hypothesis formation, data exploration, and rapid creation of graphs for presentation purposes by replicating the visual style and encoded information of Yu's drawing in an automatically generated graph. neuroMap was developed in collaboration with a group of neuroscientists and tailored to their specific needs in an iterative refinement process.

The fly brain is partitioned into different regions (neuropils). Each arborization of a neuron lies in one or more of these regions and overlaps with other arborizations where potential connections occur. A graph generated with neuroMap depicts the arborizations of neurons as child nodes within neuropil super nodes. By encoding the overlaps between arborizations as edges of varying grayscale value and size, all potential connections of the analyzed data are visualized in a clearly structured overview. Different levels of abstraction offer the user an adjustable compromise between simplicity and detail that allows the user to show more precise information where necessary. The decisions that went into neuroMap's visual and interaction design were discussed and justified. It was explained, how the open challenges in biomedical network visualization, relevant to the type and presentation of data in neuroMap, were solved. In qualitative discussions with our clients the implemented visual and interaction features were evaluated. The neuroscientists affirmed that the inclusion of neuroMap into BrainGazer facilitates

their research. It can therefore be concluded that neuroMap successfully fulfilled the stated design goal of automatically generating a circuit diagram that replicates the features of Yu's graph while further expanding them. This resulted in a visualization that eases connectivity hypothesis formation, and provides the generation of drawings for the presentation of said hypotheses and scientific findings.

## 9.2 Future Work

The discussion with our clients showed which areas of neuroMap need improvement in order to further increase its potential. Mainly the design of the anatomical layout should be adapted to look even more like the brain template. Also the placement of cell bodies according to their encoded position would result in a more recognizable and therefore intuitive look. The highlighting options for graph structures could be extended to emphasize also indirect connections between multiple selected neurons.

Additionally to the conclusions from our clients' feedback, there are many ideas for features that might broaden the usability of neuroMap or convey more useful information. It is intended to add graph analysis algorithms that give the user information about similarities and differences between two networks. Direct data exploration through neuroMap could be facilitated by allowing the user to directly show or hide graph objects in the 3D view, or load item info pages from within the context menu. Tooltip information could be improved by color coding text according to the item color and by adding preview renderings of the specific objects to the tooltip info-boxes. Also, as soon as the data is available, synaptic information and information about confirmed connections will be included in neuroMap.

# Bibliography

- [1] A.T. Adai, S.V. Date, S. Wieland, and E.M. Marcotte. LGL: Creating a Map of Protein Function with an Algorithm for Visualizing Very Large Biological Networks. *Journal of Molecular Biology*, 340(1):179–190, 2004.
- [2] C. Ahlberg, C. Williamson, and B. Shneiderman. Dynamic Queries for Information Exploration: An Implementation and Evaluation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, pages 619–626, New York, NY, USA, 1992. ACM.
- [3] M. Albrecht, A. Kerren, K. Klein, O. Kohlbacher, P. Mutzel, W. Paul, F. Schreiber, and M. Wybrow. On Open Problems in Biological Network Visualization. In *Graph Drawing*, volume 5849 of *Lecture Notes in Computer Science*, pages 256–267. Springer Berlin / Heidelberg, 2010.
- [4] J.D. Armstrong, K. Kaiser, A. Müller, K.F. Fischbach, N. Merchant, and N.J. Strausfeld. Flybrain, an On-Line Atlas and Database of the Drosophila Nervous System. *Neuron*, 15(1):17–20, 1995.
- [5] D. Auber, D. Archambault, R. Bourqui, A. Lambert, M. Mathiaut, P. Mary, M. Delest, J. Dubois, and G. Mélançon. The Tulip 3 Framework: A Scalable Software Library for Information Visualization Applications Based on Relational Data. Research Report RR-7860, INRIA, January 2012.
- [6] C.A.H Baker, M.S.T Carpendale, P. Prusinkiewicz, and M.G. Surette. GeneVis: Visualization Tools for Genetic Regulatory Network Dynamics. In *Proceedings of the Conference on Visualization '02*, VIS '02, pages 243–250, Washington, DC, USA, 2002. IEEE Computer Society.
- [7] A. Barsky, T. Munzner, J. Gardy, and R. Kincaid. Cerebral: Visualizing Multiple Experimental Conditions on a Graph with Biological Context. *IEEE Transactions on Visualization and Computer Graphics*, 14:1253–1260, 2008.
- [8] J. Bertin. *Semiology of Graphics: Diagrams, Networks, Maps*. University of Wisconsin Press, 1983.

- [9] G. Bezgin, A. Reid, D. Schubert, and R. Kötter. Matching Spatial with Ontological Brain Regions using Java Tools for Visualization, Database Access, and Integrated Data Analysis. *Neuroinformatics*, 7:7–22, 2009. 10.1007/s12021-008-9039-5.
- [10] N. Bhatla. An Interactive Visualization of the C. Elegans Neural Network . <http://wormweb.org/neuralnet#c=BAG&m=1>, June 2009. Accessed: 07/2012.
- [11] J. E. Blankenship and B. Houck. *Nervous System (Invertebrate)*. McGraw-Hill's AccessScience, 2012.
- [12] V. Braitenberg and A. Schüz. *Cortex: Statistics and Geometry of Neuronal Connectivity*, volume 249. Springer Berlin, 1998.
- [13] A.H. Brand and N. Perrimon. Targeted Gene Expression as a Means of Altering Cell Fates and Generating Dominant Phenotypes. *Development*, 118(2):401–415, 1993.
- [14] E. Bullmore and O. Sporns. Complex Brain Networks: Graph Theoretical Analysis of Structural and Functional Systems. *Nature Reviews Neuroscience*, 10(3):186–198, February 2009.
- [15] G. Burns, W.C. Cheng, R. Thompson, and L. Swanson. The NeuARt II System: A Viewing Tool for Neuroanatomical Data Based on Published Neuroanatomical Atlases. *BMC Bioinformatics*, 7:1–19, 2006. 10.1186/1471-2105-7-531.
- [16] A.S. Chiang, C.Y. Lin, C.C. Chuang, H.M. Chang, C.H. Hsieh, C.W. Yeh, C.T. Shih, J.J. Wu, G.T. Wang, and Y.C. Chen. Three-Dimensional Reconstruction of Brain-Wide Wiring Networks in Drosophila at Single-Cell Resolution. *Current Biology*, 21(1):1–11, 2011.
- [17] M. Chimani, C. Gutwenger, M. Jünger, K. Klein, P. Mutzel, and M. Schulz. The Open Graph Drawing Framework. In *15th International Symposium on Graph Drawing 2007*, pages 23–26, 2007.
- [18] W.S. Cleveland. *The Elements of Graphing Data*. Wadsworth Publ. Co., Belmont, CA, USA, 1985.
- [19] C. Collins and S. Carpendale. VisLink: Revealing Relationships Amongst Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1192–1199, nov.-dec. 2007.
- [20] S. Cook, C. Brittin, D. Hall, and S. Emmons. The Worm Wiring Project . <http://www.wormwiring.org/>, June 2012. Accessed: 07/2012.

- [21] E. Demir, O. Babur, U. Dogrusoz, A. Gursoy, G. Nisanci, R. Cetin-Atalay, and M. Ozturk. Patika: An Integrated Visual Environment for Collaborative Construction and Analysis of Cellular Pathways. *Bioinformatics*, 18(7):996–1003, 2002.
- [22] G. Di Battista, P. Eades, R. Tamassia, and I.G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 1998.
- [23] B.J. Dickson. Wired for Sex: The Neurobiology of Drosophila Mating Decisions. *Science*, 322(5903):904–909, 2008.
- [24] P. Eades. A Heuristic for Graph Drawing. *Congressus Numerantium*, 42:149–160, 1984.
- [25] J. Ellson, E. Gansner, L. Koutsofios, S. North, and G. Woodhull. Graphviz— Open Source Graph Drawing Tools. In *Graph Drawing*, volume 2265 of *Lecture Notes in Computer Science*, pages 483–484. Springer Berlin Heidelberg, 2002.
- [26] A. Frick, A. Ludwig, and H. Mehldau. A Fast Adaptive Layout Algorithm for Undirected Graphs (Extended Abstract and System Demonstration). In *Graph Drawing*, volume 894 of *Lecture Notes in Computer Science*, pages 388–403. Springer Berlin / Heidelberg, 1995.
- [27] T.M.J. Fruchterman and E.M. Reingold. Graph Drawing by Force-Directed Placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991.
- [28] D.C.Y. Fung, S.H. Hong, D. Koschutzki, F. Schreiber, and F. Xu. Visual Analysis of Overlapping Biological Networks. In *Information Visualisation, 2009 13th International Conference*, pages 337–342, july 2009.
- [29] N. Gehlenborg, S.I. O’Donoghue, N.S. Baliga, A. Goesmann, M.A. Hibbs, H. Kitanou, O. Kohlbacher, H. Neuweber, R. Schneider, D. Tenenbaum, and A.C. Gavin. Visualization of Omics Data for Systems Biology. *Nature methods*, 7(3 Suppl):56–68, March 2010.
- [30] A. Godiyal, J. Hoberock, M. Garland, and J. Hart. Rapid Multipole Graph Drawing on the GPU. In *Graph Drawing*, volume 5417 of *Lecture Notes in Computer Science*, pages 90–101. Springer Berlin / Heidelberg, 2009.
- [31] C. Gutwenger, M. Jünger, K. Klein, J. Kupke, S. Leipert, and P. Mutzel. A Diagramming Software for UML Class Diagrams. *Graph Drawing Software*, pages 257–278, 2004.
- [32] D. Hall, Z. Altun, and L. Herndon. Worm Image. <http://www.wormimage.org/>, 2002-2012. Accessed: 07/2012.

- [33] N. Henry, J.D. Fekete, and M.J. McGuffin. NodeTrix: A Hybrid Visualization of Social Networks. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1302–1309, nov.-dec. 2007.
- [34] D. Holten. Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, sept.-oct. 2006.
- [35] Z. Hu, J.H. Hung, Y. Wang, Y.C. Chang, C.L. Huang, M. Huyck, and C. DeLisi. VisANT 3.5: Multi-Scale Network Visualization, Analysis and Inference Based on the Gene Ontology. *Nucleic Acids Research*, 37(suppl 2):W115–W121, 2009.
- [36] A. Irimia, M.C. Chambers, C.M. Torgerson, and J.D. Van Horn. Circular Representation of Human Cortical Networks for Subject and Population-Level Connectomic Visualization. *NeuroImage*, 60(2):1340–1351, 2012.
- [37] R. Jianu, C. Demiralp, and D.H. Laidlaw. Exploring Brain Connectivity with Two-Dimensional Neural Maps. *IEEE Transactions on Visualization and Computer Graphics*, 18:978–987, 2012.
- [38] M. Kaiser. A Tutorial in Connectome Analysis: Topological and Spatial Features of Brain Networks. *NeuroImage*, 57(3):892–907, 2011. Special Issue: Educational Neuroscience.
- [39] K. Kojima, M. Nagasaki, and S. Miyano. An Efficient Biological Pathway Layout Algorithm Combining Grid-Layout and Spring Embedder for Complicated Cellular Location Information. *BMC Bioinformatics*, 11:1–15, 2010. 10.1186/1471-2105-11-335.
- [40] L. Krempel. Network Visualization. In *Handbook of Social Network Analysis*, chapter 37. Sage Publishing, 2009. Scott, J. and Carrington, P.J. (Eds.).
- [41] M. Krzywinski, J. Schein, Í. Birol, J. Connors, R. Gascoyne, D. Horsman, S.J. Jones, and M.A. Marra. Circos: An Information Aesthetic for Comparative Genomics. *Genome Research*, 19(9):1639–1645, 2009.
- [42] K. Li, L. Guo, C. Faraco, H. Zhu, D. and Chen, Y. Yuan, J. Lv, F. Deng, X. Jiang, T. Zhang, X. Hu, D. Zhang, and T. Miller, L.S. and Liu. Visual Analytics of Brain Networks. *NeuroImage*, 61(1):82–97, 2012.
- [43] C.Y. Lin, K.L. Tsai, S.C. Wang, C.H. Hsieh, H.M. Chang, and A.S. Chiang. The Neuron Navigator: Exploring the Information Pathway Through the Neural Maze. *Visualization Symposium, IEEE Pacific*, 0:35–42, 2011.



- [44] T.F.C. Mackay and R.R.H. Anholt. Of Flies and Man: *Drosophila* as a Model for Human Complex Traits. *Annu. Rev. Genomics Hum. Genet.*, 7:339–367, 2006.
- [45] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network Motifs: Simple Building Blocks of Complex Networks. *Science*, 298(5594):824–827, 2002.
- [46] N. Milyaev, D. Osumi-Sutherland, S. Reeve, N. Burton, R.A. Baldock, and J.D. Armstrong. The Virtual Fly Brain Browser and Query Interface. *Bioinformatics*, 28(3):411–415, 2012.
- [47] B. Mlecnik, M. Scheideler, H. Hackl, J. Hartler, F. Sanchez-Cabo, and Z. Trajanoski. PathwayExplorer: Web Service for Visualizing High-Throughput Expression Data on Biological Pathways. *Nucleic Acids Research*, 33(suppl 2):W633–W637, 2005.
- [48] T. Munzner. A Nested Model for Visualization Design and Validation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):921–928, nov.-dec. 2009.
- [49] H. Neuweger, M. Persicke, S.P. Albaum, T. Bekel, M. Dondrup, A.T. Hüser, J. Winnebold, J. Schneider, J. Kalinowski, and A. Goesmann. Visualizing Post Genomics Data-Sets on Customized Pathway Maps by ProMeTra - Aeration-Dependent Gene Expression and Metabolism of *Corynebacterium Glutamicum* as an Example. *BMC Systems Biology*, 3(1):82–96, 2009.
- [50] S. Oeltze, W. Freiler, R. Hillert, H. Doleisch, B. Preim, and W. Schubert. Interactive, Graph-Based Visual Analysis of High-Dimensional, Multi-Parameter Fluorescence Microscopy Data in Toponomics. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1882–1891, dec. 2011.
- [51] H. Ogata, S. Goto, K. Sato, W. Fujibuchi, H. Bono, and M. Kanehisa. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research*, 27(1):29–34, 1999.
- [52] S.R. Olsen and R.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.
- [53] G. Pavlopoulos, S. O’Donoghue, V. Satagopam, T. Soldatos, E. Pafilis, and R. Schneider. Arena3D: Visualization of Biological Networks in 3D. *BMC Systems Biology*, 2:1–7, 2008. 10.1186/1752-0509-2-104.
- [54] G. Pavlopoulos, A.L. Wegener, and R. Schneider. A Survey of Visualization Tools for Biological Network Analysis. *BioData Mining*, 1(1):1–12, 2008.

- [55] C. Plaisant, G. Grinstein, and J. Scholtz. Visual-Analytics Evaluation. *IEEE Computer Graphics and Applications*, pages 16–17, 2009.
- [56] M. Pohl, M. Schmitt, and S. Diehl. Comparing Readability of Graph Layouts Using Eyetracking and Task-Oriented Analysis. In *Proceedings of Computer Graphics International*, volume 7, Victoria, British Columbia, Canada, 2009.
- [57] W.A. Press, B.A. Olshausen, and D.C. Van Essen. A Graphical Anatomical Database of Neural Connectivity. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 356(1412):1147–1157, 2001.
- [58] H. Purchase. Which Aesthetic Has the Greatest Effect on Human Understanding? In *Graph Drawing*, volume 1353 of *Lecture Notes in Computer Science*, pages 248–261. Springer Berlin / Heidelberg, 1997.
- [59] D. Reinhard. *Graph Theory*. Springer Verlag, 2005.
- [60] T. Rohlfing and C.R. Maurer Jr. Nonrigid Image Registration in Shared-Memory Multiprocessor Environments with Application to Brains, Breasts, and Bees. *IEEE Transactions on Information Technology in Biomedicine*, 7(1):16–25, 2003.
- [61] M. Rubinov and O. Sporns. Complex Network Measures of Brain Connectivity: Uses and Interpretations. *NeuroImage*, 52(3):1059–1069, 2010. Computational Models of the Brain.
- [62] H.S. Seung. Neuroscience: Towards Functional Connectomics. *Nature*, 471(7337):170–172, 2011.
- [63] G.M.G. Shepherd, A. Stepanyants, I. Bureau, D. Chklovskii, and K. Svoboda. Geometric and Functional Organization of Cortical Circuits. *Nature neuroscience*, 8(6):782–790, 2005.
- [64] T. Shimada, K. Kato, A. Kamikouchi, and K. 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, Statphys-Taiwan-2004: Biologically Motivated Statistical Physics and Related Problems, 7th Taiwan International Symposium on Statistical Physics*, 350(1):144–149, 2005.
- [65] B. Shneiderman. Tree Visualization with Tree-Maps: 2-D Space-Filling Approach. *ACM Trans. Graph.*, 11(1):92–99, January 1992.
- [66] M.E. Smoot, K. Ono, J. Ruscheinski, P.L. Wang, and T. Ideker. Cytoscape 2.8: New Features for Data Integration and Network Visualization. *Bioinformatics*, 27(3):431–432, 2011.

- [67] D. Stalling, M. Westerhoff, and H.C. Hege. Amira: A Highly Interactive System for Visual Data Analysis. In *The Visualization Handbook*, pages 749–767. Elsevier, 2005.
- [68] K. Sugiyama, S. Tagawa, and M. Toda. Methods for Visual Understanding of Hierarchical System Structures. *IEEE Transactions on Systems, Man and Cybernetics*, 11(2):109–125, feb. 1981.
- [69] KiCad EDA Software Suite. <http://www.kicad-pcb.org/display/KICAD/About+KiCad>, December 2012. Accessed: 10/2012.
- [70] Cadence Design Systems. <http://www.cadence.com/us/Pages/default.aspx>, December 2012. Accessed: 10/2012.
- [71] T. Tekusova and T. Schreck. Visualizing Time-Dependent Data in Multivariate Hierarchic Plots - Design and Evaluation of an Economic Application. In *Information Visualisation, 2008. IV '08. 12th International Conference*, pages 143–150, july 2008.
- [72] C. Tominski, J. Abello, F. van Ham, and H. Schumann. Fisheye Tree Views and Lenses for Graph Visualization. In *Information Visualization, 2006. IV 2006. Tenth International Conference on*, pages 17–24, july 2006.
- [73] S. Tweedie, M. Ashburner, K. Falls, P. Leyland, P. McQuilton, S. Marygold, G. Millburn, D. Osumi-Sutherland, A. Schroeder, R. Seal, H. Zhang, and The FlyBase Consortium. FlyBase: Enhancing Drosophila Gene Ontology Annotations. *Nucleic Acids Research*, 37(suppl 1):D555–D559, 2009.
- [74] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J.J. van Wijk, J.D. Fekete, and D.W. Fellner. Visual Analysis of Large Graphs: State-of-the-Art and Future Research Challenges. *Computer Graphics Forum*, 30(6):1719–1749, 2011.
- [75] M. Ward, G. Grinstein, and D. Keim. *Interactive Data Visualization: Foundations, Techniques, and Applications*. A. K. Peters, Ltd., Natick, MA, USA, 2010.
- [76] I.R. Wickersham and E.H. Feinberg. New Technologies for Imaging Synaptic Partners. *Current Opinion in Neurobiology*, 22:121–127, 2012.
- [77] R. Wiese, M. Eiglsperger, and M. Kaufmann. yFiles - Visualization and Automatic Layout of Graphs. In *Graph Drawing Software, Mathematics and Visualization*, pages 173–191. Springer Berlin Heidelberg, 2004.
- [78] J.Y. Yu, M.I. Kanai, E. Demir, G.S.X.E. Jefferis, and B.J. Dickson. Cellular Organization of the Neural Circuit that Drives Drosophila Courtship Behavior. *Current Biology*, 20(18):1602–1614, 2010.

- [79] R. Yuste. Circuit Neuroscience: The Road Ahead. *Frontiers in Neuroscience*, 2(1):6–9, 2008.
- [80] S. Zhao, M.J. McGuffin, and M.H. Chignell. Elastic Hierarchies: Combining Treemaps and Node-Link Diagrams. In *Proceedings of the 2005 IEEE Symposium on Information Visualization*, INFOVIS '05, pages 57–64, Washington, DC, USA, 2005. IEEE Computer Society.