FAKULTÄT FÜR !NFORMATIK

Visual Queries in Neuronal Data Exploration

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Computergraphik/Digitale Bildverarbeitung

eingereicht von

Veronika Šoltészová
Matrikelnummer 0326311

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Betreuer: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Mitwirkung: Dipl.-Ing. Dr.techn. Stefan Bruckner

Wien, 13.06.2009

_____ (Unterschrift Verfasserin)

_____ (Unterschrift Betreuer)

Veronika Šoltészová

Ďurgalova 6
SK-83101 Bratislava

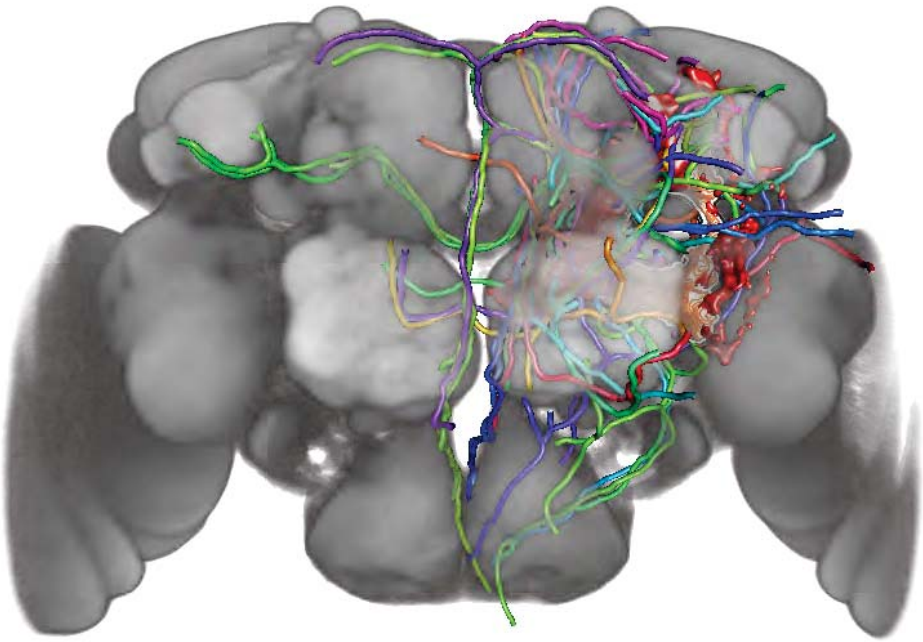
Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit einschließlich Tabellen, Karten und Abbildungen - die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen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

Veronika Šoltészová, Wien, 18. Dezember 2009

Veronika Šoltészová

Visual Queries in Neuronal Data Exploration

Master's Thesis



supervised by

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller

Dipl.-Ing. Dr.techn. Stefan Bruckner

Institute of Computer Graphics and Algorithms

Vienna University of Technology

Abstract

The major goal of neuroscientists' work is to explain specific behavior of living beings, especially humans. However, human behavioral traits are complex and difficult to comprehend. For this purpose, the researchers explore the anatomy and morphology of neuronal circuits of simpler species to identify their meaning and functionality. The fruit fly *Drosophila melanogaster* is a favorite organism in neurobiology research because it facilitates studies of complex systems on a simple model. For this purpose, large databases of neuronal structures acquired by microscopy scans were built and adapted for computer-aided exploration and visualization. Commodity products feature standard visualization techniques tailored for exploration of biological structures. However, orientation in large collections of structures still poses a problem. Traditional table-view database interfaces allow filtering of items and accessing known subsets of data, but do not support selection based on spatial relationships.

In this thesis, we address this problem in the following way. We describe a system which facilitates visual exploration of a large collection of neuroanatomical structures. We combined standard visualization techniques with a novel visual approach for exploration and queries. Our system provides three basic types of queries. *Path queries* use an intuitive sketching interface and give access to structures located in the proximity of the sketched path. *Object queries* select objects based on their mutual spatial distance. *Semantic queries* allow fast browsing using semantic relationships stored in the database. The system was designed in an interdisciplinary collaboration with domain experts, who affirmed that availability of such a system would be very useful for their research.

Kurzfassung

Das Hauptziel der Neurobiologen besteht darin, lebewesenspezifisches Verhalten, insbesondere das des Menschen, zu erforschen und zu erklären. Allerdings sind die menschlichen Verhaltenscharakteristika sehr komplex und daher auch schwer zu verstehen. Aus diesem Grund wird versucht, die grundlegende Funktionsweise von neuronalen Schaltkreisen anhand der Anatomie und Morphologie einfacherer Lebewesen zu analysieren. Die Fruchtfliege *Drosophila melanogaster* ist ein in der neurobiologischen Forschung sehr beliebter Organismus, da er Studien von komplexen Systemen auf einem vergleichsweise simplen Modell erlaubt. Aus diesem Grund werden die neuronalen Strukturen der Fruchtfliege mikroskopisch gescannt und in großen Datenbanken gespeichert, um sie datenverarbeitenden Systemen zugänglich zu machen. So sind neuartige Visualisierungsmethoden bereits speziell an diesen biologischen Daten adaptiert und werden aktiv in der Forschung eingesetzt. Allerdings bleibt die Orientierung in den großen Datenbanken problematisch: Die traditionellen Tabellen-Interfaces erlauben zwar das Filtern und die Selektion von ausgewählten Gruppen von Einträgen, aber die Selektion anhand der räumlichen Lage bleibt weiterhin unmöglich.

Diese Diplomarbeit beschreibt ein neuartiges System, welches die visuelle Erforschung einer großen Menge an neurobiologischen Daten wesentlich erleichtert. Ermöglicht wird dies durch die Kombination der Standardvisualisierungstechniken mit einem visuellen Explorations- und Abfrageansatz für Datenbanken. Unserer System stellt drei Grundtypen von Datenbankabfragen zur Verfügung. *Path queries* liefern mit Hilfe einer intuitiven Skizzenschnittstelle alle Strukturen, die sich in der Nähe eines gezeichneten Pfads befinden. *Object queries* erlauben die Selektion von Strukturen anhand deren räumlicher Distanz. *Semantic queries* ermöglichen schnelle Suche anhand von semantischen Zusammenhängen, die in der Datenbank definiert sind. Das System wurde im Rahmen einer interdisziplinären Zusammenarbeit mit Experten aus dem Gebiet der Neurobiologie entworfen, welche den hohen Nutzen eines solchen Systems für ihre Forschungsarbeit bestätigten.

Contents

CHAPTER

1 Introduction

Nature and Nature's laws lay
hid in night: God said, 'Let
Newton be!' and all was light..

epitaph of Sir Isaac Newton by
Alexander Pope

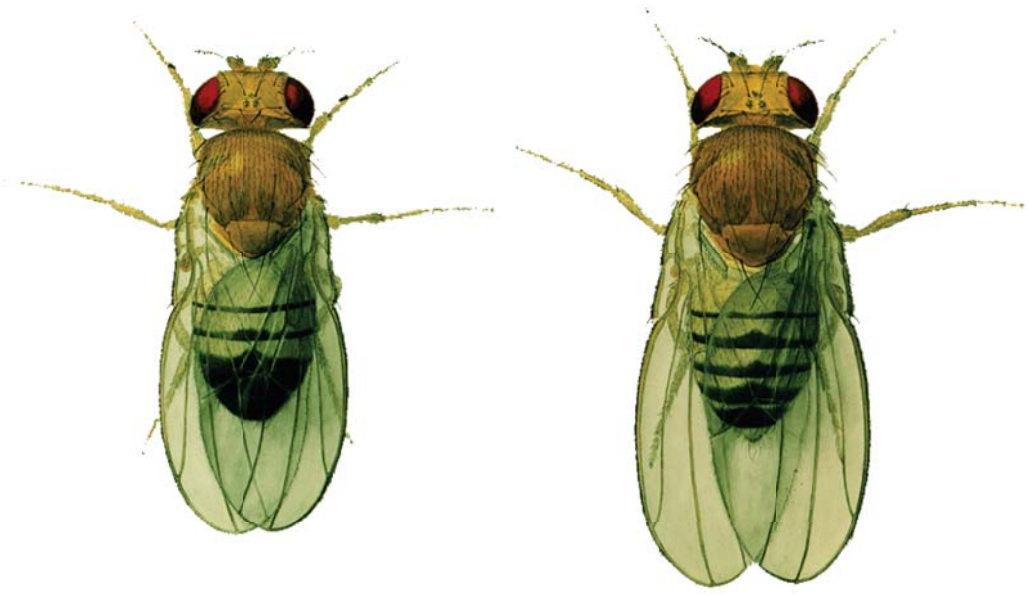
A MAJOR AMBITION of neuroscientists is to elucidate the complex behavior of organisms. The neural organs trigger and coordinate actions based on sensory perception, experience and circumstances. The exact mechanics of this process are still being investigated. For this purpose, neuroscientists observe the biochemical and physical actions in neuronal circuits and study the anatomy and the morphology of the neuronal systems. The fruit fly *Drosophila melanogaster* is a favorite model organism in the community of neurobiology, because it allows studies of complex processes on a simple example. Our collaborators from the Institute of Molecular Pathology at the Vienna University use molecular genetic techniques to study the function of individual neural circuits in Drosophila to explain the information processing which conducts complex behavior [20]. For this purpose, they build a database of microscopic images and segmented structures which allows them to study and explore their data. However, orientation in large collections of structures still poses a problem. Traditional table-view database interfaces allow filtering items and accessing known subsets of data, but do not

support selection based on spatial relationships. This thesis addresses this problem and describes visual queries as a method for accessing structures from the database based on the spatial information. In this chapter we give an introduction into the biological background in Section 1.1 and describe the workflow of our collaborators in Section 1.2. In Section 1.3, we outline the scope of this thesis.

1.1 Background

Drosophila melanogaster

Drosophila melanogaster or fruit fly gathers and feeds on spoiled food. Figure 1.1 illustrates a male and a female *Drosophilae*. This about 3*mm* little insect is a valuable model organism in genetics and development biology. This fact has been proven by many researchers. Dr. Edward B. Lewis, Christiane Nüsslein-Volhard, and Eric Wieschaus provided valuable information about human birth defects by



(a) (b)

Figure 1.1: Drawing of a male (a) and a female (b) *Drosophila melanogaster*. Image courtesy of Flybase [21].

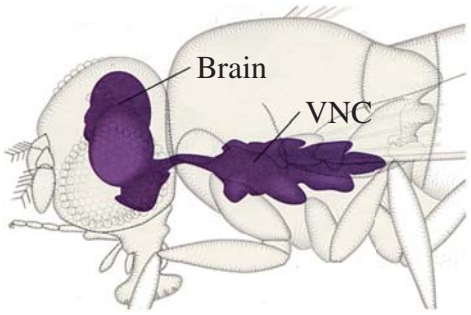


Figure 1.2: Central neuronal system of *Drosophilae* - the brain and the VNC. Image by the courtesy of Flybase [21]

studying the genetics of *Drosophila melanogaster*. They were awarded a Nobel Prize in physiology and medicine in 1995 for their work in the fields of genetics and development. The work of Mackay [43] highlights examples how studies of the simple organism *Drosophila melanogaster* helps to demystify relevant complex human traits such as sleep, alcohol dependency, and neurodegenerative diseases such as Parkinson and Alzheimer.

Drosophila melanogaster is a favorite model for research because it facilitates studies of intricate systems on a simple example. For example, human genetics and environmental factors influencing traits and behavior are very complex. Furthermore, accurate studies of the human genetics require large samples of individuals. Unlike humans, *Drosophila melanogaster* can be easily genetically modified. Furthermore, large samples of genetically identical individuals can be reared in laboratories under controlled environmental conditions and within short time [43]. The shortest generation time, i.e., egg to adult of *Drosophila melanogaster* takes 7 days under 28 °C [4]. The generation interval is an obstacle even for small mammals used in laboratories such as mice [43].

Nervous System of Drosophilae

The central nervous system of *Drosophila melanogaster* consists of two major organs - the brain and the ventral nerve cord (VNC), which is an important neural

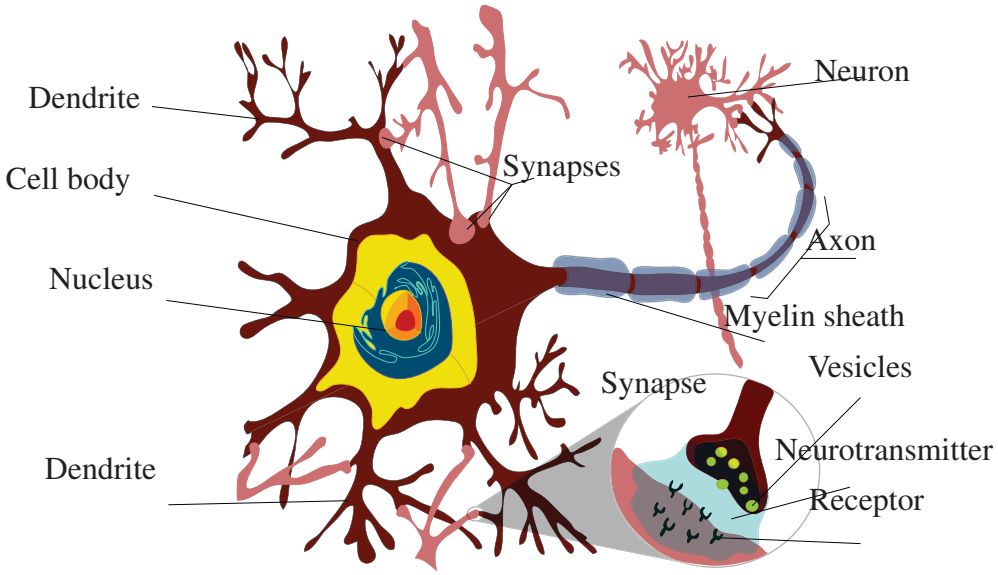


Figure 1.3: Structure of a neuron and the synapse. Figure inspired by the web of the Faculty of Chemistry of USCA [22].

cluster or ganglion on the ventral part of the fly's body. Figure 1.2 depicts the brain and the VNC with their anatomical context.

Neuronal organs such as brain and VNC are composed by neuronal cells called neurons. Figure 1.3 describes a neuron and its most important parts. Neurons transmit excitements by electrochemical signaling. The communication between individual neurons happens via connections called synapses as illustrated in Figure 1.3. The short processes called dendrites are excitement-receptive and lead the signal into the cell body. The long process called axon transmits the excitement from the cell body to other cells. The neuronal tissue is supported and nourished by tissues called neuroglia. The network of dendrites, axon and glial branchings, which form the bulk matter called neuropil, embeds the neuronal cell bodies [29].

Confocal Microscope

Fruit flies have little bodies about 3*mm* long. Their heads correspond to one sixth of their body length, so their brain does not exceed 0.5*mm*. With a spatial resolution 0.25*mm* of the scanning device, the fly brain would appear as a single dot. To observe structures in the fly brain, much higher spatial resolution is needed. In this case, neurobiologists cannot use familiar imaging techniques such as computer tomography or magnetic resonance imaging which have resolution about 0.5*mm*. Therefore, they use microscopical scanning methods, in particular confocal microscopy, which uses fluroescence, to image the specimen.

Before we explain how a confocal microscope works, we recall some facts about fluorescence. Some materials have the following property - they absorb light of some wavelength and emit light of a different wavelength. Furthermore, these materials still emit light, even if the light source is not present anymore. The physical background of the fluorescence effect is as follows. Particles in the material such as molecules and atoms absorb quanta of energy from the light of a certain wavelength in the form of photons. After absorbing photons, they move from the ground energy state to a higher energy state. The absorption of energy triggers emission of photons at a longer and less energic wavelength as has been

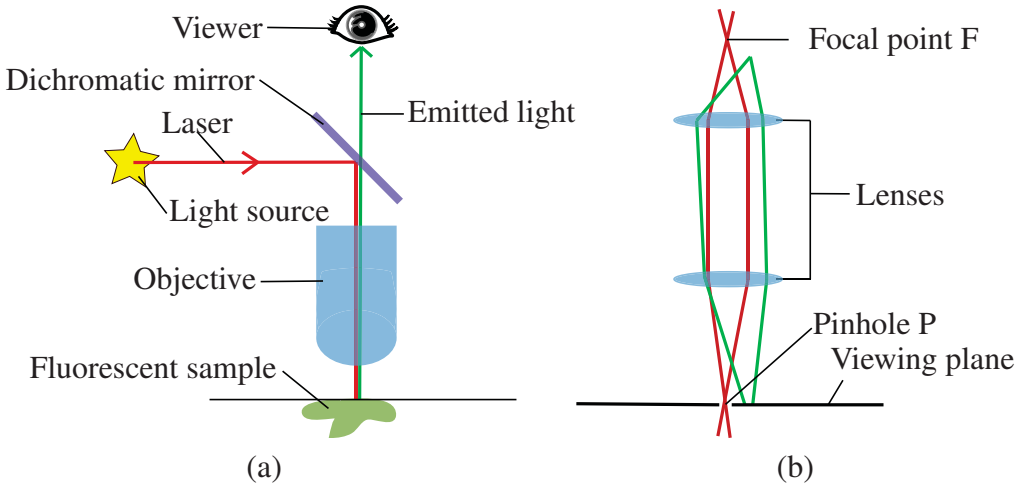


Figure 1.4: Optics of a confocal microscope - a dichromatic mirror (a) and a scheme with a pinhole (b). Figure inspired by the work of Prasad et al. [52].

Figure 1.5: A confocal microscope. Image courtesy of Carl Zeiss MicroImaging Inc. [67].

emitted from the light source. After emission, the particle returns to its ground energy state. The emission persists some time after exposure because not all particles emit absorbed energy immediately. Biologists know techniques to insert foreign substances into target tissues of a specimen which gives the tissues this property.

Figure 1.4 describes the optics of a confocal microscope. Fluorescent materials are exposed to monochromatic light, such as laser. In monochromatic light, only one wavelength is present. After exposure, fluorescent tissues emit light of a known wavelength. Confocal microscopes employ a dichromatic mirror to separate light going from the light source and light being emitted from the fluorescent tissue as illustrated in Figure 1.4a. Dichromatic mirrors reflect all light of a wavelength below a certain threshold. Light of wavelengths above this threshold passes through the dichromatic mirror. Confocal microscopes use dichromatic mirrors which reflect light going from the light source. The fluorescent tissue necessarily

emits light of longer wavelengths and thus passes through the dichromatic mirror of the microscope.
Normally, the whole sample is illuminated equally by the light source and thus the whole fluorescent tissue emits light. But only the image of the object in the focal point of the optical system F forms on the viewing plane as we show in Figure 1.4b. The image of F forms with the highest intensity. Nevertheless, the whole fluorescent tissue emits light and thus other parts of the fluorescent tissue contribute to the image of F as background haze. A pinhole P in the viewing plane where the image of F forms blocks the haze contribution. The focal point F and the pinhole P are conjugates - so this kind of microscopy is called confocal (conjugate-focal). The information about confocal microscopes sources from the work of Prasad et al. [52, 63].
Understanding the fluorescence effect, the principle of dichromatic mirrors and the lens optics is essential to comprehend the system of a confocal microscope. In Figure 1.5, we illustrate the whole system. The pinhole model creates at once only the image of the focal point representing one pixel in the scan. The adjoint computer creates the image pixel-by-pixel. In practice, three images of 512×512 pixels can be build-up per second [52, 63].

Gal4/UAS

Originally, neuronal tissue of *Drosophilae* does not virtue natural fluorescence. However, neurobiologists invented techniques, such as the Gal4/UAS system [51], which allow them to manipulate the targeted subset of neuronal tissue so that it exhibits fluorescence. We describe this technique as explained by Phelps et al. [51] without going into deep detail of the underlying biochemical actions. To learn more details, we direct the reader to the literature [51, 40, 58].
Genetic information of all living creatures from a virus to a human being is coded in nucleic acids DNA or RNA. The molecules of these nucleic acids include bases whose order codes genes. In Figure 1.6, we observe four types of bases in DNA - adenine (A), guanine (G), cytosine (C) and thymine (T). The genes direct production of proteins in cells, which geneticists call gene expression. However, not

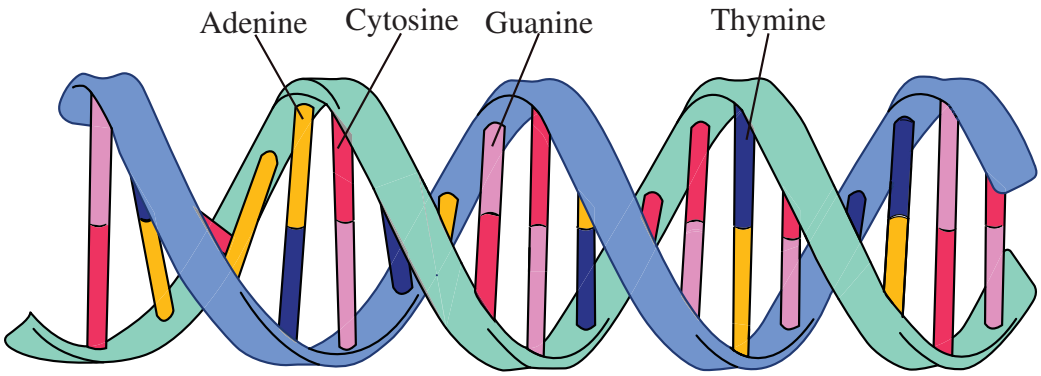


Figure 1.6: Bases of the DNA.

every gene is expressed as this process depends on many chemical activators and inhibitors. In genetics, a gene which is not expressed is called silent. Scientists know techniques, e.g., the Gal4/UAS system, to manipulate this process. They insert a target gene into the DNA which is expressed if and only if the activating Gal4-protein is present in the cell. They create two groups of genetically identical flies, i.e., lines. One line carries an endogenous promoter Gal4 responsible for the Gal4-protein production. The second line carries the target gene which is silent because the Gal4-protein is missing. If these two lines cross, the progeny of the cross expresses the Gal4-protein and also the target gene, because the Gal4-protein is present.
With the Gal4/UAS system, biologists can use the UAS-GFP target gene. Figure 1.7 depicts the Gal4/UAS when the UAS-GFP serves as the target gene. If a cell expresses UAS-GFP, a green fluorescent protein (GFP) is produced. Thus, tissues which express UAS-GFP virtue natural fluorescence and can be scanned using a confocal microscope.

1.2 Workflow

In this section, we outline the workflow of our collaborators. In particular, we focus on the preparation of their specimen, acquisition, registration and processing of their data.

Preparation of Specimen

Our collaborators use different techniques of injecting special fluorescent substrates into the neuronal tissue: the Gal4/UAS system as described in Section 1.1 and neuropil staining. In contrast to the Gal4/UAS, neuropil staining is less intricate - prepared tissues are washed in a staining (fluorescent) substrate. Our collaborators follow exact protocols while performing neuropil staining which are irrelevant for this work. As the Gal4/UAS system serves to evoke fluorescence in cells, they use it to label neuronal cell bodies and neuronal processes. In con-

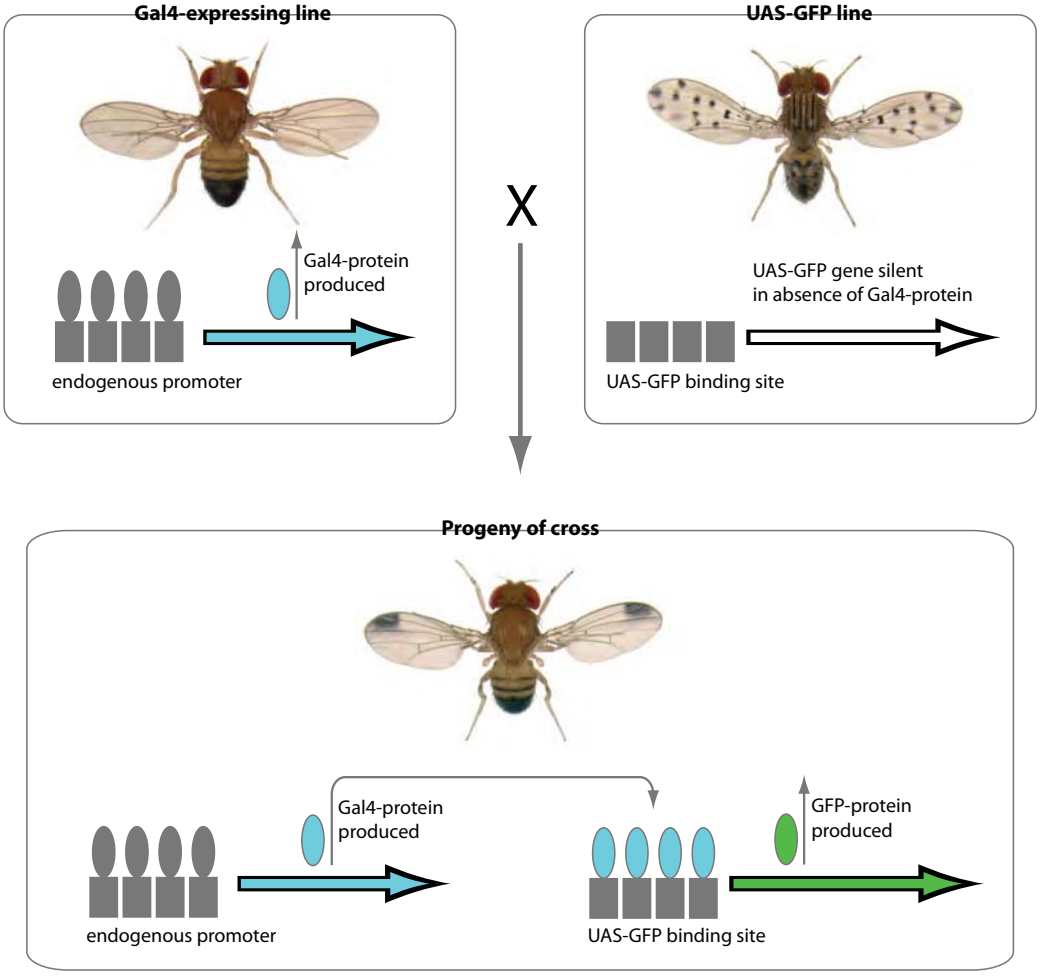


Figure 1.7: Gal4/UAS system. Image inspired by the article of Phelps [51]. The images of the fruit flies by the courtesy of Flybase [21].

trast to the Gal4/UAS system, neuropil staining is used to highlight extracellular synapses. With these techniques, our collaborators aim to highlight the neural organs entirely, or only some particular circuits and synapses. In this fashion, they prepare their samples for confocal microscopy scanning.

Acquisition

Our collaborators use confocal microscopy techniques to obtain images of their specimen at a spatial resolution $1\mu m$ using Carl Zeiss LSM 510 microscope with $25\times$ magnification. With this technique they generate images from a thin section of their samples. By acquiring many thin sections, they build-up a 3D image using volume reconstructions techniques. Our collaborators acquire a stack of 165 images of the brain and VNC at $1\mu m$ interval, both with resolution 768×768 pixels.

Registration

After acquiring the images of individual slices, our collaborators register the data. They use an additional staining of the neuropil which provides a stable morphological reference. The neuropil staining used for registration is independent from the neuropil staining used for highlighting synapses. They apply non-rigid registration algorithms [57] to register the data. Although this process is automatic, only manually verified scans are considered to have sufficient accuracy and are used for further processing.

Processing

In the acquisition step, our collaborators produced stacks of images, i.e., volumes, of the neural organs and registered them using a reference scan. The volumes of the entire organs are called *template volumes* and serve as anatomical context. Furthermore, our collaborators use them to segment important anatomical parts of neural organs, i.e., *template regions*, such as lobes and cavities.

In addition to the *template volumes*, scans of selected neuronal tissue have been generated. In these scans, i.e., *confocal images*, only particular neurons and synapses are highlighted. Our collaborators average them to reduce the interindividual variability and store the results as *Gal4-volumes*. For example, biologists observe separately male and female flies with the same Gal4-staining scheme. They aim to determine the differences between the male and female neuronal patterns. The averaging procedure diminishes the interindividual variablity within

Table 1.1: Types of Neuronal Data

Database representation	Representation	Staining method	Biological meaning
Cell body	Geometry	Gal4/UAS	Cell body
Projection	Skeleton graph	Gal4/UAS	Neuronal process
Arborization	Geometry	Neuropil	Synapse
Template volume	Volume	Neuropil	Brain or VNC
Template region	Geometry	Neuropil	Region of brain or VNC
Confocal image	Volume	Gal4/UAS	Selected neuronal tissue
Gal4-image	Volume	Gal4/UAS	Average of confocal images

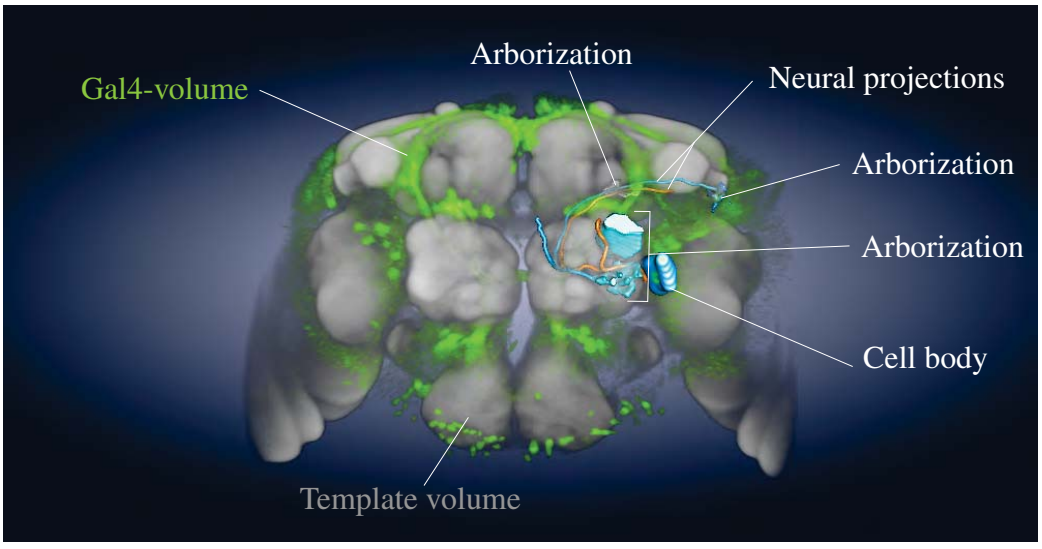


Figure 1.8: A template volume in conjunction with a Gal4-volume and a neuronal cluster consisting of three arborizations, two neuronal projections, and one cell body.

each gender group. The average scans contain only relevant features of each gender.
Gal4-volumes serve as a basis for segmentation of neurons. Our collaborators characterize neurons by the morphology of their cell body, the pattern of their processes, and by their synapses. For this purpose, they use the *Gal4-volumes* to segment neuronal cell bodies, processes and synapses. They refer to processes as to *neuronal projections* and to synapses as *arborizations*. Overall, four different objects are segmented: *template regions*, *cell bodies*, *neuronal projections* and *arborizations*. Segmentations are conducted by an expert user in Amira [1]. Our collaborators store the segmented surface of *template regions*, *cell bodies* and *arborization* as triangular meshes. In contrast, *neuronal projections* are represented as skeleton graphs generated by the skeletonizer plugin of Amira [62].
All acquired volumes and objects: *template volumes*, *confocal images*, *Gal4-volumes*, *template regions*, *cell bodies*, *neuronal projections* and *arborizations* are stored on a file server and their references are inserted into the relational database. The database also facilitates storage of annotations of individual objects and allows our collaborators to associate different objects into neural clusters. Table 1.1 gives an overview of different types of structures stored in the database. Figure 1.8 depicts a *template volume* of the brain which provides anatomical context in conjunction with a *Gal4-volume* and a neural cluster which consists of three *arborizations*, two *neuronal projections* and one *cell body*.

1.3 Scope of the Thesis

Our collaborators use the neuronal structures in their database for their research. They explore above all their morphology, spatial position and mutual connections to identify neuronal circuits responsible for specific behavior. However, the retrieving of objects from a database is difficult without a visual approach. For this reason, we provide visual queries for efficient structure retrieval.
The structure of this thesis is as follows. In Chapter 2, we give an overview of the state of the art in the fields of neuronal data exploration and visualization of microscopic data. In Chapter 3, we explain in detail visual queries we employ for

efficient structure retrieval. In Chapter 4, we provide additional implementation details. In Chapter 5, we present results of our work and a typical scenario of work with our system. We conclude the thesis in Chapter 6.

CHAPTER

2 State of the Art

Copy from one, it's plagiarism;
copy from two, it's research.

Wilson Mizner

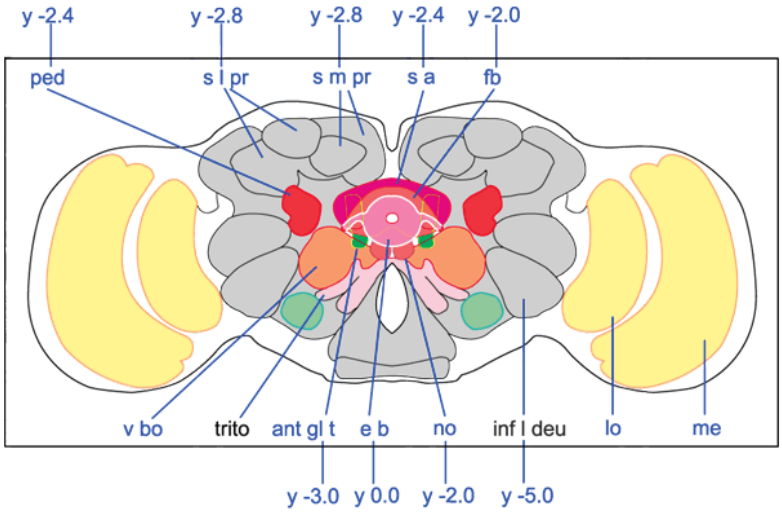
THE RESEARCH WORK we discuss in this thesis builds-up on three major pillars: databases of neurological data, visual and spatial queries, and visualization of microscopy data. In this Chapter, we give an overview over existing research in these fields. Among the numerous works we highlight only those which are relevant for this thesis.

2.1 Atlases and Databases of Neurological Data

An introduction into the challenges of databasing the neurological data together with a survey of the underlying techniques are given by Koslow and Subramaniam [38] and Chicurel [15]. In their book, Koslow and Subramaniam introduce their reader to current issues, methods and tools in neurobiology. They also present the state of the art in applications in clinical and basic research. This includes above all the new ways to acquire, store, visualize, analyze and share the neurological data. Chicurel [15] discusses requirements for a brain database and gives numerous examples.

Atlases offer illustrated and annotated descriptions of organs. They are essential

for research and education. They help to localize neuronal structures and bring a system and coherency into the nomenclature of anatomical structures. Coherency, a logical system, and localization are crucial in computer-aided applications [10]. In neuroscience, the construction of such tools is necessary for easy and quick localization, search and analysis of neurological structures. In their article, Maye and al. [45] present an overview of methods for neurological data reconstruction and visualization, creating atlases and registering the data. The latter should also support a researcher in comparing individual structures imaged in his or her experiment with structures in the atlas. At present, several atlases of neurological structure are available online.



Abbreviations			
ped	pedunculus	fb	fan-shaped body
s l pr	superior lateral protocerebrm	v bo	ventral body
s m pr	superior medial protocerebrum	trito	tritocerebrum
s a	superior arch	ant gl t	antennoglomerular tract (a.k.a. antennocerebral tract [a c t])
e b	ellipsoid body	no	nodulus
inf l deu	inferior lateral deutocerebrum	lo	lobula
me	medulla		

Figure 2.1: A typical illustration in the Flybrain on line atlas. Image courtesy of Flybrain [2].

The Lundeck Institute in Denmark provides a web interface to human brain called Brain Explorer [31]. The tool enables visual browsing in the database of annotated structures of a human brain. Concerning the neuronal system of flies, we mention Flybrain [2] and FlyBase [21]. They offer a classical atlas describing the brain of *Drosophila melanogaster* and an interactive navigation in the database of images and sketches by clicking on their labels or structures themselves. However, these online tools use databases containing only 2D images.
Bertrand et al. [7] present an atlas of the mouse brain called Neuroterrain. Their database consists of segmented 3D structures represented as geometry and a broad set of confocal image stacks for 3D visualization. Nevertheless, they do not describe interaction with neurological structures. Bezgin et al. [9] described an interactive system for neurological data exploration. Their work focuses on **co**llations of **co**nnectivity data on the **mac**aque brain (CoCoMac). Displaying connectivity is essential to understand the team work of individual structures in the brain. Their system includes databases containing both - morphology and connectivity data. Burns et al. [14] presented a framework NeuARt II for atlas-viewing. Users can view, annotate and save their loaded data for later access. The application supports also import, viewing and saving of copyrighted atlases.

2.2 Visual and Spatial Queries

Employing humans in data-mining often leads to better results, especially when automated procedures fail. However, incorporating humans has disadvantages when a huge amount of data has to be analyzed, which is often the case. Finding valuable information without supporting tools is complicated and laborious. Semi-automated visual analysis is faster than purely human-conducted. Visualization of the working data helps the user to identify important trends and exceptions [17]. Thus, visual exploration techniques are highly on demand and often come conjoined with automatic techniques. In Section 2.2 we give an overview of existing visual approaches to queries and data-mining.

Visual Queries and Data-mining

Dethrick el al. [19] discuss fundamental concepts of giving visual feedback for queries. Their database models relationships of multidimensional data. Their system supports queries created with drag and drop. Ahlberg et al. [3] presented and evaluated a concept of queries using graphics widgets and sliders. This supports direct filtering of the results and direct visual feedback.
Further prominent techniques comprise brushing, picking, and linking of multidimensional data. Reina et al. [56] describe brushing in volumetric data which modifies the actual query. Becker describes [6] mapping the contents of a database to a volume. His system implements brushing and picking as methods for interactive filtering. Martin el al. [44] treat their multidimensional data with parallel coordinates. Using interactive brushing over the visualization using parallel coordinates, they achieve interactive filtering. Keim [36] classifies information visualization techniques. He separates the user-interaction into three steps employing interactive techniques such as zoom, filtering, linking and brushing. First, the user gets an overview where he identifies important trends. Second, he or she focuses and interacts with the data, e.g., zooms into interesting regions of data. Third, details about selected regions are displayed on-demand.

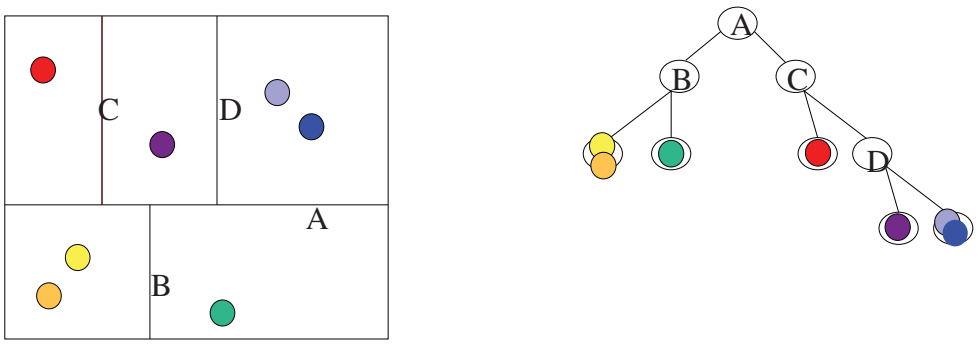
Queries in Spatial Databases

An exhaustive surveys on multidimensional and metric data structures and query methods for spatial databases are given in the works of Gaede et al. [24] and Samet [61]. Spatial data are defined by an explicit information about their extent and position in space. In their book, Gaede et al. discuss novel types of queries on these types of data - point and region queries. A point query returns all objects intersecting a selected point in space. A region query selects all objects intersecting a selected region in space. Users can interact with the query results, e.g., make a union or intersection.
There exists no linear mapping of a multidimensional space into 1D space which fully preserves spatial locality. Voxels which are direct neighbors in a 3D volume, might not be direct neighbors when serialized on a disk. However, there exist

techniques which attempt to maximize locality. They are essential for fast search operations in spatial databases and ensure database scalability. Among numerous approaches to organization of spatial data on a disk, we highlight the most prominent ones - Kd-trees, Octrees, BSP-trees and space-filling curves.

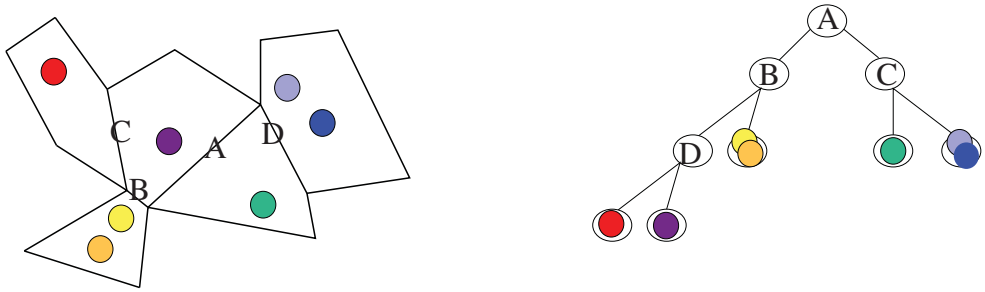
Kd-trees

Kd-trees belong to the most essential search data-structures. The tree represents a recursive binary subdivision of a d-dimensional space. Every time, a (d-1)-dimensional hyperplane splits the parent-space or parent-subspace is divided into two child-subspaces. The subdividing hyperplanes are always orthogonal to each other as illustrated in Figure 2.2. Similar to kd-trees are quadtrees and octrees, where the 2D or 3D subspaces are represented as nodes. In each partitioning step, the nodes are split into 4 quadrants or 8 octants. In practice, quadtrees and octrees are used for effective storage of solid objects [27]. A post-order traversal (left, right, root) of these trees produces an ordering of data well-preserving spatial proximity of individual nodes.



(a) (b)

Figure 2.2: Principle of a Kd-tree - a recursive space-subdivision (a) and a corresponding tree (b).



(a) (b)

Figure 2.3: Principle of a BSP-tree in 2D- a recursive polyhedral (polygonal) subdivision (a) and a corresponding tree (b).

BSP-trees

In contrast to Kd-trees which split space with orthogonal hyperplanes, BSP-trees represent recursive subdivision using arbitrary hyperplanes. In each recursion step, a (d-1)-hyperplane cuts a parent polyhedron into two child polyhedrons as illustrated in Figure 2.3. This approach is commonly used in computer graphics for scene subdivision to determine, e.g., visibility of objects [27].

Space-filling curves

Space-filling curves $s(t)$ are fractal curves which provide one-to-one mapping between points in a multidimensional and a 1D space $\mathbb{R}^n \rightarrow \mathbb{R}$ and well-preserve the spatial distance between points. A family of 2D space-filling curves was introduces by the mathematician Giuseppe Peano in 1890. Figure 2.4 illustrates a space-filling curve $s(t)$ which maps a 2D square into a 1D curve $\mathbb{R}^2 \rightarrow \mathbb{R}$.
A discrete representation of these curves can be computed using an L-System[54]. An L-System consists of an initial shape, alphabet and grammar building rules. Iterative application of grammar rules transforms the initial shape and converges to the solution. In Figure 2.5, we show building rules for different types of space-filling curves - a Peano curve, Hilbert curve[28], and Z-order curve [49]. After an infinite number of iterations on a 2D curve each point of a square belongs to

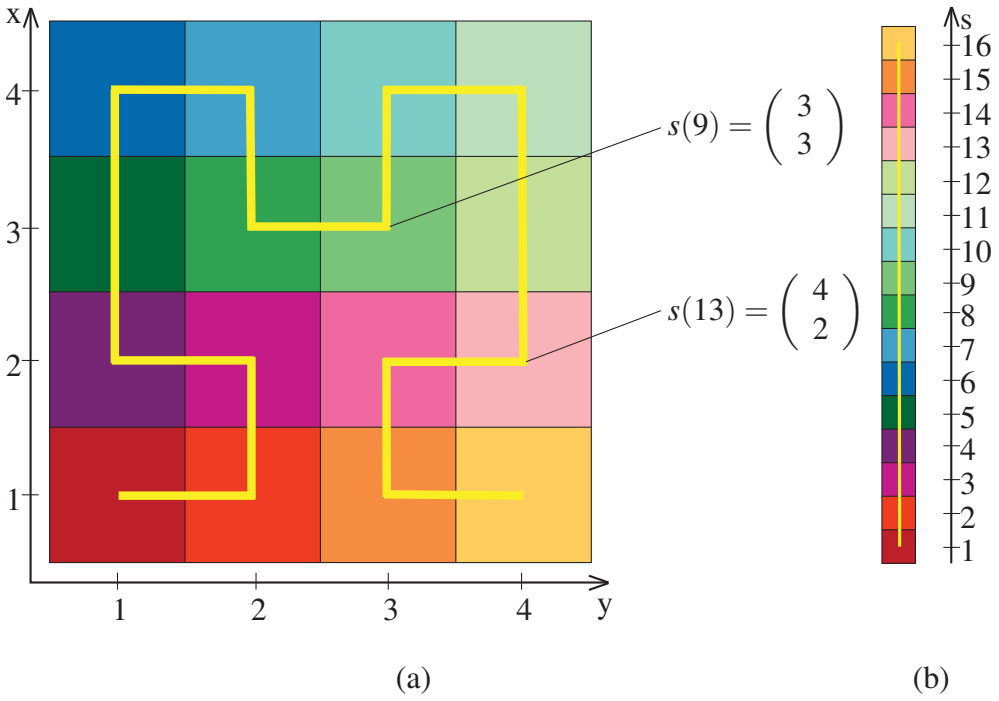


Figure 2.4: A 2D Hilbert curve $s(t)$ (a) maps of a 2D space onto a 1D curve (b).

the curve. Space-fillings curves are used to order discrete multidimensional data on a disk. Figure 2.4 shows a serialization, i.e., a linear order, of a $2^b \times 2^b$ matrix of points. In the remainder of this thesis, we refer to linear ordering of discrete multidimensional data as to a scan.

For linear ordering of multidimensional data on a disk, a Hilbert curve has proven to be a good choice, because in the family of space-filling curves, it preserves the proximity the best [32]. In his article [25], Gilbert describes construction of a cube-filling Hilbert curve. There exist many algorithms for 3D Hilbert-scanning - algorithms by Quinqueton [55], Sagan [59], Millar [48], and Kamata [33, 34]. However, they are restricted to cube regions of size $2^b \times 2^b \times 2^b$ and use recursion. The problem of Hilbert scanning of arbitrarily-sized cuboid regions is addressed in the article by Zhang and Kamata [68]. The authors propose an efficient algorithm for 3D Hilbert-scanning of such regions.

Hilbert's curve provides a scanning order for multidimensional data which well preserves locality. Thus, it is widely used in digital image processing. However,

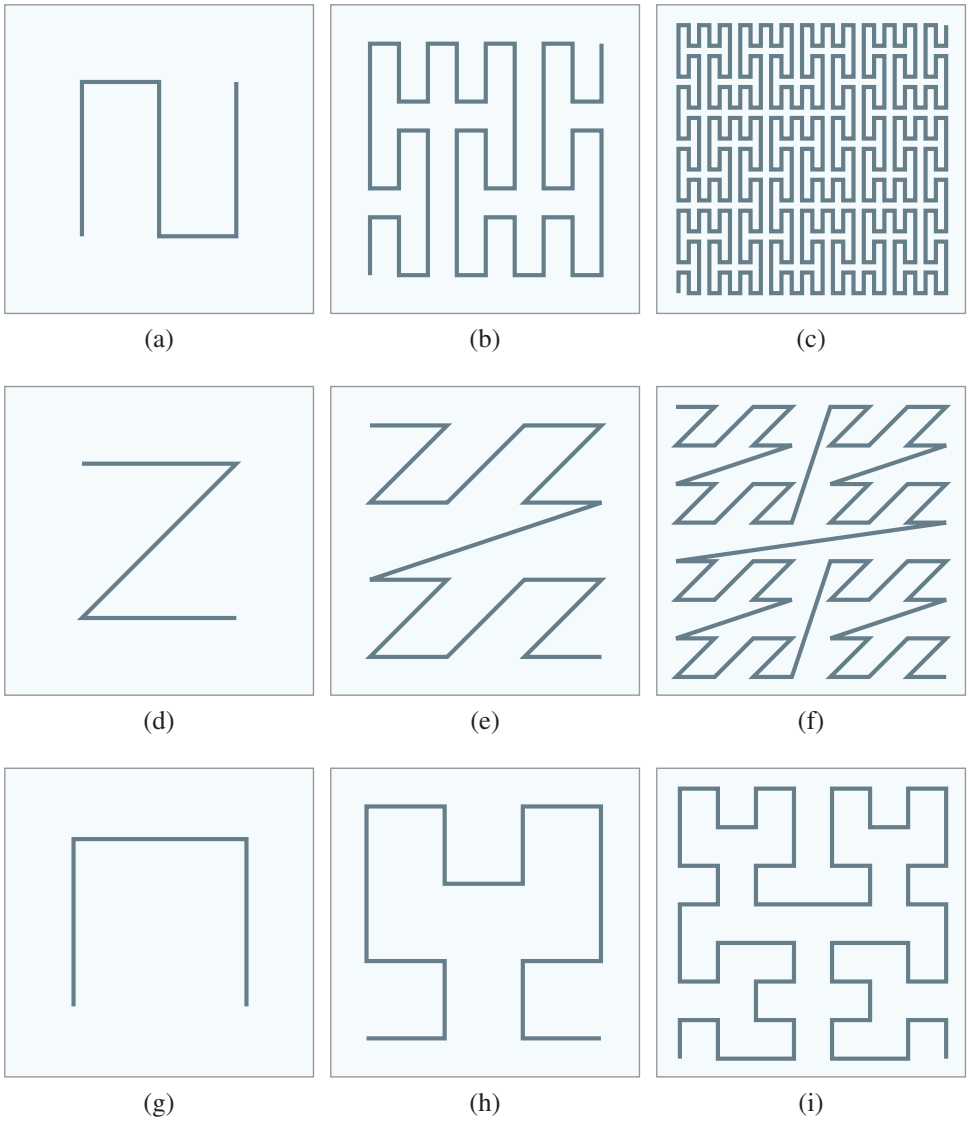


Figure 2.5: A Hilbert curve mapping of a 2D space onto a 1D curve. Initial shapes of the Peano curve (a), the Z-order curve (d), and the Hilbert (g). The curves after first (b), (e), (h) and second (c), (f), (i) iterations.

sampling of space-filling curves $s(t)$ poses a difficulty. Given a cuboid region and a point $s(t) \in \mathbb{R}^n$, the deduction of t is very costly. There exists no formula for direct computation of the curve $s(t)$ or inversely $s^{-1}(s(t))$. The whole curve $s(t)$ must be computed for the whole region using one of the proposed algorithms.

Queries in Databases of Neurological Data

The exploration of neuronal databases has been investigated by several researchers. Fredriksson et al. [23] describe an online accessible visual human brain database system. Their framework provides direct queries on raster data such as visual queries on images. User selects two rectangles in slice views and defines a volume of interest (VOI). These can be linked with different boolean operations. Then, he or she queries images within the selected VOI using operations such as thresholding. However, the authors do not use explicit representation of structures, i.e., geometry. Press et al. [53] focused on the graphical search within neuroanatomical datasets. Their system called XANAT allows studies, analysis and storage of neuroanatomical connections. Users perform searches by graphically defining a region of interest to display the connectivity information for this region. Furthermore, their system supports also textual search using keywords describing a particular region. Sherbondy et al. [64] describe a novel set of interaction techniques that makes the exploration and interpretation of pathways in brain easy. The queries in their approach are interactive - the user marks a box or ellipse-shaped region and all pathways crossing this region are shown. Again, the framework supports boolean operations on the regions of interest.

2.3 Visualization of Microscopy Data

Visualization techniques for microscopy data are being explored since computer-aided microscopy is used in bioscience. Early work on microscopy data focused on image processing techniques, notably noise reduction, because the early microscopy scans were of worse quality than today. The microscopy scanning devices progressed in the last decade and the microscopy images of today are of much higher quality. The evolution of computer and microscopy hardware solved many of the problems addressed in the earliest works. Thus, we offer only a brief survey of the early research in this field.

The early works [35, 5, 60] refer mostly to the visualization of cells. Kaufman et al. [35] explored visualization of cells focusing on depth cues. They employed gradient dependent shading and dynamic positioning of the light source to en-

hance depth perception. The article by Avila et al. [5] focuses on visualization of nerve cells. They used cutting planes to reveal inner structures which would be otherwise occluded by the cell's surface. Sakas et al. [60] provided a universal tool for visualization of both - static and dynamic structures of confocal microscopy data.

More recent approaches employ modern and innovative user interfaces and state-of-the art visualization techniques. Leeuw et al. [16] also work with time dependent confocal microscopy data. They presented a tool tailored for analyzing of chromosomes during the cell division. Their system supports a virtual environments, e.g., Cave. The VR setup includes shutter glasses with a head-tracker and a projection plane. The tool provides volume and isosurface rendering and numerous interaction and inspection methods. The VR-setup enables pointing and interaction with a "virtual hand". Furthermore, their framework supports multi-channel volume data where each color channel (RGBA) represents a concurrent dataset.

O'Conor et al. [50] and Wang [65] used programmable graphics hardware to achieve higher performance. O'Conor et al. implemented the marching cubes algorithm [41] for isosurface generation, direct volume rendering techniques and interactive transfer functions.

In numerous cases, researchers want to analyze and process a collection of datasets simultaneously, e.g., to test their hypothesis. Processing each dataset individually is time-consuming. On the other side, in fully-automated batch processing of the whole collection of datasets, only final results are displayed, and the user cannot intervene into the process. The article of Leeuw et al. [17] describes techniques for visualization and analysis of large data collections applied on confocal microscopy data. The authors present a system using a hybrid approach - combining the two processing approaches. Furthermore, they incorporated a plethora of visualization tools such as histograms, scatter plots and parallel coordinates [30].

Melek et al. [47] work with fibrous, thread-like data acquired with electron microscopy scanners [18] and knife-edge microscopy [46] invented at the Texas A&M University. In their article, Melek et al. proved that photo-realistic rendering using global illumination of the fibrous data leads to their better visual perception. They applied a model for hair rendering to visualization of neuronal

networks.

The visualization techniques highlighted in this survey employ elaborate techniques of visualization and enhancement. Direct volume rendering techniques were combined with shaded isosurfaces using techniques for mixing volumetric images and geometry [13, 39], and even rendering of multichannel volumes [8]. The system described in this thesis uses the state-of-the-art visualization techniques in conjunction with a novel approach of visual search in a database of neurobiology structures using the spatial and semantic information.

CHAPTER

3 Visual Queries in Neuronal Data Exploration

Pigmaei gigantum humeris impositi plusquam ipsi gigantes vident.

Sir Isaac Newton

OUR SYSTEM demonstrates a compact computer-aided visualization tool tailored for a neuroscientist. It supports user's own relational database which is connected to a file server. It includes powerful visualization techniques and enables efficient querying of the individual structures from the database. In this chapter, we give a general overview of our system and a detailed description of its parts. Furthermore, we describe the used visualization techniques, and, in particular, the novel approach to visual queries.

3.1 System Overview

The core of our system is the open-source volume rendering framework VolumeShop [11] where we smoothly integrated the specialized plugins. VolumeShop has a flexible architecture which allowed us to rapidly integrate new functionality. The architecture of the system consists of a relational database connected to a file server, spatial indices of the data, and the visualization tools. The entities communicate via interfaces. The latter are responsible for the construction of the

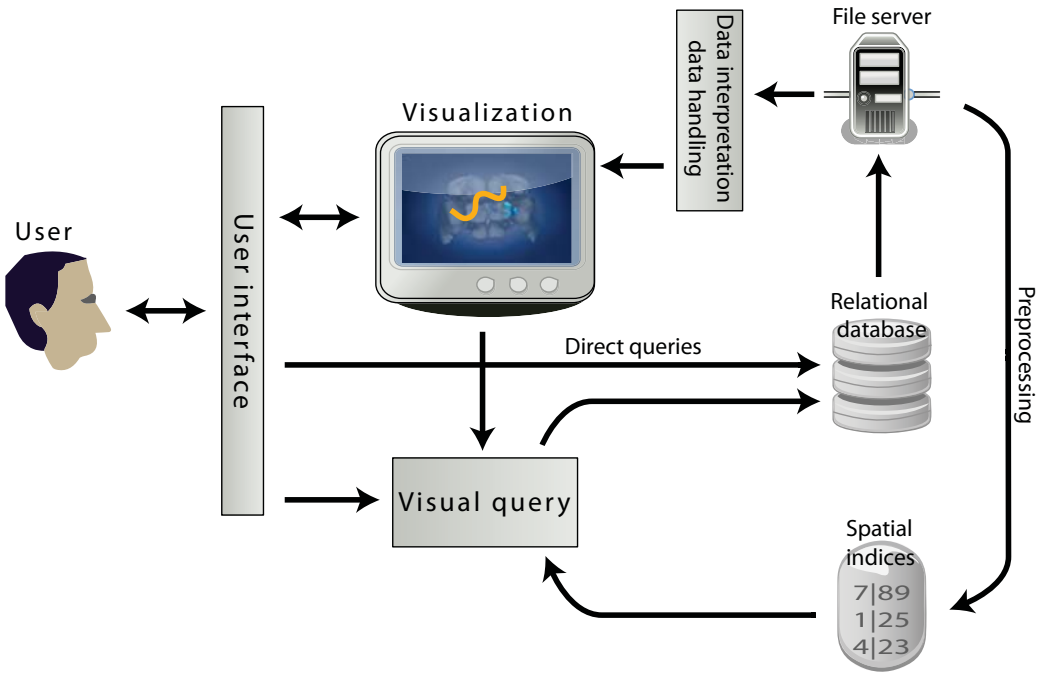


Figure 3.1: Overview of the system.

visual queries. Figure 3.1 illustrates the connections of entities and concept of the entire system. In this Section, we discuss each important part of the system in greater detail.

File Server

All data provided by our collaborators are stored on a file server. The application also supports offline access where the files are stored on a local medium and a local relational database. However, the database and the server are intended for access over a network and the datasets are transferred to the local machines on-demand. In this manner, our collaborators are able to provide and maintain their data on a server and their clients view and explore them at their local machines. In the future, our collaborators plan to extend the collection of their datasets and insert them onto the server. This requires our system to be scalable.
Our collaborators use Amira [1] for preprocessing and segmentation of their data.

Thus, all data they provided were in the Amira's native general-purpose file format called *AmiraMesh*. This file format is very flexible - it supports storage of a variety of objects. We mention only object types relevant to us: volumetric data, surface geometry including materials, and skeleton graphs. As we show in Table 1.1, *template volumes*, *confocal images*, and *Gal4-images* are stored as volumes. *Arborizations*, *template regions*, and *cell bodies* are represented as geometry and *projections* as skeleton graphs.

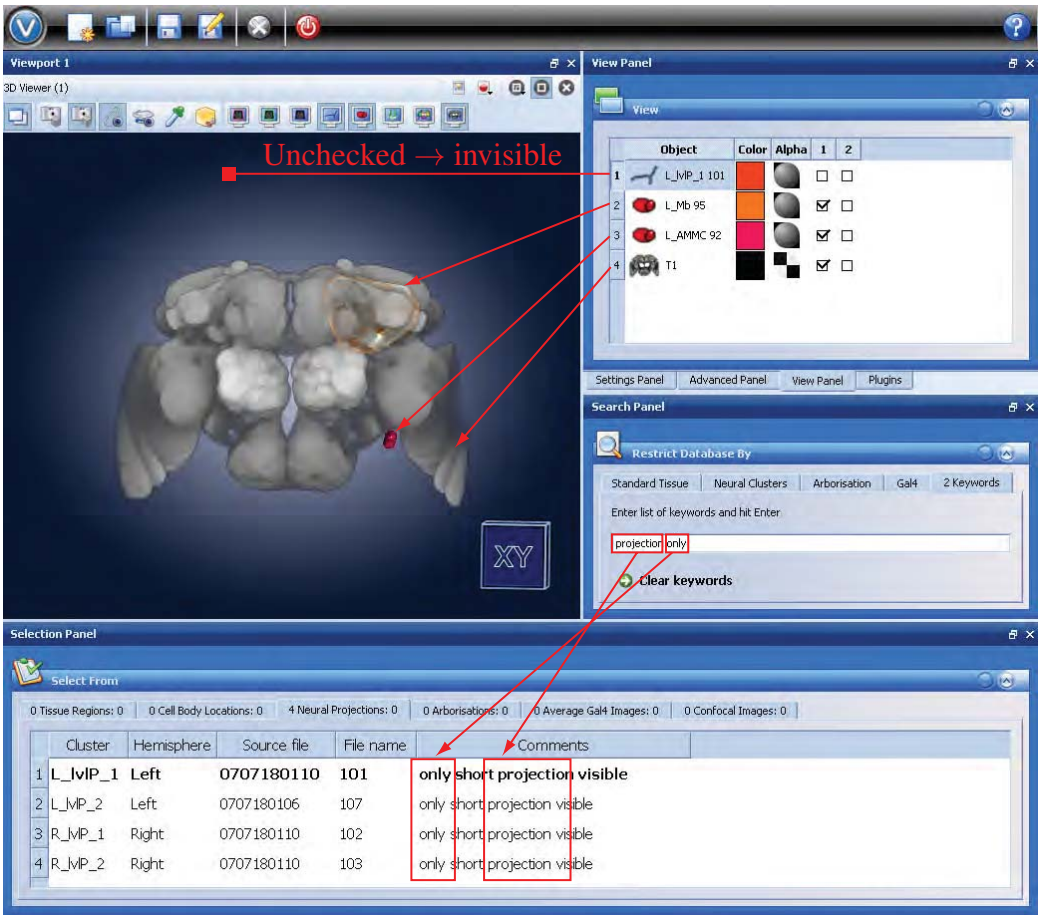
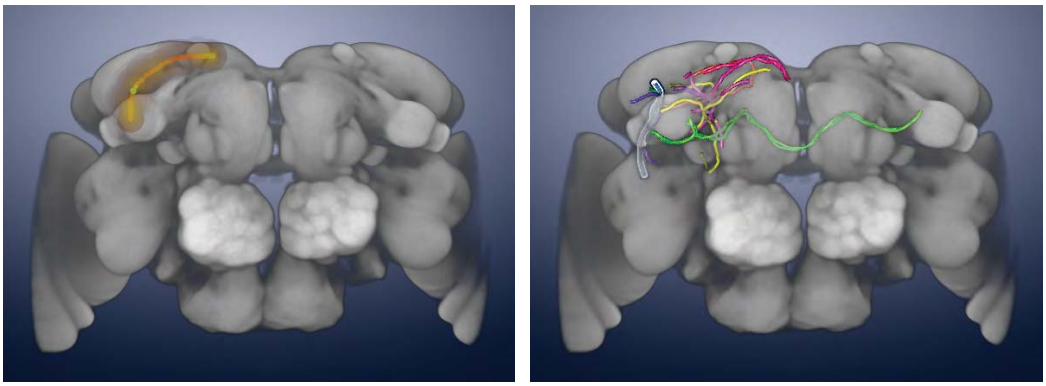


Figure 3.2: A screenshot of our system. The list of structures in the selection panel is restricted by the keywords *projection* and *only*.

Relational Database

The relational database stores additional information about the present neuronal structures. This additional information comprises properties such as expert's comments, clusters, whether the structure was retreived from a brain or from a VNC, and the gender of the organism it was retreived from. The user interface contains a *selection panel* which shows a list of structures in the database. User selects individual structures from the file list to be added into the visualization, which then also appear in the *view panel*. The file list in the *selection panel* can be filtered using the additional information stored in the database. In the *search panel*, the user makes a set-up to display only the structures which were retrieved from a female brain. The structures which were retreived from a VNC or from a male organism do not appear in the list. The *search panel* also contains a field for keyword-based search. Figure 3.2 depicts a typical screenshot of the user interface showing a visualization window, a *view panel*, a *search panel*, and a *selection panel*. In the *search panel*, we restricted the list with the keywords *projection* and *only*. The restricted list in the *selection panel* contains only files containing both keywords in their annotations. This feature easies the orientation in the long file listings stored in the database.



(a) (b)

Figure 3.3: A path query - a sketched path with a tight region around (a) and a neurons crossing this tight region (b).

Visual Queries

The database provides context-based search. Users picks structures by their name or by using filtering options. However, search and exploration based on the spatial information is not supported. Users cannot automatically select structures crossing particular points in the volume (point-based queries [24]), the closest objects to an object of interest (object-based queries), and directly retrieve information about objects by clicking on them in the visualization (semantic queries). However, these tasks are of the major interest of our collaborators. We extended the idea of point-based queries to path-based queries. A path-based query provides efficient retrieval of structures crossing a tight region around a path sketched over the visualization. In Figure 3.3 we explain the concept of a path-query. First, the user interactively sketches a path over the visualization as in Figure 3.3a. Consequently, a list of available structure crossing this region appears in the menu. Second, he or she loads, e.g., all *projections* from the list path as shown in Figure 3.3.

Our system provides the user with three types of visual queries - spatial queries, object queries and semantic queries. In Figure 3.1, they are represented with the entity *visual query*.

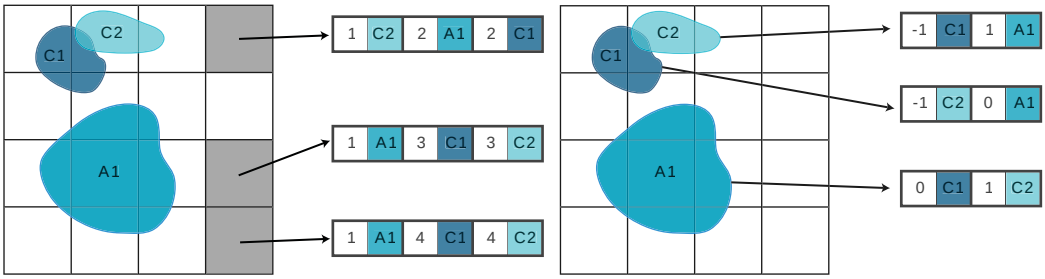


Figure 3.4: Distance tables simplified to 2D containing point ditances (a) and object distances (b).

Spatial indices

Spatial indices represent packed spatial information about individual objects stored on the file server. They are necessary for efficient object and path queries. At runtime, we must be able to rapidly calculate distances to n closest structures from any point in the volume. Furthermore, we must be able to quickly provide a list of n closest objects to any object of interest. Considering that our file server stores several hundreds of structures, a runtime calculation is impossible. Thus, we compute two tables containing point and object distances in a preprocessing step using a separate tool. Figure 3.4a illustrates fields of the mentioned distance tables. Each row of this table refers to a point in space. The distances are measured using the chamfer metric. In Figure 3.4b, each row refers to an object. When two objects intersect, the corresponding value in the table is negative and its absolute value indicates the intersection volume. The application accesses these precomputed tables for distance look-up at runtime which makes the distance calculation fast.

Visualization

The central node of our system is the *visualization* entity as we indicated in Figure 3.1. The numerous featured visualization tools make our application a compact toolbox for 3D data exploration. The toolbox includes standards for viewpoint control such as rotation, zooming and panning, masking control such as clipping planes and cropping boxes and different renderers. The latter provide slice-rending, direct volume rendering with interactive transfer function and geometry rendering. Other utilities supporting user's work comprise synchronized viewing windows, image and video-capture and saving of the current session. Figure 3.5 presents some of the featured visualization techniques and utilities. We describe applied visualization techniques in Section 3.2 without going into deep detail as this thesis focuses on the novel visual queries and the featured visualization techniques are state-of-the-art. The visualization entity is essential to concept a visual query.

3.2 Visualization techniques

One of the major requirements of our collaborators is the support for combined visualization of geometry and volumes and concurrent rendering of multiple volumes. A typical case is, e.g., locating more structures such as *cell bodies* and *arborizations* inside a *template volume*, which provides an anatomical context. At the same time, they want to explore additional volumes, e.g., *Gal4-images* representing some stained neuronal tissue of interest. In this particular case, as demonstrated in Figure 3.6, concurrent visualization of multiple volumes and geometry is necessary. Furthermore, exploration of numerous structures at the same time leads to a visual clutter. Our aim is to maximize the information the user can get from the visualization and to minimize the clutter. In this section, we discuss our approaches to the concurrent visualization of volumes and to the geometry rendering.

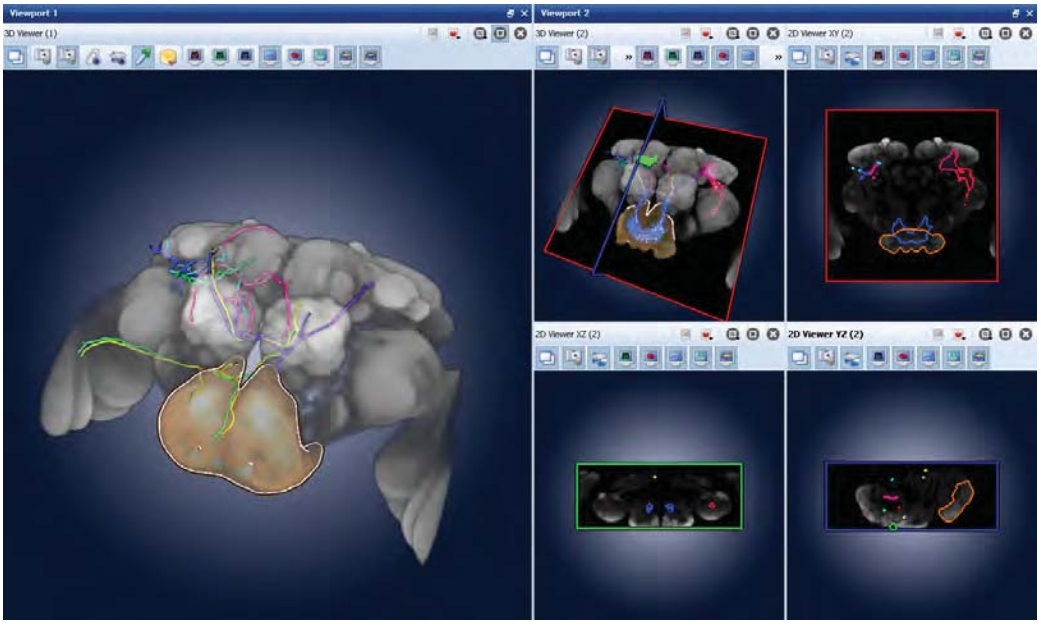
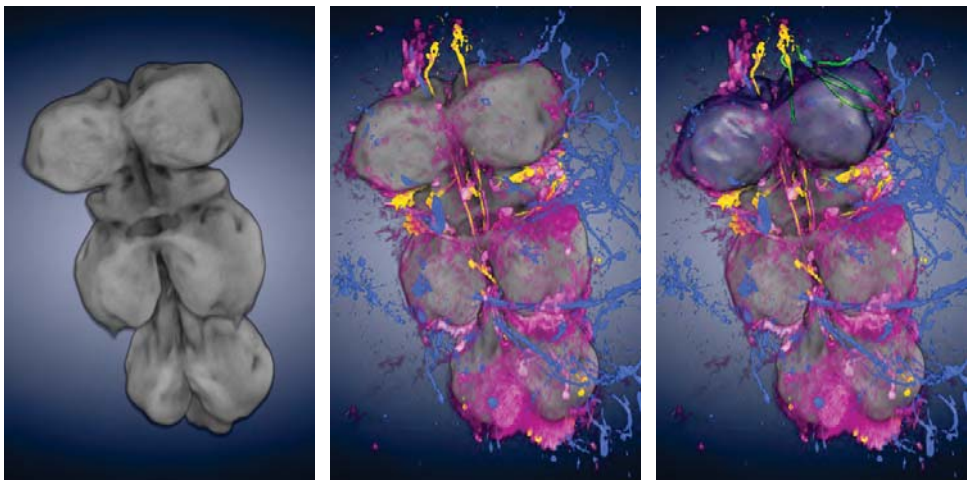


Figure 3.5: A screenshot of our system showing some of the featured visualization techniques, notably direct volume rendering combined with geometry rendering and slice rendering in multiple synchronized views.



(a) (b) (c)

Figure 3.6: Visualization of VNC - a template volume of a VNC (a), concurrent rendering of the VNC volume and three Gal4-volumes (b), concurrent rendering of VNC volume, three Gal4-volumes with template regions and neuronal projections represented as geometry (c).

Volume Rendering

Maximum Intensity Projection (MIP) is often used in biomedical visualization. This technique retains the highest values along the projection rays. Our collaborators also preferred MIP over direct volume rendering (DVR). However, employing visualization using MIP destroys the depth perception. On the other hand, using

(a) (b) (c)

Figure 3.7: Comparison of DVR (a), MIP (b) and MIDA (c).

DVR instead of MIP leads to occlusion of important maxima. This is a significant problem especially when a researcher explores several Gal4-volumes and the template volume. The template volume should build an anatomical context but should not occlude the important information provided by Gal4-volumes. Therefore, our system uses a hybrid approach called Maximum Intensity Difference Accumulation (MIDA) described by Bruckner et al. [12]. This technique ensures that the strong features of the stained data remain unoccluded by the template volume. Figure 3.7 compares visualizations using these three techniques.

Maximum Intensity Difference Accumulation

The article of Bruckner et al. [12] describes MIDA for mono-channel data. They employed an additional modulation factor for the color and opacity composition equation used for DVR. Equation 3.1 presents the color composition in DVR. The color and opacity are accumulated along the rays. $\mathfrak{C}_i$ and $\mathfrak{A}_i$ denote the color and opacity of the i-th sample along the ray-traversal. C_i and A_i represent the accumulated color and opacity at the i-th step of the ray-traversal. The initial values C_0 and A_0 are both zero.

$$\begin{aligned} A_i &= A_{i-1} + (1 - A_{i-1})\mathfrak{A}_i, \;\; i = 1..n \\ C_i &= C_{i-1} + (1 - A_{i-1})\mathfrak{A}_i\mathfrak{C}_i, \;\; i = 1..n \end{aligned} \tag{3.1}$$

Equation 3.2 denotes a compositing method employed in MIDA. The additional parameter $\mathfrak{B}_i$ represents the modulation factor at the i-th step of the ray traversal.

$$\begin{aligned} A_i &= \mathfrak{B}_i A_{i-1} + (1 - \mathfrak{B}_i A_{i-1})\mathfrak{A}_i, \;\; i = 1..n \\ C_i &= \mathfrak{B}_i C_{i-1} + (1 - \mathfrak{B}_i A_{i-1})\mathfrak{A}_i\mathfrak{C}_i, \;\; i = 1..n \end{aligned} \tag{3.2}$$

We calculate the modulation factors $\mathfrak{B}_i$ at each sampling step as in Equation 3.3. We introduced the parameter δ_i to make the later transition to multi-channel MIDA easier. Parameter s_i denotes the density sampled at the i-th step and $max_i(s_k)$ denotes the maximal density of samples taken at steps $k = 0..i-1$. We

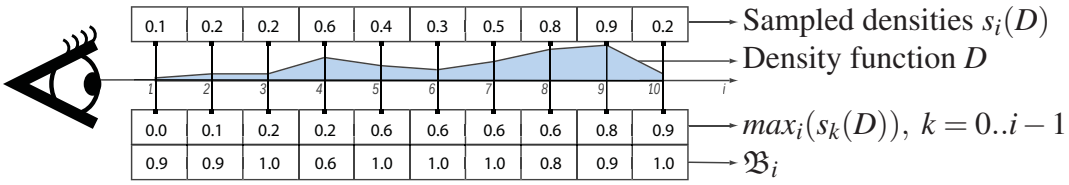


Figure 3.8: Ray traversal and calculation of $\mathfrak{B}$.

illustrate this procedure in Figure 3.8.

$$\mathfrak{B}_i = \begin{cases} 1-\delta_i = 1-(s_i - max_i(s_k)), & \text{if } s_i > max_i(s_k),\ k = 0..i-1 \\ \\ 1, & \text{otherwise} \end{cases} \tag{3.3}$$

This composition method strengths maxima changing from a lower to higher value. This way, the maxima remain visible as in MIP. The other tissue along the ray also contributes, but with subtler transparency as in DVR (see also Figure 3.7c).

MIDA and Multi-channel Data

Our application can handle up to four different volumes. Our volumetric data are scalar fields, thus we are able to use four components (RGBA) to store four different volumes. As we deal with multi-channel data, we adjusted the mono-channel approach of MIDA [12] for multi-channel data as follows. For color and opacity mixing of each channel, we used the composition scheme described by Kniss et al. [37]. Assuming we have n volumes $v_j(x),\ j = 1..n,\ x \in \mathbb{R}^3$ and n transfer functions assigning color and opacity to samples - $c_j(v_j(x))$ and $\alpha_j(v_j(x)),\ j = 1..n,\ x \in \mathbb{R}^3$ or simply c_j and α_j. We calculate the opacity of each sample $\mathfrak{A}_i$ as the sum of opacities of each channel α_{ji}. The color of each sample $\mathfrak{C}_i$ equals the sum of all colors at each channel c_{ji} weighted by opacities as denoted in Equation 3.4.

$$\mathfrak{A}_i = \sum_{j=1}^{n} \alpha_{ji} \quad \mathfrak{C}_i = \frac{\sum_{j=1}^{n} \alpha_{ji} c_{ji}}{\sum_{j=1}^{n} \alpha_{ji}} \tag{3.4}$$

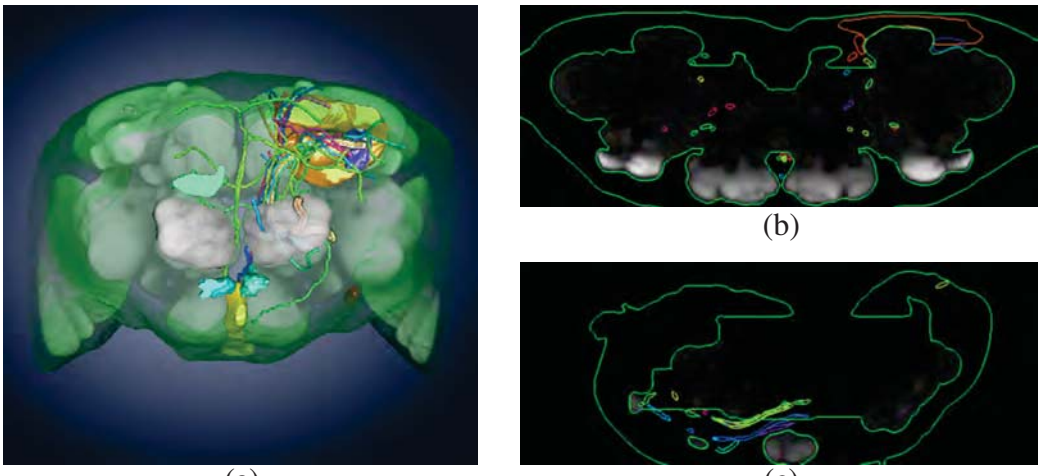
Even though the compositing equation (Equation 3.2) for multi-channel data is the same as for mono-channel data, the computation of parameters $\mathfrak{B}_i$ is more complex. For each channel j and each sample along the ray i, we calculate δ_{ji} as in Equation 3.6 where s_{ji} denotes the i-th sample of the channel j. We use the parameters δ_{ji} for further calculations of the modulation factors $\mathfrak{B}_i$ in Equation 3.6. The additional weighting of δ_{ji} by opacity suppresses contribution of invisible samples.

$$\delta_{ji} = \begin{cases} s_{ji} - max_{ji}(s_{jk}), & \text{if } s_{ji} > max_{ji}(s_{jk}),\ k = 0..i-1 \\ \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

$$\mathfrak{B}_i = 1 - max(\delta_{ji}\frac{\alpha_{ji}}{max(\alpha_{ji})}),\ j = 0..i-1 \tag{3.6}$$

Geometry Rendering and Enhancement

We use standard geometry rendering techniques for all non-volumetric structures. All of them except of neuronal projections are given as triangle meshes (see also



(a) (b) (c)

Figure 3.9: Contours in the slice view - 3D visualization providing the reference (a), slice views with contours of different structures(b),(c).

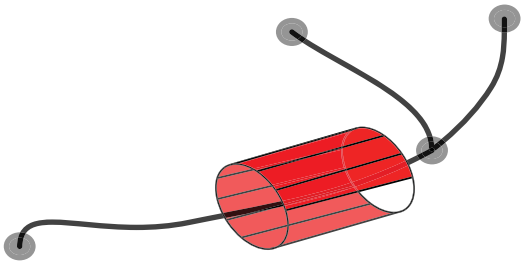


Figure 3.10: Tessellation of a skeleton graph representing a neuronal projection.

Table 3.4b). Geometry rendering has many advantages. First, the standard per-pixel phong shading, we implemented, is a realistic smooth surface representation. Second, this method is fast because it is hardware-accelerated. Third, surface representation with geometry allows easy rendering of outlines in the slice views as illustrated in Figure 3.9. However, the neuronal projections are originally stored as skeleton graphs. We created a tessellation using cylinder-like extrusion of the graph. This allowed us to generate cylindric representation of the neuronal projections as illustrated in Figure 3.10.

To improve the depth perception of numerous structures in the image, we enhance strong depth discontinuities as described by Luft et al. [42]. The technique bases on unsharp masking of the depth buffer. We compute the difference between the original depth buffer of the rendered scene and its low-pass-filtered variant. This operation corresponds to subtraction of the low frequencies. Thus, the difference buffer retains the high frequencies which correspond to the high discontinuities in the rendered scene. In this manner, we are able to find where to modulate the opacity of the context volume to make the high depth discontinuities shine through more.

3.3 Visual Queries

From Section 3.1 we understand the basics of visual queries. In this section, we exhaustively describe the algorithmic concept of visual queries - path, object and semantic queries, as implemented in our system. We describe in detail the interface - how a user defines each type of query. Path and object queries require

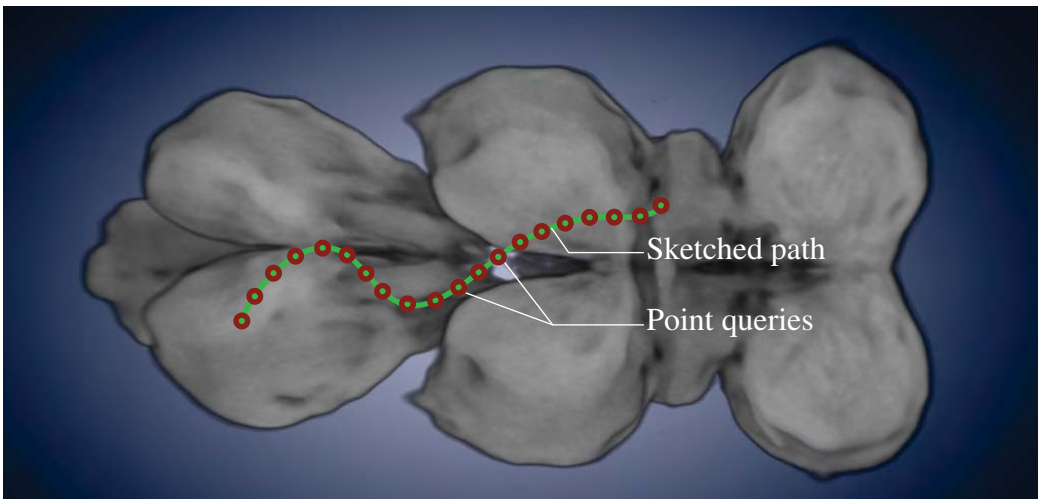


Figure 3.11: Sketching Interface - path decomposition into points, where point queries are applied.

precalculation of spatial indices. We especially focus on the algorithmic preprocessing of datasets, the efficient storage, and merging of indices when new data is inserted. Our system gives the user visual feedback about the spatial distribution of the the objects in a form of a hypertext label and a proximity cloud in the case of a path query.

Path Queries

Path queries are the most powerful searching method in our set of queries - they enable search for structures crossing a defined region. This regions is user-defined using an intuitive sketching interface. Defining a region of interest while sketching an arbitrary path is more flexible than selecting regions with some rectangular or elliptical markers. The query provides all structures crossing an area tight to the user-defined path. We restricted this area to the maximum radius of 40 voxels around the path. In practice, each path query falls apart into a set of point queries as we illustrated in Figure 3.11.

Individual point queries work as follows. For each point, we know the (x, y)-coordinates in the screen space and retrieve the z-coordinate from the depth buffer.

We transform the point from the screen space coordinates p_s into the volume space coordinates p_v. As the coordinates of p_v are rounded, p_v refers also to a voxel in the volume. With p_v as input, the point query delivers a list of pairs $< d, id >$ with distance d and structure identifier id, sorted by d. The distance d refers to the shortest distance from p_v to the surface of the structure id. When p_v lies inside of a structure, d is negative. The distance restriction of 40 voxels does not apply for negative distances, because we are interested in every object if p_v lies inside. Figure 3.12 shows the concept of the point query. In a preprocessing step, we prepare the lists for each point in the volume, i.e., a distance table. The distance table needs to be updated once we insert new data into the database. The size of the distance table file grows with the number of objects on the server. As our collaborators plan to insert new files, we expect a significant growth of the distance table file size. For these reasons, we keep this file in the off-core memory and not in the runtime memory (RAM). We use a precomputed look-up table (LUT) to efficiently retrieve rows of the distance table stored in the off-core memory. For each point in the volume p_v, the LUT provides, first, an offset of the row corresponding to p_v in the distance table. Second, the LUT provides the count of the pairs $< d, id >$ in each row. The size of the LUT is constant and independent from the number of structures, so we keep it in the runtime memory. In Figure 3.13, we demonstrate the concept of the distance table and LUT and their use for a query on a 2D example. In the example, we restricted the distances d to maximum of $d_{max} = 3$ voxels. In Figure 3.13a, we investigate every point, determine the sorted list of pairs $< d, id >$ and create the LUT. In Figure 3.13b, the user selects a point. We retrieve the offset and count from the LUT. This is the key informa-

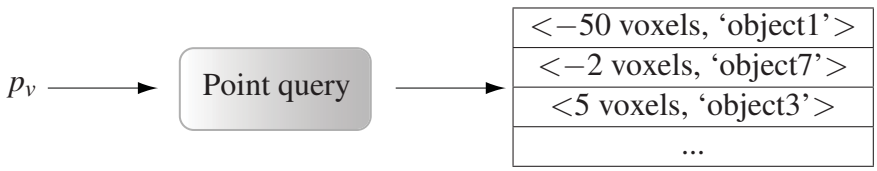
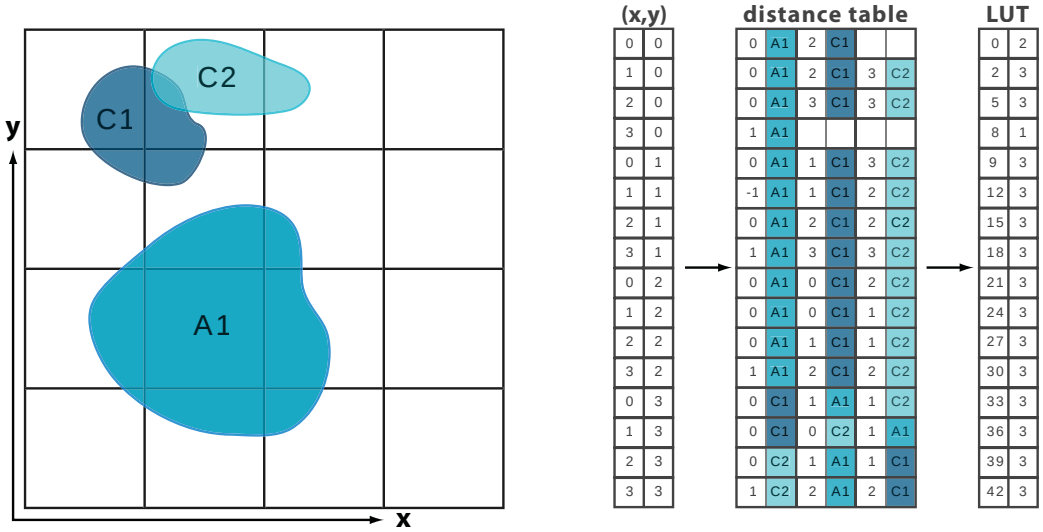
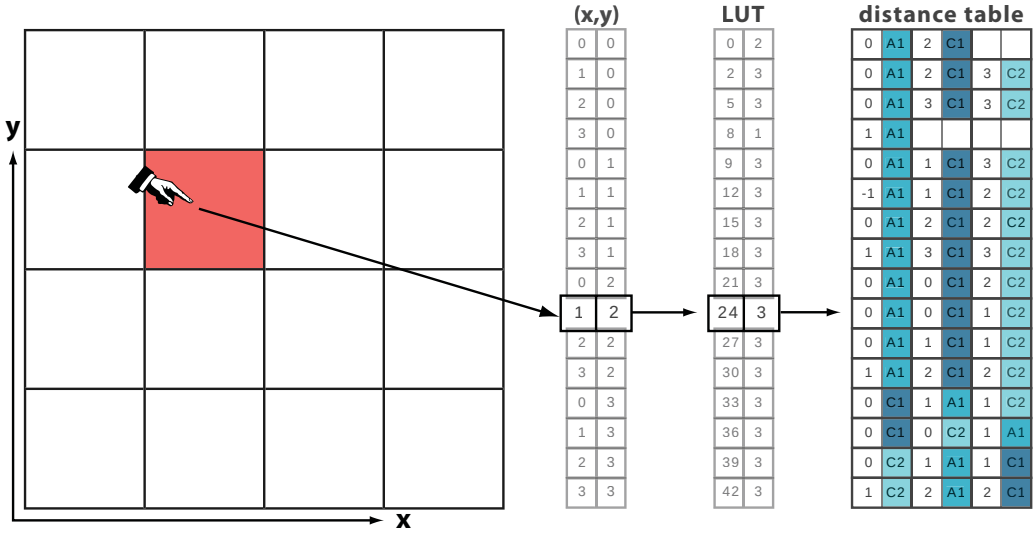


Figure 3.12: Scheme showing the concept of a point query. The query takes the volume coordinates of the point p_v and outputs the sorted list of the $< d, id >$-pairs.



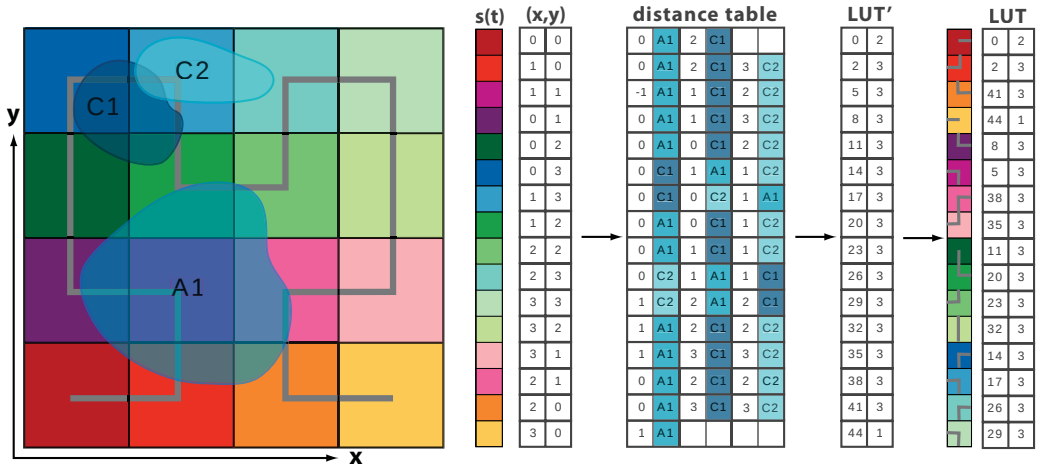
(a)



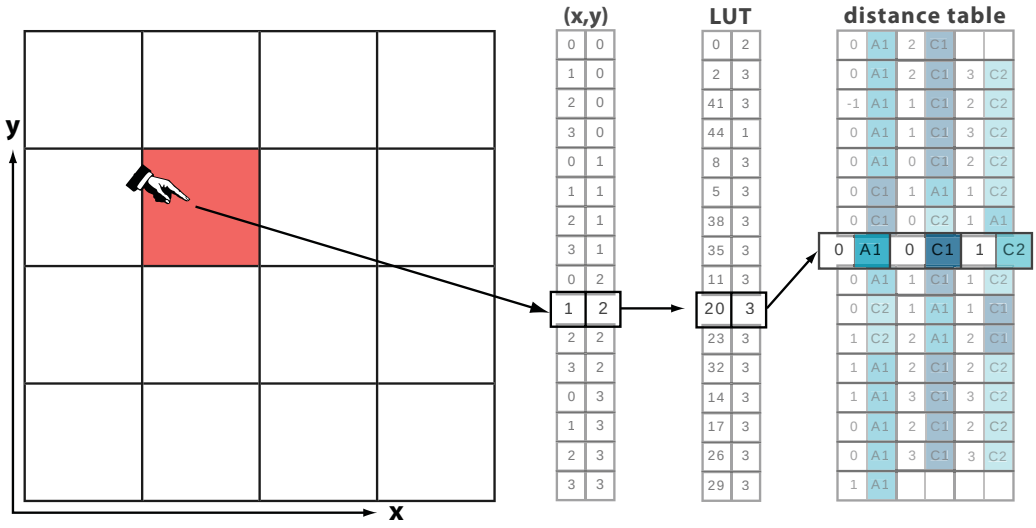
(b)

Figure 3.13: Calculation of the distance table and LUT in 2D (a) and the use of the tables for a point query (b).

tion to read the row corresponding to the list of pairs $< d, id >$ we created in the preprocessing step. The disk read head skips $offset = 24$ pairs of the file and reads subsequent $count = 3$ pairs. The distance table and the LUT are serialized, i.e., by their ascending linear indirections $lin(\vec{p})$ as defined in Equation 3.7 with



(a)



(b)

Figure 3.14: Calculation of the distance table and LUT using 2D Hilbert scan (a) and the use of the tables for a point query (b).

$\hat{x} \times \hat{y} \times \hat{z} = volumesize$. Under indirections we understand an ordering pattern of voxels.

$$lin(\vec{p}) = p_x + p_y\hat{x} + p_z\hat{x}\hat{y} \tag{3.7}$$

While drawing a path, we call point queries for points located close to each other. The point queries fetch the respective rows from the distance table. To improve

the performance, we used neighborhood-preserving storage pattern of rows using a 3D Hilbert scan. Thus, we store the rows of the distance table using 3D Hilbert scan indirections as we demonstrate in Figure 3.14. The $< d, id >$-pairs inside the rows are not permuted. The final LUT must be accessible using linear indirections. It contains the offset and count information with relation to the permuted distance table. Comparing Figures 3.13 and 3.14, we see that the fetch of a row from the distance table has the same logic if we use the permutation using 3D Hilbert scan indirections. Using our sophisticated storage pattern enables us to keep the easy look-up in the LUT and to gain performance while reading from the distance table file.
The results of all point queries along the sketched path are merged together and presented to the user in an in-window hypertext label. If there are more $< d, id >$ pairs for one object id, we keep only the pair with lowest distance d.

Preprocessing Pipeline

The preprocessing pipeline consists of six stages (see also Figure 3.15). In this section, we describe each of them.

1. The structures in our database are represented as geometry or skeleton graphs. In the first step, we gradually create distance field volumes for each structure. As our files are provided in Amira's native file format, we generate the distance field volumes using Amira [1]. Amira features many

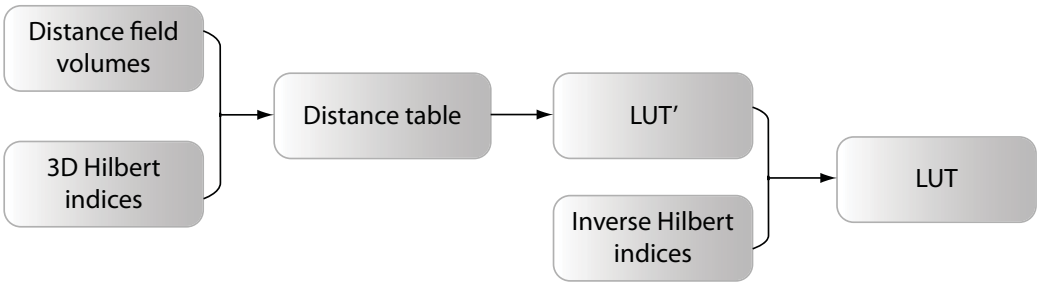


Figure 3.15: Preprocessing pipeline.

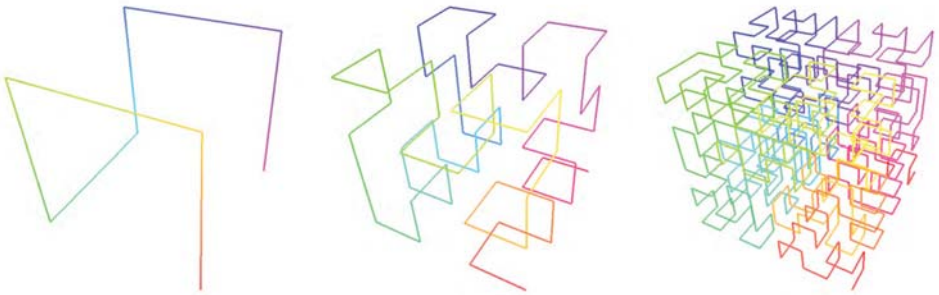
(a) (b) (c)

Figure 3.16: Slice of the distance field volume. Positive distances (outside) (a) and negative distances (inside) (b) from the surface area. The slice combined with a 3D rendering of the respective structure (c).

image processing tools and supports batch processing. For skeleton graphs, we are able to extrude the graph with some radius and create the surface geometry representation. For geometry, we compute signed distance fields directly using image processing tools of Amira. We choose signed distance fields using chamfer metric. Each voxel of the distance field volume stores the distance to the surface in voxels. Signed distance fields enable us to distinguish between the inside and the outside regions of each structure. The negative distances represent the inside area. Figure 3.16 illustrates slices of the distance field volume. We record the mentioned steps and automate the processing using batch files. This enables us to generate a distance field volumes for each structure without our intervention.

2. We generate the 3D Hilbert scan for indirection indices. We scan the cuboid region with the size of the volume $\hat{x} \times \hat{y} \times \hat{z}$. Figure 3.17 illustrates a 3D Hilbert scan of differently sized regions. In Table 3.1, we provide the indirection indices corresponding to the scan shown in Figure 3.17a.

3. We prepare a distance table with $n = \hat{x} \times \hat{y} \times \hat{z}$ empty rows and process the distance field volumes voxel-by-voxel. For each voxel v with position $\vec{v} = (v_x, v_y, v_z)^T$, we investigate the distance d which is the voxel value. If $d \leq 40$, we insert the pair $< d, id >$ into the r-th row of the distance table. In



(a) (b) (c)

Figure 3.17: 3D Hilbert scans of different size - $2 \times 2 \times 2$ voxels (a), $4 \times 4 \times 4$ (b) voxels and $8 \times 8 \times 8$ voxels (c).

Table 3.1: Indirections for 3D Hilbert scan of size $2 \times 2 \times 2$

3D Hilbert scan $s(t)$	Volume coordinates $\vec{p_v}^T$	linear indirection $lin(\vec{p_v})$
s(0)	$(0,0,0)$	0
s(1)	$(0,1,0)$	2
s(2)	$(0,1,1)$	6
s(3)	$(0,0,1)$	4
s(4)	$(1,0,1)$	5
s(5)	$(1,1,1)$	7
s(6)	$(1,1,0)$	3
s(7)	$(1,0,0)$	1

this case, we use simple linear indirection of the volume where $r = lin(\vec{v})$. The *id* is the identifier of the currently processed distance field volume. After the last distance field volume has been processed, we sort the rows of the distance table ascendingly with respect to the distance d of the pairs $< d, id >$. We write the entire rows of the distance table on the disk obeying 3D Hilbert scan indirections. For example, the rows of a distance table of a $2 \times 2 \times 2$ volume would be stored in a permuted order $(0,2,6,4,5,7,3,1)$ as indicated in Table 3.1.

4. We generate the look-up table LUT' corresponding to the permuted distance table. Consequently, the LUT' is also permuted and cannot be accessed via simple linear indirections. However, we will need the permuted LUT' for later merging of tables discussed in Section 3.3. Thus, we write this table

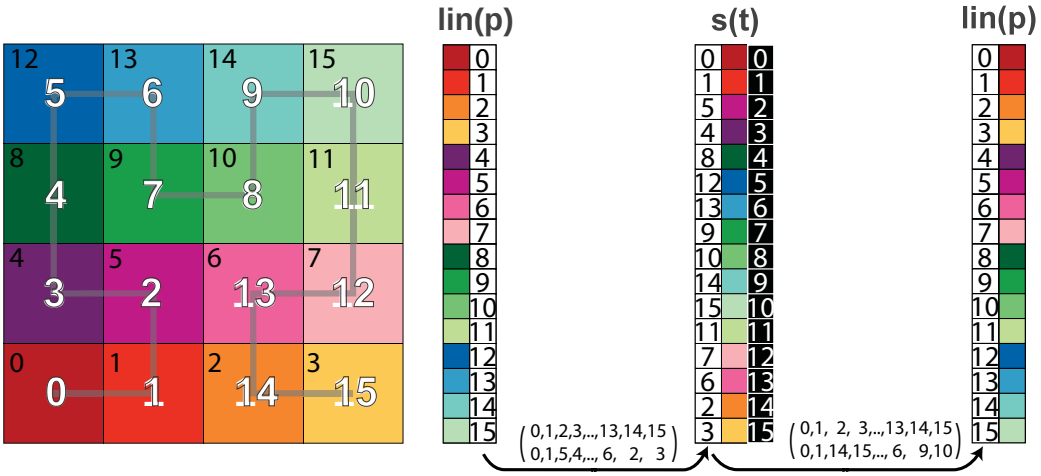


Figure 3.18: Indirections.

as temporary data.

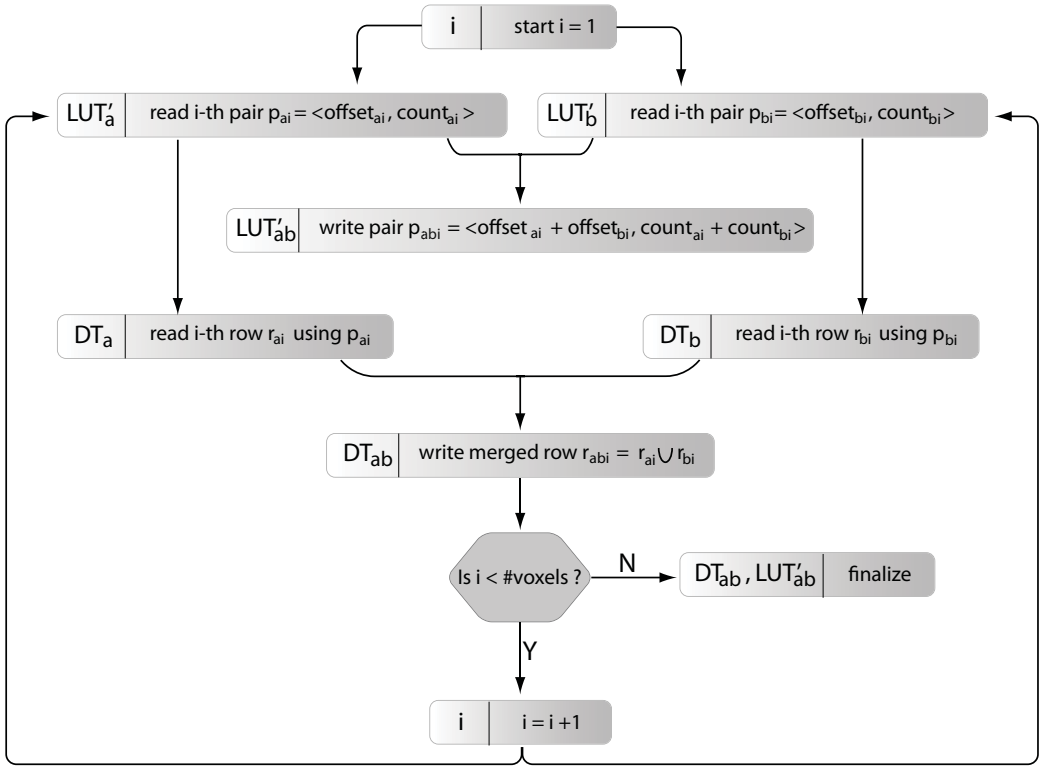
5. The application must access the LUT using linear indirections which is not possible to do with a permuted LUT'. Thus, we need to permute it back. We need the inverse 3D Hilbert scan and inverse indirections to perform the inverse permutation. The inverse indirections map the 3D Hilbert scan back to the linear indirections. In Figure 3.18 we illustrate the mapping from linear indirections to 2D Hilbert indirections and mapping from 2D Hilbert indirections back to the linear indirections. Both mappings are represented as permutations.

6. We use the inverse indirection to permute the LUT' to obtain a valid look-up table LUT which can be accessed using simple linear indirections.

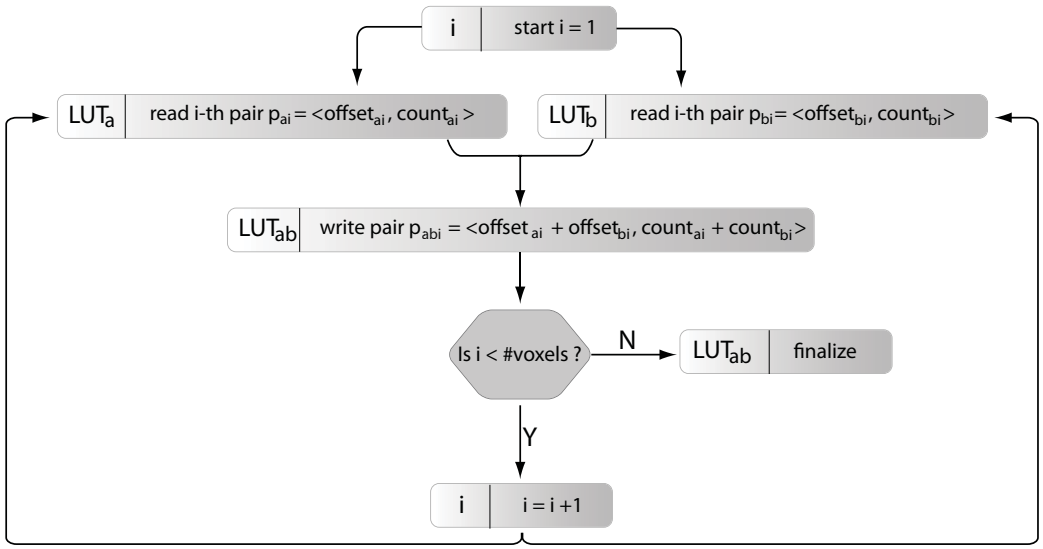
After the last stage of the preprocessing procedure, we have generated two tables - LUT and the distance table, ready to be used by the application. We keep in mind, that we also have the temporary permuted LUT' necessary for merging.

Merging of Tables

Expecting a significant growth of the database, we designed an efficient algorithm for merging distance tables and LUTs. Furthermore, the preprocessing calculation



(a)



(b)

Figure 3.19: Schemes representing the merging of tables. Flowchart of merging two distance tables DT_a and DT_b. We store also LUT'_{ab} because in the future we might need to merge DT_{ab} with some other distance table (a). Flowchart of merging two look-up tables LUT_a and LUT_b (b).

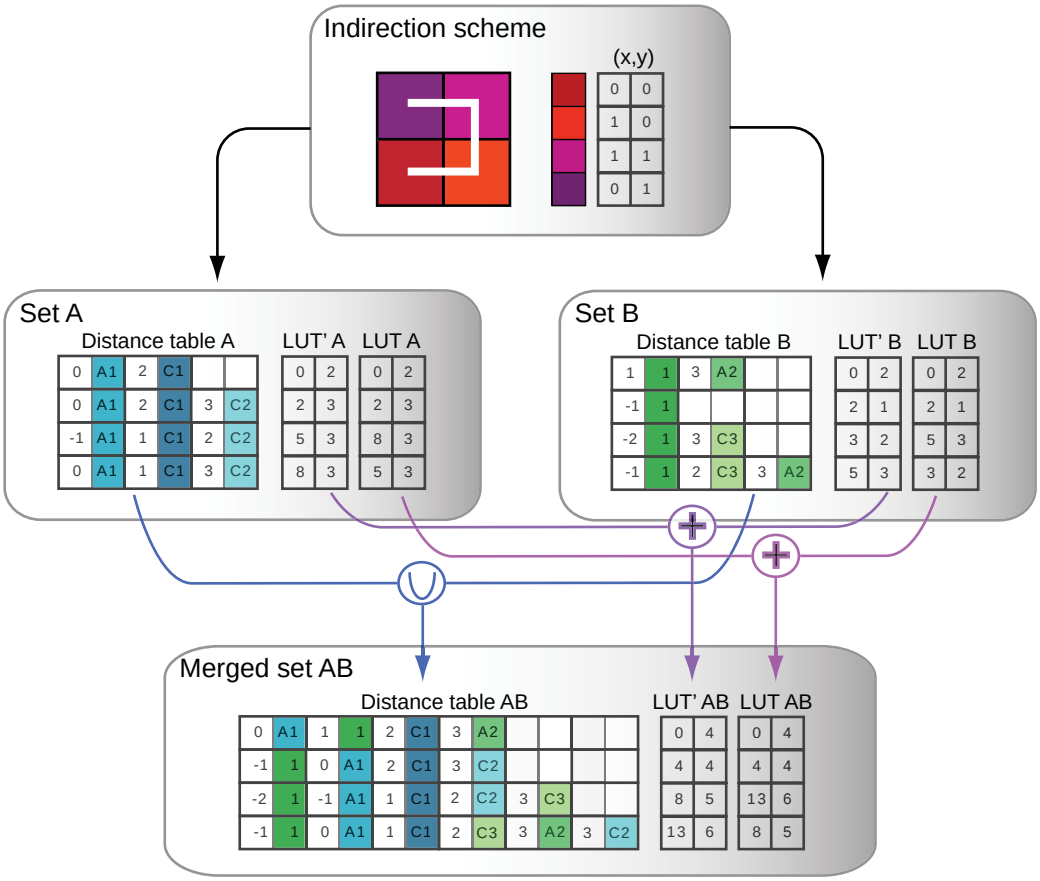


Figure 3.20: Example of merging using Hilbert indirection. We use element-wise addition to concatenate look-up tables and union operation to concatenate distance tables.

requires a lot of RAM. However, the amount of available RAM is limited, so we cannot guarantee that all data can be successfully processed at once. We overcome the RAM limitation as follows. We split the data into n disjoint groups, generate n distance tables and LUTs, and merge them pairwise. Finally, we provide a single distance table and LUT representing all structures. The major requirement for the merging algorithm are handling of large data and requiring as low amount of RAM.

Figure 3.19 illustrates the flowchart of merging two distance tables DT_a and DT_b and two look-up tables LUT_a and LUT_b. Rows of the distance tables DT_a and DT_b

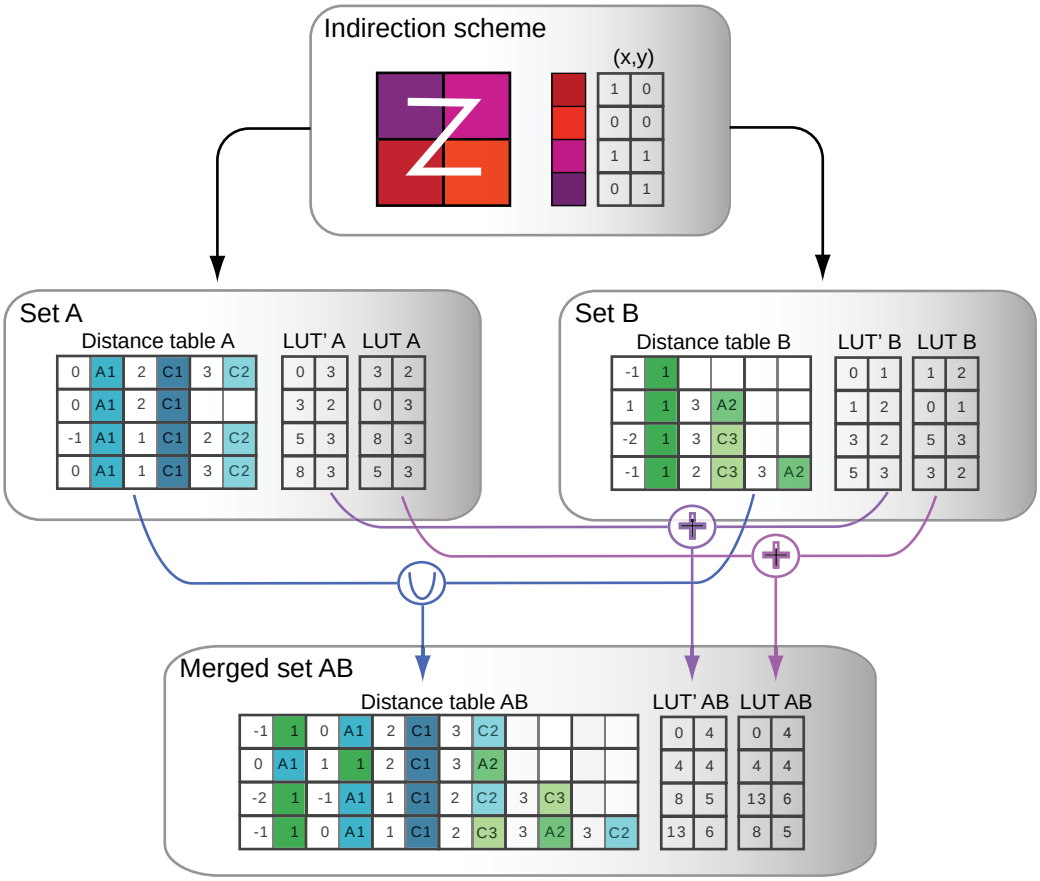
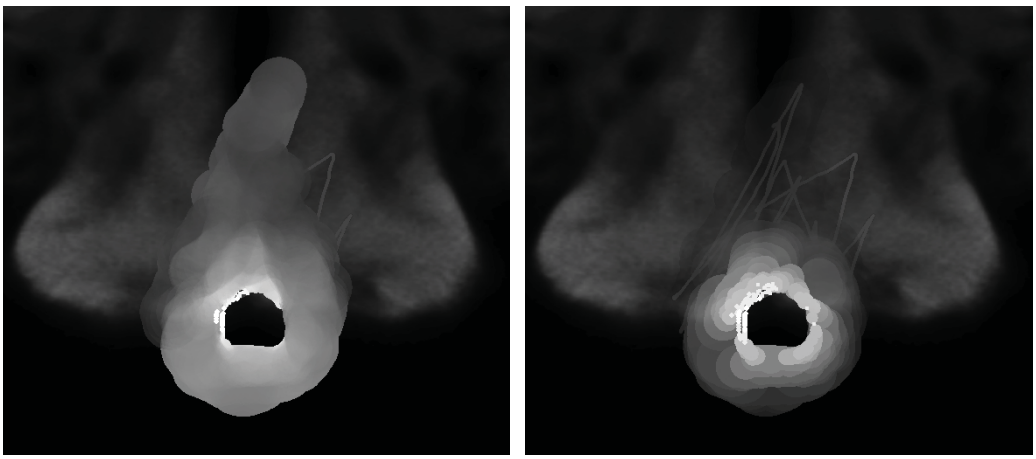


Figure 3.21: Example of merging using Z-order indirections. We use element-wise addition to concatenate look-up tables and union operation to concatenate distance tables.

can be concatenated using a union operation because they refer to two disjoint set of objects. The number of pairs in the new row equals the sum of the pairs in the corresponding rows in DT_a and DT_b. This property makes merging of look-up tables (LUT' and LUT) easy. We simply add $offset$s and $count$s of two old look-up tables and get a new valid look-up table. In Figures 3.20 and 3.21, we give two examples of pairwise merging of tables. We show that addition of the look-up tables and union of distance tables works also for different indirections.



(a) (b)

Figure 3.22: Blending of proximity colds - average blending (a) and minimum blending (b).

Advantages of the Storage Method

In our approach, we use a precomputed distance table to determine closest objects for each point which has the following advantages. We precompute the table only once in a preprocessing step and merge the tables when new data are inserted. At runtime, we perform only one look-up per point of the sketched path. Moreover, the distance table stores the $< d, id >$-pairs in a sorted order, so we perform no sorting and no distance calculation at runtime. The knowledge of the point distance d is essential as the key for sorting and for approximation of the real distance. The performance of the application also benefits from the locality-preserving storage pattern applied to the distance table using the 3D Hilbert scan indirections.

We compared our approach to Fibonacci coding of the spatial data which bases on the Zeckendorf theorem [66]. The theorem states that every positive integer n can be uniquely represented as a sum of nonconsecutive Fibonacci numbers as denoted in Equation 3.8.

$$n = \sum_{k=0}^{L} \varepsilon_k F_k \mid \varepsilon_k \varepsilon_{k+1} = 0, e_k \in \{0,1\} \tag{3.8}$$

We assign every object in our database an $id = F_k$. When objects i and j intersect, and $id_i = F_k$, then $id_j \neq F_{k+i}$. The voxels store n which is the sum of the ids of all incident objects. This refrains us from storing the whole list of ids for each voxel. However, the Fibonacci numbers increase rapidly - F_{48} exceeds the range of the 32-bit unsigned integer. As we foresee the number of objects to index $L > 300$, this method becomes inconceivable regarding storage. Furthermore, this method requires decomposition of the indices into F_ks and calculation of the distances at runtime which is a disadvantage.

In comparison to our method, approaches such as space-partitioning, e.g., kd-trees or octrees have the following disadvantages. Many neuronal structures in our database have intersections. Space-partitioning methods would represent them as single tree-nodes. Consequently, for each node we must foresee saving a list of id-pairs of all incident objects. In contrast, the distance table stores the lists of $< d, id >$-pairs for every point in the volume. The storage requirements of a kd-tree or of an octree are inferior to the storage requirements of our distance table because the tree-nodes cover more than one voxel. If the tree-nodes store $< d, id >$-pairs, the storage requirements are equal to our approach, so the space-partitioning method brings no advantage. Assuming that the tree-nodes do not implicitly store the distance information to the incident objects, the application must determine them at runtime while performing computational costly operations. Therefore, the performance of the space-partitioning methods is inferior to the performance of our method. As one of our goals is to implement an interactive tool, the performance is the primary benchmark and storage the secondary. Furthermore, the implementation of our method is intuitive and simpler as the implementation of a space-partitioning method. Also, merging of two space-partitioning trees is more complex and costly then merging of two distance tables as proposed in this section.

Proximity Clouds

Giving users visual feedback while he or she performs tasks is important. Our sketching interface offers immediate visual feedback in fashion of a proximity cloud which forms around the path. The cloud enables the user to identify the

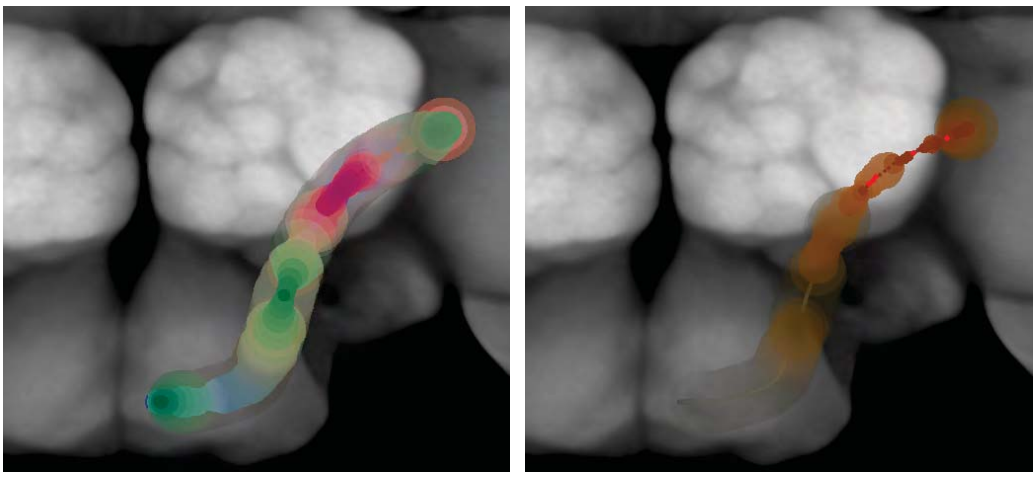
spatial distribution of structures in the region of search. By default, the radius of the region is set to 0 voxels, i.e., we search only structures which intersect or contain the path. User can lower or increase this value with a slider. In practice, the underlying point queries always return the whole row of the distance table, i.e., all structures with distance d up to 40 voxels. However, each row is sorted by distances, so we can easily cut-off a part of it and retain only those structures whose distances are less than some threshold.

The rendering of the cloud around a point p_c works as follows. For each object i in the point query result, we render a semi-transparent circle around p_c with radius equal to the distance to the object i. Circles with low radius indicate close objects and are more important, so we render them with higher opacity. Rendering and blending circles for every object at every point along the path forms a cloud of variable opacity. The opacity accumulates more in areas where we rendered more circles over each other, thus the cloud looks more dense around points with many objects in the vicinity.

Blending We experimented with different blending options (see also Figure 3.22). In Figure 3.22a we show circles using *average* blending modus. In the first pass, we render to texture T - we write $(d, 1.0, 0.0, 0.0)$ to the fragment color and use *ADD* as hardware blending modus. In the second pass, we use T to fetch the color *rgba*. The ratio $\frac{r}{g}$ signifies the average distance and we use it to calculate opacity α and grayvalue γ of the fragment as in Equation 3.9. The gray values and opacities have values in the range of the interval $[0.2, 0.8]$.

$$\alpha = \gamma = 0.8(1 - \frac{r}{g}) + 0.2 \tag{3.9}$$

However, proximity clouds rendered in *average* blending modus lead to significant loss of information which worsens the orientation. On the other hand, *minimum* blending is more comprehensive even if the clouds exhibit discontinuities in color. Furthermore, we are able to render the cloud in a single pass. We assign circles with lower normalized distance $\frac{d}{40}$ opacity as in Equation 3.10 and use *MAX* hardware blending modus. Consequently,



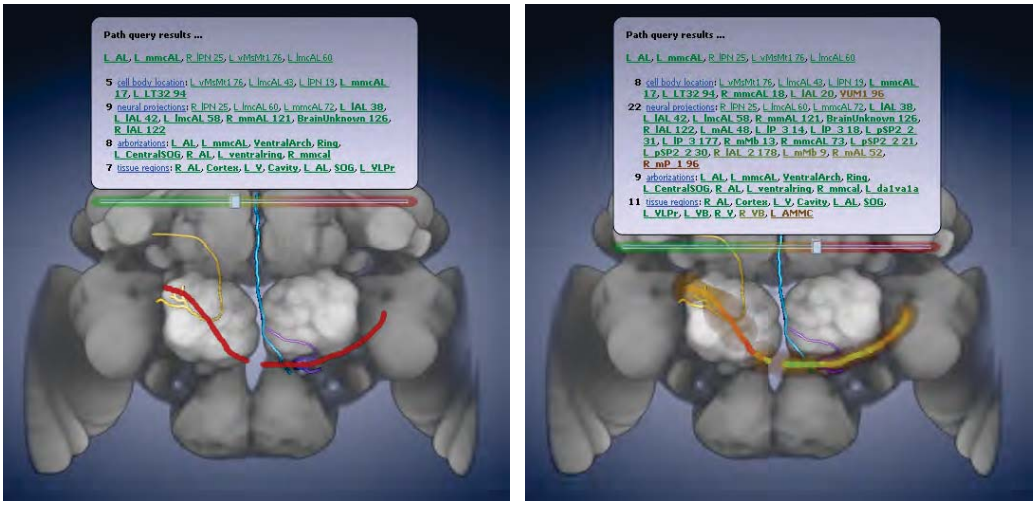
(a) (b)

Figure 3.23: Color coding of proximity clouds. Mixing of all colors (a). Only the selected type, i.e., arborization, remains highlighted (b).

fragments with lower input distance always overwrite higher values in the frame buffer. We keep the *min* blending modus in the remainder of this work.

$$\alpha = \gamma = 0.8(1 - \frac{d}{40}) + 0.2 \tag{3.10}$$

Color-coding Our visual queries handle four different subgroups of structures - cell bodies, neuronal projections, arborizations and template regions. However, gray-scaled proximity clouds does not allow us to distinguish spatial distribution of individual subgroups. We experimented with color-coding to delimit between spatial locations of different subgroups as in Figure 3.23. We chose matching but distinguishable color groups - orange, purple, blue and green using on line tool Colorbrewer [26]. We assign a color to each subgroup - to arborizations orange, to cell bodies green, to neuronal projections purple, and to template regions blue color. The color coding supports two modi. In the first mode, all subgroups are active and clouds of different colors are blended together as in Figure 3.23a. In the second mode, exactly one subgroup is active, e.g., arborizations. Consequently, the cloud con-



(a) (b)

Figure 3.24: Results of point query in a hypertext label for different search radii. The green part of the slider indicates negative radii and thus, only green hypertext links are displayed. These links refer to structures which cross the sketched path (a). The red part of the slider refers to positive search radii and the labels displays red hyperlinks. These refer to structures which do not intersect the path but lie in the proximity (b).

tains only tones of orange. Furthermore, we marked segments of the path which enter inner regions of the selected type of structures with red spots as in Figure 3.23b.

However, different colors make the point cloud unintuitive. Thus, we keep one color and retain the possibility to activate a single subgroup. In this case, the proximity cloud displays the spatial distribution only for the selected subgroup.

Hypertext Labels

We use hypertext labels titled “Path query results” to present query results to the user. They pop up in-window which enables the user to keep focus on the visualization. The label appears in the corner of the window when user clicks with the mouse button and starts sketching the path. We attached a color-coded

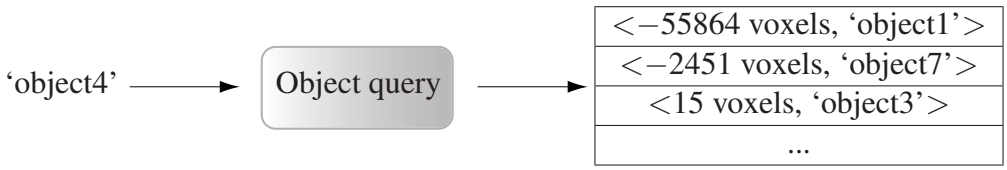


Figure 3.25: Scheme showing the concept of an object query. The query takes the object *id* of a selected object and outputs the sorted list of the $< d, id >$-pairs.

slider to the label which enables control over the search radius around the path. The radius is by default set to 0 voxels. The label displays hyperlinks to whole subgroups, and to individual structures within each subgroup, and also the number of structures found for each subgroup. The hyperlinks use the same color-coding scheme as the slider which enables the user to quickly associate the structure some degree of proximity. This information is updated while the user keeps sketching. When the user releases the mouse button, the path is finished, and the label moves to the center of the window. The user can adjust the search radius and inspect the results (see also Figure 3.24). Eventually, he or she clicks on hyperlinks to load structures individually or a whole subgroup of structures at once.

Object Queries

In addition to path queries related to spatial distribution of objects around points in the volume, our set includes also object queries. As opposed to path queries, they are related to spatial distribution of objects around a selected object of interest. In practice, this query provides a list of closest objects and their distances in voxels. If two objects intersect, we provide the volume of their intersection instead of the distance.

Object queries work as follows. We provide an *id* of the selected objects as input and the query returns a list of object sorted by the distance (see also scheme in Figure 3.25). The negative distances indicate intersection volumes. Similarly as for the path queries, we create a table, i.e., object table, in a preprocessing step.

Preprocessing

In practice, we start with an $n \times n$-matrix M. M_{ij} stores the minimal distance between objects i and j, or eventually their intersection volume. We initialize all matrix elements M_{ij} to the highest distance which can occur, i.e., $d_{max} = \sqrt{\hat{x}^2 + \hat{y}^2 + \hat{z}^2}$ with $\hat{x}\hat{y}\hat{z} =$ volumesize. We inspect each pair of distance field volumes described in Section 3.3 voxel-by-voxel. The matrix M is symmetric, so once we inspected the pair of distance fields ij, we do not inspect the pair ji. We gradually investigate voxel pairs v_i^p and v_j^p, i.e., voxels of the object i and j at the position p in the volume. We distinguish following cases (see also Figure 3.26).

- The value of $v_i^p = 0$ so v_i^p belongs to the surface of the object i. We modify the matrix M as follows. If $v_j^p > 0$ and $v_j^p < M_{ij}$, we replace the value of M_{ij} by v_j^p. For example, the sampling position A in Figure 3.26a fulfills this condition - it lies on the surface of the object i, but outside of the object j. If $v_j^p \leq 0$ then p belongs to the intersection volume. For example, sampling positions B and C in Figure 3.26a comply with this condition, because they both belong to the surface of the object i and lie either on the surface or inside of the object j. In this case, we set M_{ij} to -1 if $M_{ij} \geq 0$ or decrement M_{ij} otherwise.

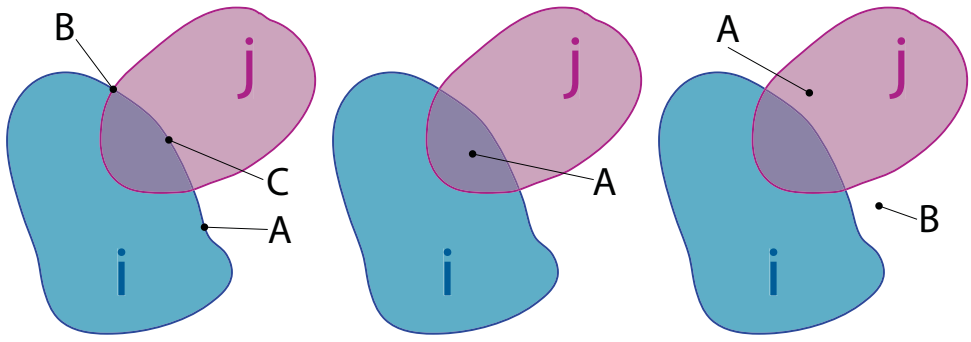


Figure 3.26: Investigation of voxel pairs for the objects i and j - on the surface of the object i (a), in the intersection of the objects i and j (b), and outside of the object i (c).

- The value of $v_i^p < 0$, and $v_j^p < 0$. Both voxels belong to the intersection of the objects i and j, i.e., sampling position A in Figure 3.26b. We modify the matrix M as follows. If $M_{ij} \geq 0$ then we set M_{ij} to -1, else we decrement M_{ij}.

- The value of $v_i^p > 0$. Thus, v_i^p belongs nor to the surface of the object i, nor to its inside area, as it is the case of the sampling positions A and B in Figure 3.26c. In this case, we do not modify the matrix M.

We fill elements of $\hat{M}_{ij}$ above the diagonal ($i > j$) and use the symmetry to copy $\hat{M}_{ij}$ to elements below the diagonal $\check{M}_{ji}$. From each row of the matrix M, we generate a list L consisting of $< d, id >$-pairs where $d = M_{ij}$ and $id = j$. We are not interested in elements M_{ij} such that $i = j$, so we exclude them from L. In the final step, we sort L by increasing distances d. The finalized object table stores minimal distances between objects, or the total volume of their intersection. In the case that objects have multiple disconnected intersections, we store the sum of the volumes of their partial intersections.

Merging

The object table needs to be updated only when new objects are inserted into the database. Let A be the set of old objects and B the set of inserted objects. Then $A \bigcap B = \emptyset$, $|A| = n$, and $|B| = m$. We proceed as follows.

- We initialize an $(n+m) \times (n+m)$-matrix M and use the old object table to fill the maximum of elements.

- For all objects $\in B$, we compute a $m \times m$-matrix M' of distances and intersection volumes. M'_{ij} stores a distance or an intersection volume between the objects i and j, such that $i, j \in B$. We use M' to fill the additional elements of M.

- For all objects $\in B$, we compute an $m \times n$-matrix M''. Matrix elements M''_{ij} store distances or intersection volumes between objects i and j, such that $i \in B$ and $j \in A$. We use M'' to fill the remaining elements of M and proceed to sorting as in Section 3.3.

Hypertext Labels

Similarly to path queries, we also use hypertext labels to present the results of object queries. When a user selects an object with a mouse click, a label titled with the name of the selected object appears in-window. It can occur, that some objects are occluded by other objects. In this case, the next click selects the next object below. The selected object is highlighted with a contour. The pop-up label contains object query results and semantic information retrieved by a semantic query.

The object query results display closest objects and divides them into subgroups, i.e., arborizations, cell bodies, neural projections and template regions. Only objects closer than some cut-off distance ε are listed. User can set ε with a color-coded slider. The colors of the slider help to associate objects which appear in hypertext label with their degree of proximity. Objects with intersections appear

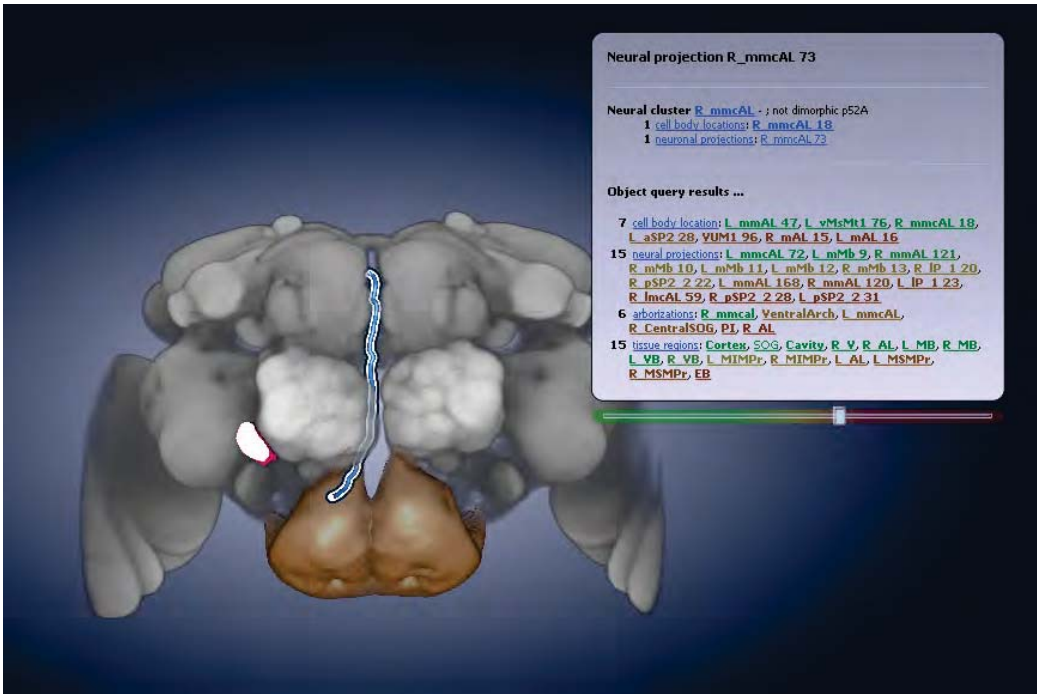
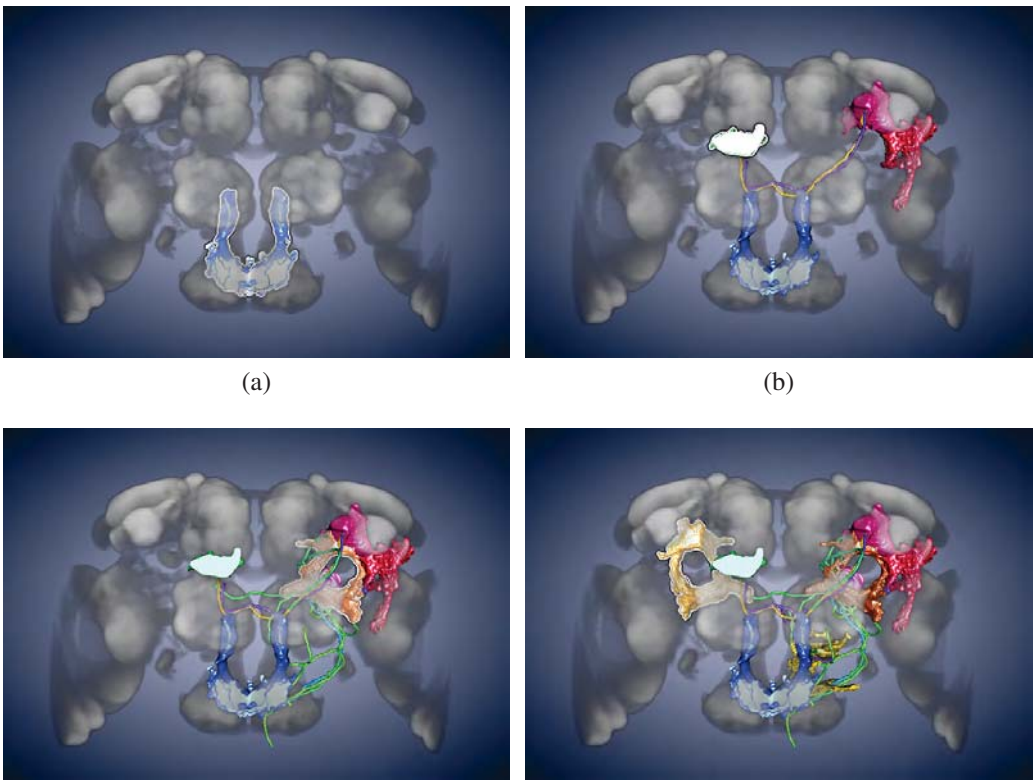


Figure 3.27: A hypertext label displaying object and semantic query results for a neuronal projection *R_mmcAL 73*.



(a) (b)

(c) (d)

Figure 3.28: Browsing through data using semantic query starting with an arborization (a). In the second step, an associated cluster is loaded (b). Results after the third (c) and the fourth iteration (d).

green and distant objects appear dark red. In Figure 3.27, we show a screenshot with a selected neuronal projection and a corresponding hypertext label.

Semantic Queries

Our collaborators annotated some objects in the database and associated them to clusters. Semantic queries retrieve this information and display it together with the object query results in the hypertext label as denoted in Figure 3.27. Associated structures are listed as hyperlinks. Activating them triggers loading of the respective objects. This is a powerful method of browsing through the data. Structures such as arborizations represent synapses which are often part of more

clusters. Results of a semantic query allow the user to load them and explore the further connections. In this manner, neuroscientist can follow neural paths. In Figure 3.28, we show gradual exploration of associated neural clusters.

CHAPTER

4 Implementation

If I have ever made any valuable discoveries, it has been owing more to patient attention, than to any other talent.

Sir Isaac Newton

OUR APPLICATION is a proof-of-concept of the methods presented in this thesis. The whole system is implemented in C++, and uses OpenGL with GLSL vertex and fragment programs. Furthermore, it uses Qt libraries for the user interface and PostgreSQL for the database. It runs on Windows Vista/XP machines equipped with an Intel processor operating at least at 2 GHz, and with at least 1 GB of available RAM. The graphics hardware must support Shader Model 3.0 or higher. The system has a very flexible architecture - the application core, the user interface and plugins are not coupled. This enabled us to rapidly implement and integrate new functionality. In this chapter, we refer to some implementation specific details. In Section 4.1 we discuss the scripting in Amira [1] and generation of distance fields. Section 4.2 focuses on methods we used to create Hilbert indirections. Section 4.3 describes data types and compression of the preprocessed tables, and the object-id generation.

4.1 Amira Workflow

Workflow in Amira is based on creating networks. Data and functional modules are represented as entities. The user places them with drag-and-drop and joins them to define a network. Functional modules have adjustable parameters. These can be set in a dialog which appears when the user clicks on the respective module. Figure 4.1 shows an example of a network in Amira.

Amira allows to save the networks as text files. They include all functional calls, set-up of modules and their properties. Thus, they are an excellent start-up for writing scripts. We created a workflow network for one dataset and used it as a basis for our script.

Neuronal projections in our data collection are stored as skeleton graphs. Originally, Amira does not support this format, and requires an installation of an additional plugin [62]. However, this tool is supported only by the 4-th version of Amira which does not allow to connect skeleton graphs with any of the computation modules we needed. For this reason, we exported the skeleton graphs into the neuron format *.hoc* and proceeded with the 5-th version of Amira.

All objects in the database are registered to volumes of size $768 \times 768 \times 165$ voxels. However, we downsampled our distance field volumes to the size $384 \times 384 \times 82$ voxels. The goal of the downsampling operation is to decrease the requirements on RAM while precomputing the distance table. Reduction by a factor of $f_{reduce} = 2$ still allows us to determine the distances in the volume with a sufficient precision of ± 1 voxel. Our distance field volumes store the surface distances in voxels corresponding to the original volumes, but with lower precision. However,

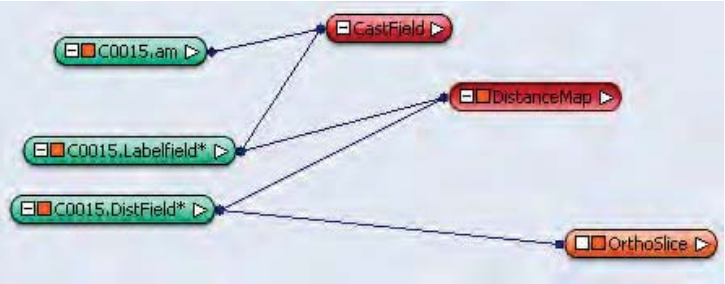
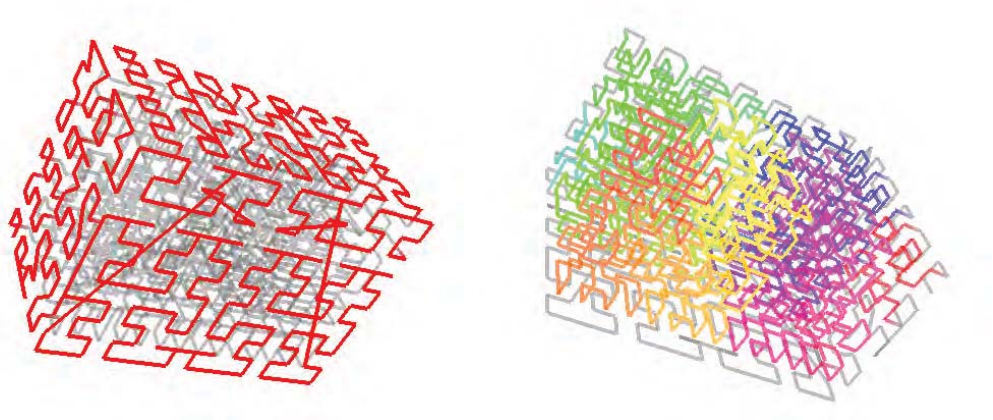


Figure 4.1: A network in Amira.

we must consider that we downsampled the distance fields while calculating the intersection volumes as described in Section 3.3. The volume calculation is incremental, i.e., we count intersection voxels in downsampled volumes. One voxel in the downsampled volume corresponds to $(f_{reduce})^3 = 8$ voxels in the original volume. Consequently, we multiplied resulting intersection volumes by a factor of $(f_{reduce})^3 = 8$.

4.2 Generation of Hilbert Indirections

For computation of the 3D Hilbert scan we use a recursive implementation of an L-System. Our volumes have sizes of an arbitrary cuboid region $(\hat{x}, \hat{y}, \hat{z})$. We compute the Hilbert scan for a cube of the size next power of two for $max(\hat{x}, \hat{y}, \hat{z})$ and clip the unnecessary vertices. Consequently, the spatial proximity breaks for vertices which lie on the surface of the cuboid region. However, this does not influence the performance of our tool. The cuboid region corresponds to the volume and we barely access voxels located on the surface of the volume's bounding box. Therefore, our clipping method of scanning arbitrary cuboid regions is sufficient for our system and thus, we do not need to implement the scanning algorithm



(a) (b)

Figure 4.2: 3D Hilbert scan of an cuboid region with the size $(9 \times 11 \times 15)$. Region where the Hilbert scan is broken is shown in red (a). The Hilbert scan remains correct in the inside of the volume which is shown in color (b).

introduced by Zhang et al. [68].

4.3 Object-ID Generation and Data types

In the preprocessing step, we generated two tables - a distance table storing point distances and an object table storing object distances. In practice, they store $< d, id >$-pairs with different interpretation. In both tables, the distance $d \in (-\infty, 40]$. The size of the distance table is expected to be large, so we must carefully chose the data type for the distance d and id. We used *unsigned short* representation for both d and id for two reasons. First, even if d can be negative, we can easily pack and unpack as denoted in Equation 4.1. Second, if all entries in the distance table have the same data type, we are able to read a large chunk from the disk at once. In many cases, one row of the distance table consists of many $< d, id >$-pairs. Using the same data type for d and id enables us to read the whole row at once.

$$d_{packed} = -(d_{unpacked} - 40), \quad d_{unpacked} \in (-\infty, 40], \; d_{packed} \in [0, \infty) \tag{4.1}$$

We dispose of seven different types of data (see also Table 1.1). Each file is well-defined with a number n within its subgroup. Thus, to composite an overall id, we need to consider both, subgroup and the number n. We used 8 most significant bits of the *unsigned short* to code the subgroup and the remaining bits to code the number n.

CHAPTER

5 Results

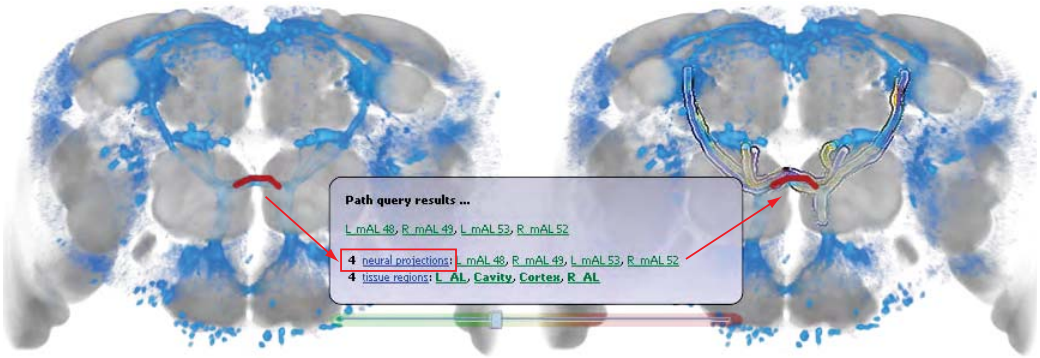
All you need is faith and trust...
and a little bit of pixie dust!

Peter Pan

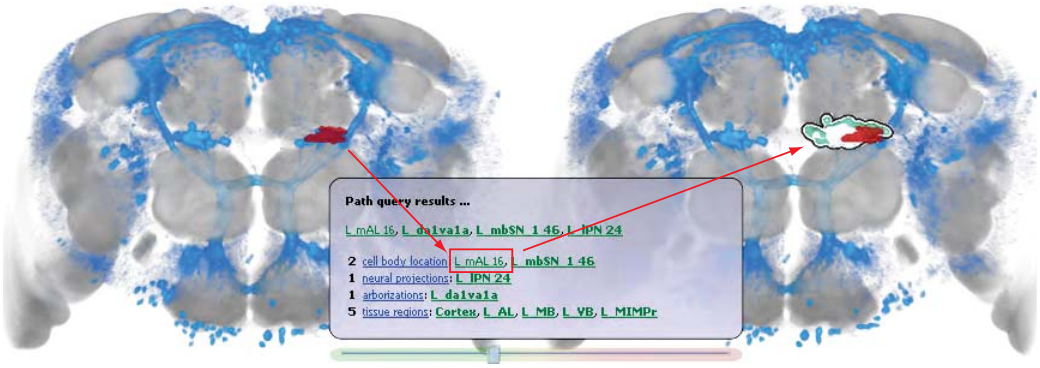
WE DESCRIBED a system for visual exploration of microscopy neurobiological data which supports multi-channel volumetric data in combination with segmented structures represented as geometry. In this chapter, we present several typical use-case scenarios of exploration using visual queries. To evaluate our system, we measured its performance and conducted an informal discussion session with domain experts. In Section 5.2, we present the performance results, and in Section 5.3, we summarize the discussion.

5.1 Use-case Scenarios

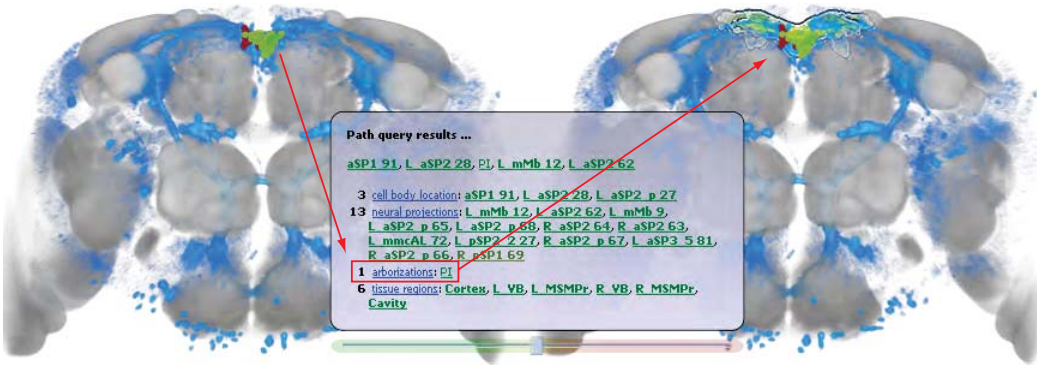
To demonstrate the results achievable with our system, we present four use-case scenarios. We display the results obtained with the path, object and semantic queries separately and in combination.



(a)



(b)



(c)

Figure 5.1: Path-query scenarios: Template volume is shown in conjunction with a blue average Gal4-volume. Search of structures crossing particular regions of the average Gal4-volume: neuronal projections (a), the closest cell body (b) and the closest arborization (c).

Path-query Scenarios

To illustrate a typical scenario of using a path-query, we provide several screen captures of the workflow (see Figure 5.1). First, the user explores an average Gal4-volume in conjunction with a template volume which provides anatomical context. The average Gal4-volume displayed in blue depicts a particular neuronal tissue of interest which includes, e.g., neurons and synapses. An expert user is able to recognize structures within the average Gal4-volume. Consequently, he or she desires to further inspect these regions to exactly identify the structures. For example, they sketch a path over the region which they identified as a bundle of neuronal projections as in Figure 5.1a. The hypertext label shows the neuronal projections which cross this region, and the user loads them all by clicking on the *neuronal projections* label. In the same fashion, the user marks a region and loads the closest cell body as in Figure 5.1b, and the closest arborization as in Figure 5.1c. In Figure 5.2, the search region is enlarged. Consequently, a proximity cloud indicating the extent of the search region appears. Figure 5.2 depicts two cell bodies from the list of the cell bodies displayed in the label - the closest and the furthest.

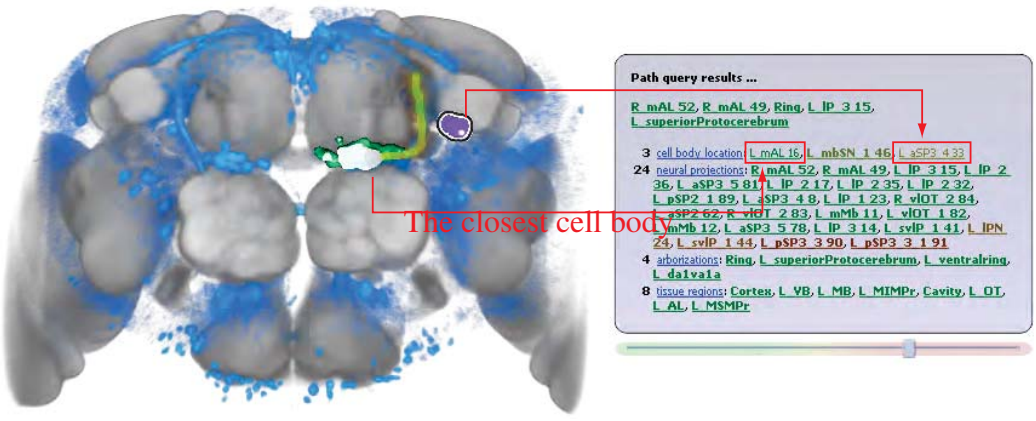
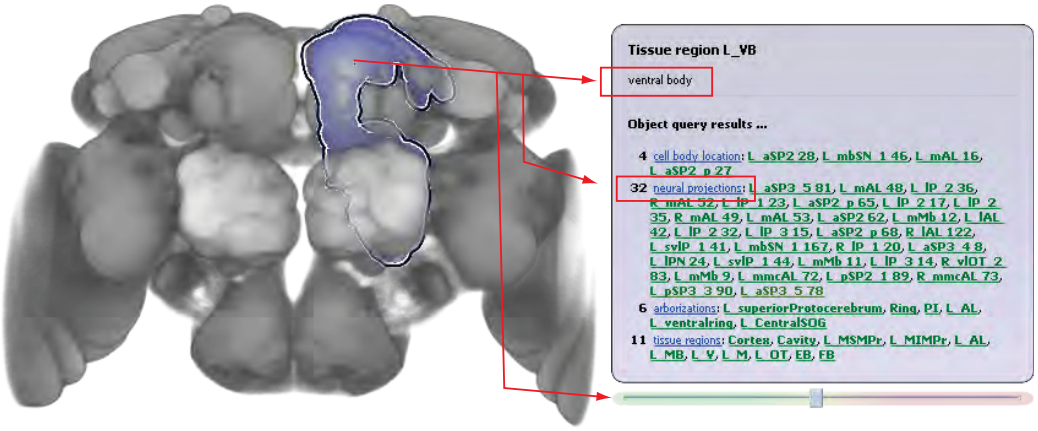


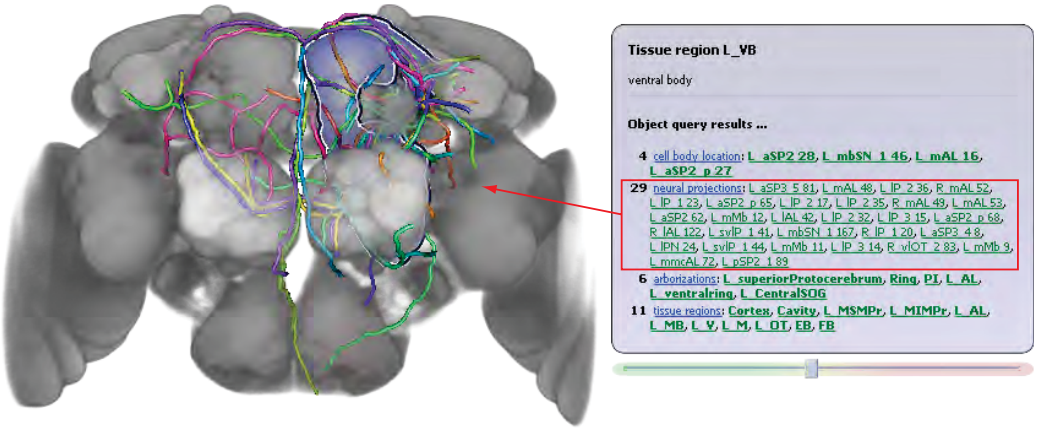
Figure 5.2: Structures with different distances from the sketched path.

Object-query Scenarios

Object queries provide information about distances and intersections between objects which enables fast and simple browsing. For example, the user is able to quickly determine which neurons cross a particular part of the brain as illustrated in Figure 5.3. Regions of the brain which have been segmented from the template image of the brain can be loaded via the table-view interface to the database and selected with a mouse click on their visualization. The hypertext label displays the object query results. The user adjusts the slider so that only structures which

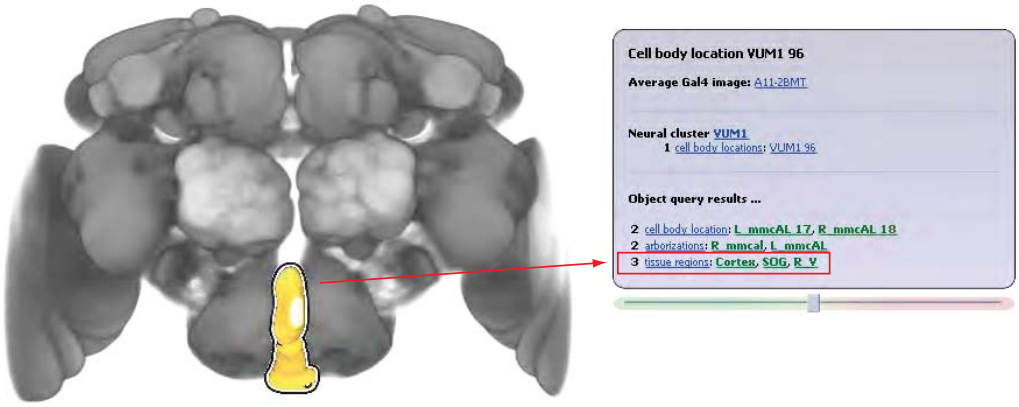


(a)

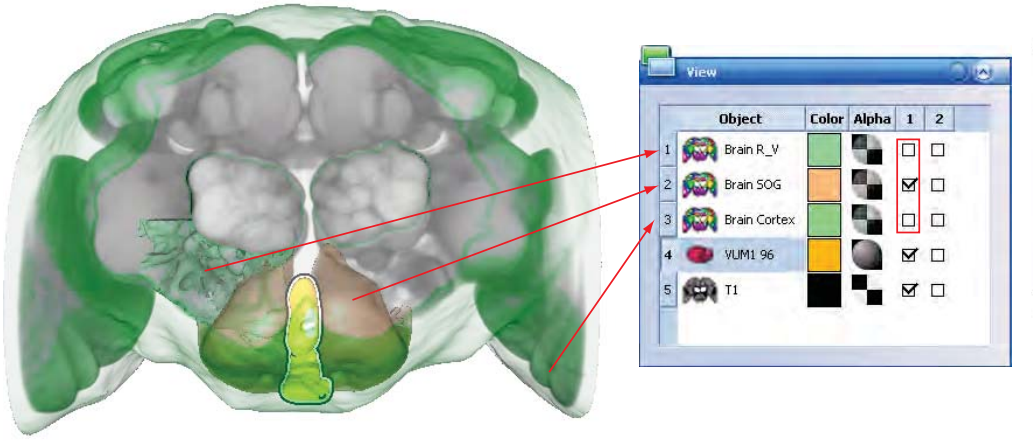


(b)

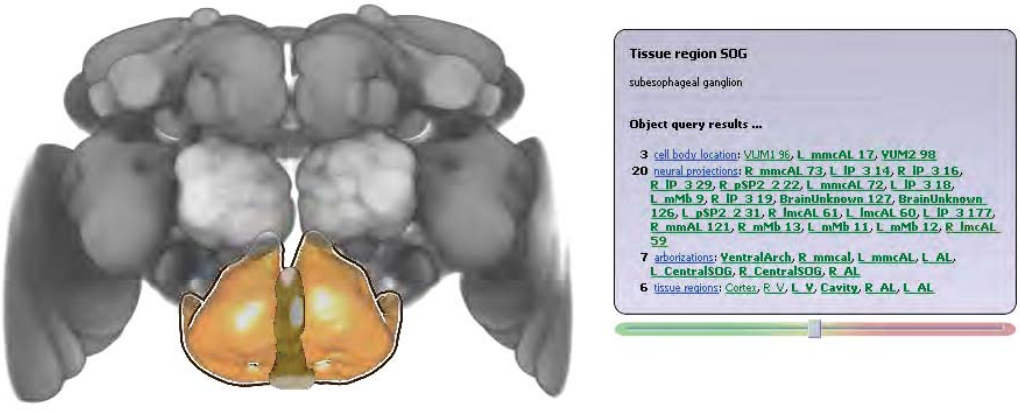
Figure 5.3: An object-query scenario I.: Selection of a tissue region of the brain (a). Selection of neuronal projections crossing a selected tissue region (b).



(a)



(b)



(c)

Figure 5.4: An object-query scenario II.: Searching for tissue regions which include a particular structure (a). Removing of tissue regions in the view-panel (b). Selecting one of the proposed tissue regions for further inspection (c).

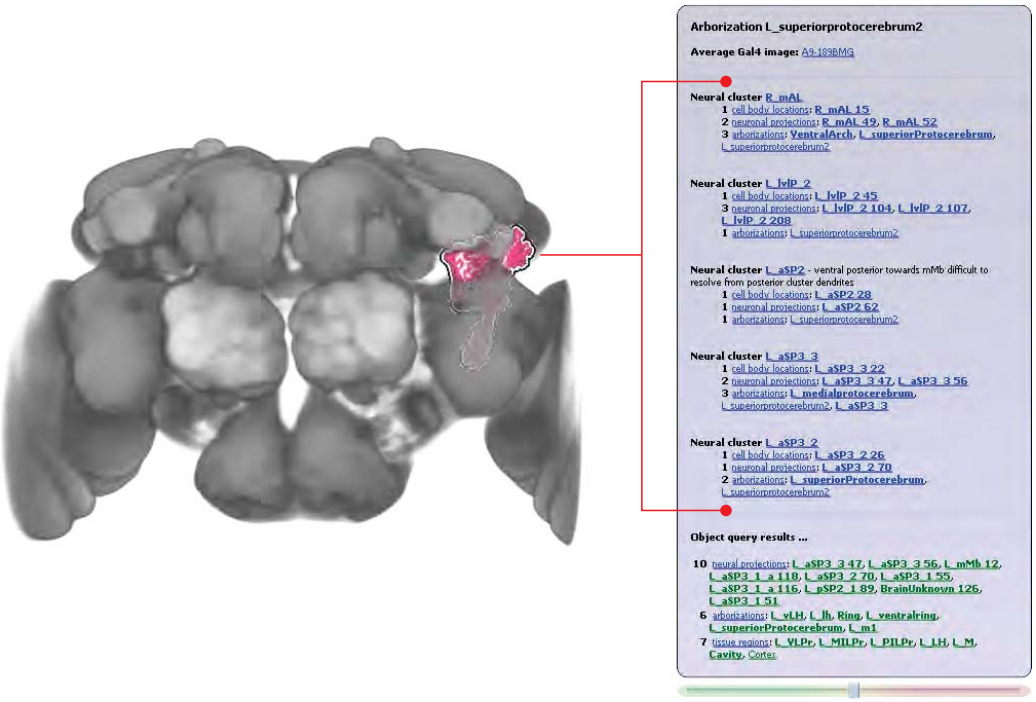
have intersections are shown. Thus, only green filtered structures appear in the filtered list. To load all filtered structures, he or she clicks on the hyperlink *neuronal projections* as shown in Figure 5.3a. Consequently, all neuronal projections crossing the selected brain region appear in the visualization as illustrated in Figure 5.3b. Object queries also enable the user to quickly find a region of the brain to which a particular structure belongs to as illustrated in Figure 5.4. He or she clicks on the depicted structure in the visualization to prompt the object-query results label as in Figure 5.4a. The slider is adjusted to show only structures which intersect the selected structure. Subsequently, the user loads all tissue regions with intersections. In the view panel, he or she unchecks regions identified as not appropriate as in Figure 5.4b. In Figure 5.4c, only one tissue region remains in the visualization for further inspection.

Semantic-query Scenario

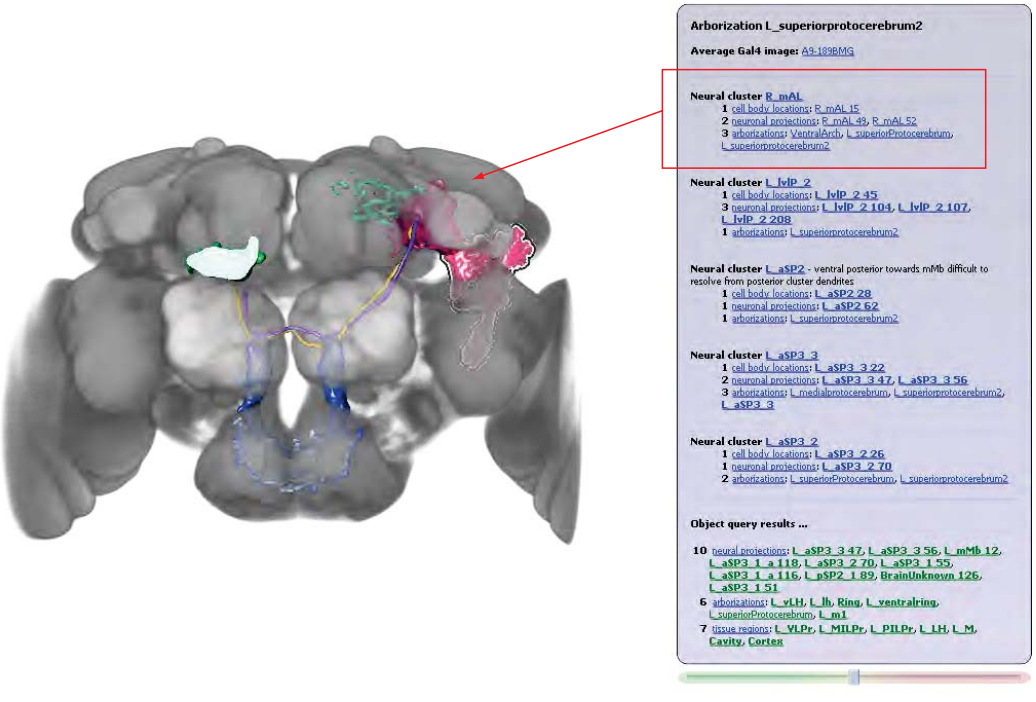
A typical case of applying a semantic-query is finding neural clusters which are related to a particular structure. In particular, arborization are members of several neural clusters. Semantic-query enables the user to view and compare the entire clusters. In Figure 5.5, we illustrate this process step-by-step. First, the user selects a particular arborization. The semantic-query displays all cluster to which the selected arborization belongs to as illustrated in Figure 5.5a. Second, the user selects one of the proposed clusters, which is consequently loaded into the visualization as in Figure 5.5b.

Combined Query Scenarios

In practice, all three types of queries we presented in this thesis are used in conjunction with each other. Figures 5.6, 5.7, and 5.8 give examples of combining different types of queries for browsing in the database. In Figure 5.6a, the user desires to identify a particular region of the brain. He or she uses the path-query to find the closest tissue region which includes the sketched path. Consequently, they load the tissue region displayed in the query results at the most left. By clicking on the loaded region, they display all information which is stored in the database

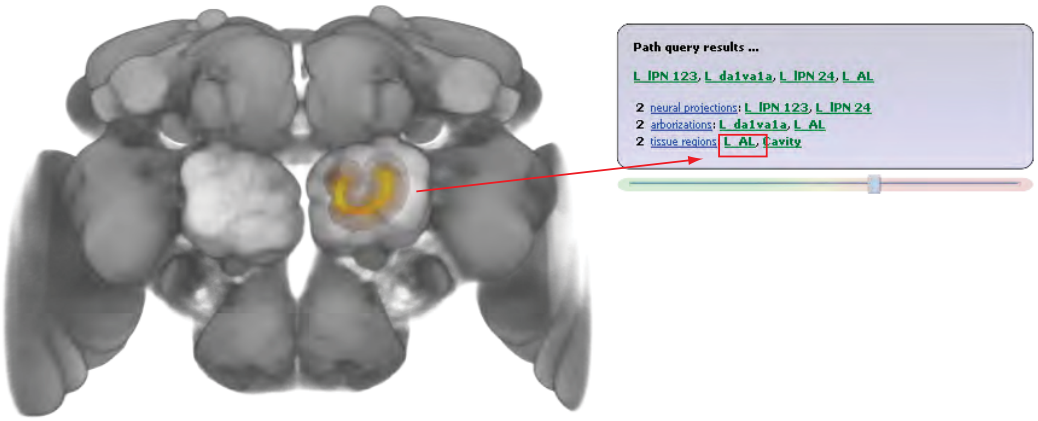


(a)

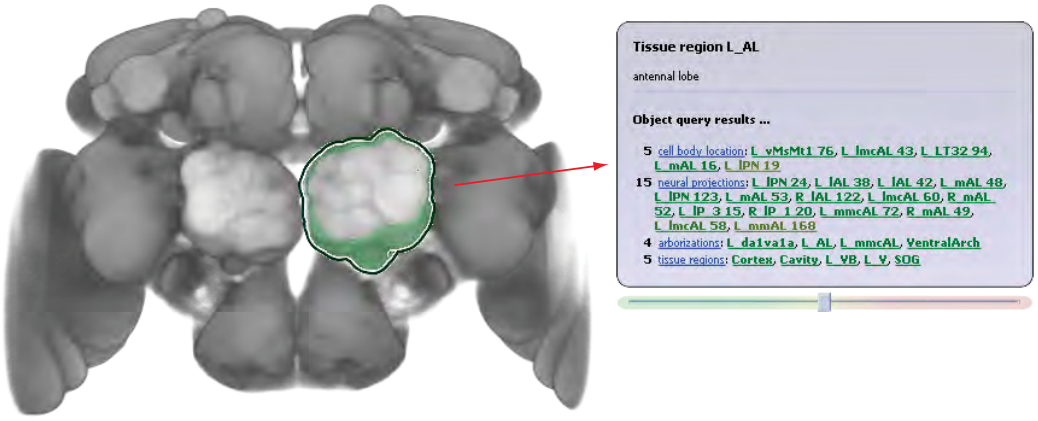


(b)

Figure 5.5: An semantic-query scenario: Selection of an arborization (a) and subsequent loading of the associated neural cluster (b).



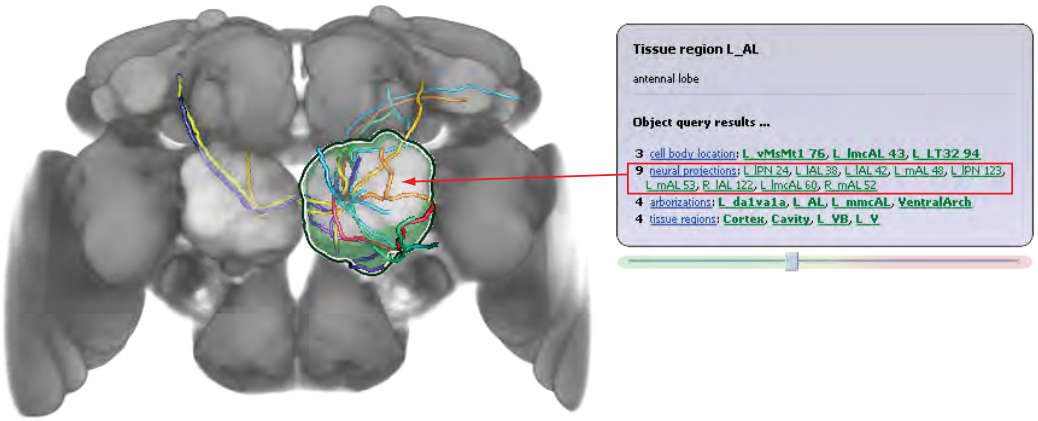
(a)



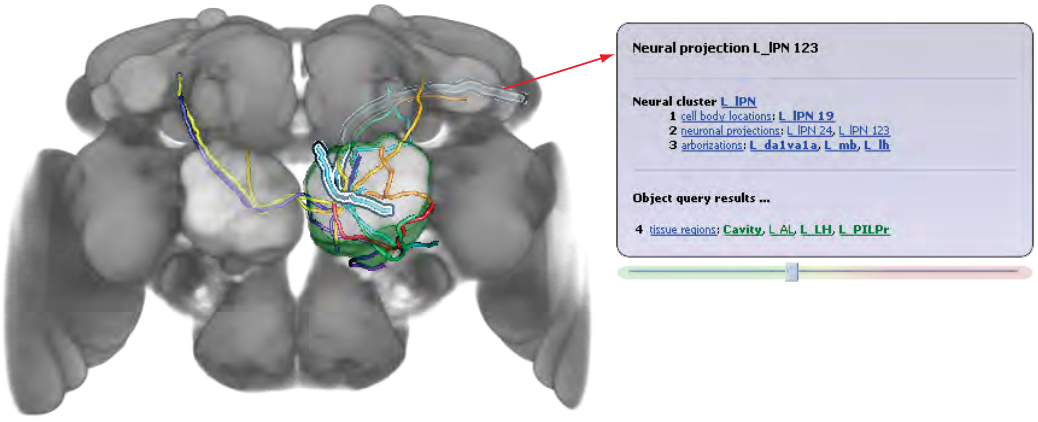
(b)

Figure 5.6: A combined query scenario I.: Identifying a region using a path-query (a). Selection of the closest region for further inspection and displaying the annotation using a semantic-query (b).

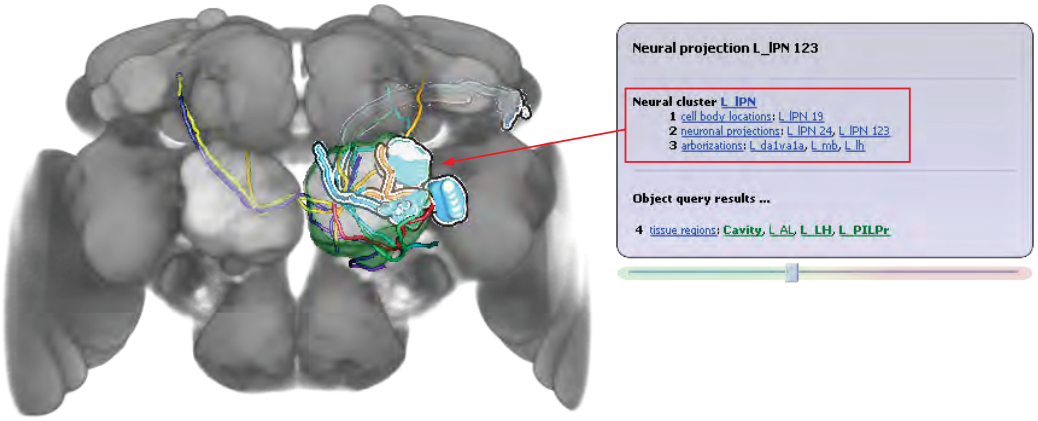
about this region and the results of the object-query (see also Figure 5.6b). They continue browsing by loading all neuronal projections crossing the selected tissue region as indicated in Figure 5.7a. In Figure 5.7b, the highlighted neuronal projection is selected for further inspection. The user applies the semantic-query to load the associated neuronal cluster as shown in Figure 5.7c. Further options while browsing through the structures are as follows. The user observed a bundle in the neuronal projections crossing the depicted tissue region (see also Figure 5.8a). Consequently, he or she desires to search for other structures, e.g., synapses, con-



(a)

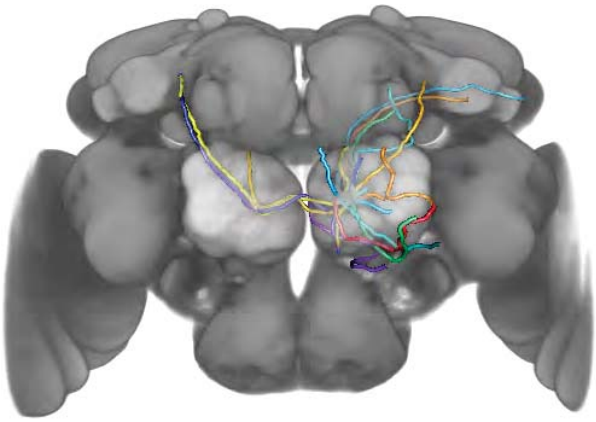


(b)

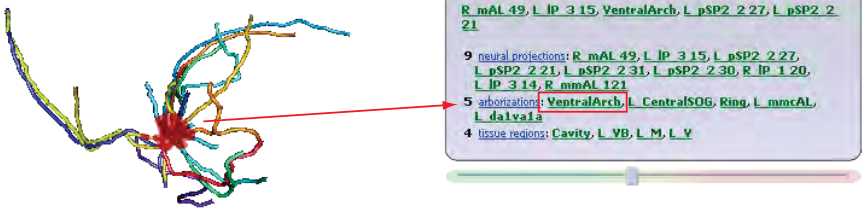


(c)

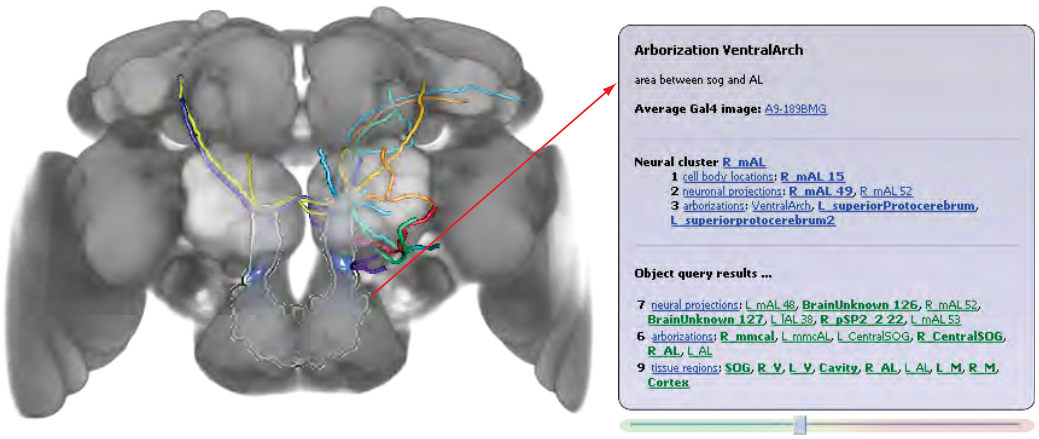
Figure 5.7: A combined query scenario II.: Loading all neuronal projection which cross the selected region (a). Selecting one of the loaded neuronal projections (b). Loading the associated neural cluster (c).



(a)



(b)



(c)

Figure 5.8: A combined query scenario III.: A bundle in the collection of neuronal projections (a). Inspecting the bundle using a path-query (b). Loading the closest arborization to the bundle (c).

nected to the neuronal bundle. He or she marks the bundle using the path-query which delivers the proximal structures as in Figure 5.8b and load the most closest arborization as in Figure 5.8c. They continue browsing in a similar fashion or save the current session and continue their work later.

5.2 Performance

We evaluated the performance of our system as follows. First, we measured the latencies in millisecond (*ms*) while reading rows of the distance table from the disk. To provide comparative values, we focused on the following criteria. First, we compared the latencies while reading rows of the distance table which we employ in practice (185MB) and while reading rows of a large distance table (2GB). For this purpose, we generated an additional distance table with distance restriction of 256 voxels instead of 40 voxels. Second, we compared the latencies while reading rows of the distance tables which use the 3D Hilbert indirections as in Figure 3.14 and the linear indirections as in Figure 3.13. Third, we inspected the latencies for sketching done on the 3D visualization and on 2D slices. While the user sketches a path on a 2D slice, the path becomes a plane curve. However, while sketching

Table 5.1: Latencies measured on the 185MB-large distance table

	3D Hilbert indirections	Linear indirections
Slice XY	0.86ms	0.6ms
Slice YZ	1.09ms	1.6ms
Slice ZX	0.49ms	3.6ms
DVR	0.38ms	1.92ms

Table 5.2: Latencies measured on the 2GB-large distance table

	3D Hilbert indirections	Linear indirections
Slice XY	2.46ms	1.32ms
Slice YZ	3.1ms	4.95ms
Slice ZX	1.3ms	2.97ms
DVR	3.04ms	4.15ms

on the 3D visualization, e.g., DVR, the path defines a space curve. The results of the latency measurements are summarized in Tables 5.1 and 5.2. Each comparative value presented in the tables is the average value over five independent measurements. During sketching, the system processes path queries and renders with frame rates up to 30Hz. All measurements were performed on a workstation equipped with an Intel® 6600 Core Duo™ CPU 2.4GHz, 2.0GB RAM, a hard disk VelociRaptor 10000RPM with 16MB cache and 4.2ms seek time and a NVIDIA® 8800 GTX GPU.

5.3 Discussion

To evaluate the usability of our system, we conducted an informal discussion session with our collaborators. We received a positive and encouraging feedback and suggestions. Based on their comments, we attached the color-coded distance slider to the hypertext label to enable interactive filtering of the results. Also, we displayed the hyperlinks to individual structures in the list of results using the color scheme of the distance slider to allow an immediate association of the displayed objects with their degree of proximity.

The neurobiologists affirm that availability of such a system would ease their research and consider this direction of research to be promising. However, it is difficult to assess all benefits and to foresee the impact of the system on the neurobiology research at this stage, as its potential has not been exploited to its full extent.

CHAPTER

6 Conclusion

When I examine myself and my methods of thought, I come to the conclusion that the gift of fantasy has meant more to me than any talent for abstract, positive thinking.

Albert Einstein

THIS CHAPTER summarizes the main contributions and concludes this thesis. Furthermore, it draws future perspectives of the research direction described in this thesis.

6.1 Summary

In this thesis, we introduced a novel approach of efficient browsing in the database of neuronal data using visual queries as a part of a visualization system. We described three basic types of visual queries - path queries, object queries, and semantic queries. Path queries display objects in a proximity of a sketched path in an in-window label. The extent of the proximity region is defined interactively with a slider. The user receives an immediate feedback concerning the spatial distribution of objects around the path. In practice, each path decays into

a discrete series of points which are reprojected into the volume space. For every point of the series, the query delivers a sorted list of objects which cross the neighborhood within the given extent. To enable interactivity, the query retrieves the lists of objects from a preprocessed distance table. The table contains a list of $< distance, objectid >$-pairs for every point of the volume within a neighborhood with maximal extent of 40 voxels. We calculated the distances using precomputed distance field volumes of each object. The sizes of the distance table depends on the number of objects stored in the database. As we expect a significant growth of the database, we designed the distance table for off-core storage. In our implementation, we employed a storage pattern using a 3D Hilbert scan which has good locality-preserving properties which decreases disk latency. Regarding the scalability of the database, we proposed an effective merging algorithm which allows us to merge two distance tables computed for two disjoint collections of objects.

Object queries deliver a sorted list of $< distance, objectid >$-pairs for objects in the vicinity. They are sorted in an ascending order with respect to the distance. If two objects have an intersection, we provided the intersection volume instead of the distance. To facilitate interactivity, we precomputed the table containing the sorted lists of $< distance, objectid >$-pairs for every object stored in the database. The results are displayed in a label which appears in-window and prompts further inspection.

Semantic queries retrieve the contextual information about an object selected for inspection. The selection is done by a simple mouse click on the object in the visualization. The results are displayed in a label which summarizes annotatations made by neurobiologists and lists all neural clusters to which the selected object belongs to.

We described a proof-of-concept system which features state-of-the-art visualization techniques in 3D volume and geometry rendering and facilitates visual queries. The system presents a compact tool for visualization and effective exploration of neurobiology data. The design process benefitted from a constant input of neurobiologists. To evaluate the usability of the system, we performed an informal discussion session with our collaborators. We received encouraging feedback and suggestions for future work. However, it is difficult to foresee the

thorough impact of the system on the neurobiology research at this stage, as the potential of this direction in research has not been exploited to its full extent.

Future Work

In this thesis, we presented a system which couples state-of-the-art visualization techniques and effective visual browsing in a database of neuronal data. However, there are still possible extensions and enhancements worth to be explored. The goal of our collaborators is to build an online of anatomy of frutfly's neuronal system and make if available for their research community. The future system aims to facilitate collaborative research and intercommunal exchange of knowledge in neurobiology. In this scope, it will support interactive insertion and modification of the annotations and semantic relationships.

Furthermore, our collaborators proposed to include the average Gal4-volumes into the visualization-aided browsing. The average Gal4-volumes depict high-intensity stained structures immersed in a relatively homogenous tissue. This allows us to represent the high-intensity structures with a not necessarily connected skeleton graph and use it for generation of a ditance field volume. An alternative is to find a threshold value which separates the high-intensity structures from the homogenous tissue. The threshold value will be used for isosurface generation which will allow us to generate the respective distance field volumes.

We incorporated visual feedback while executing path queries in a form of a proximity cloud. The cloud allows the user to estimate the spatial distribution of all objects in proximity, or alternatively the spatial distribution of objects which belong to one subgroup. It does not enable the user to estimate the spatial position with relation to the sketched path for a single object. The further work might support visual feedback about the position of individual objects. This would allow the user to recognize if a particular object has multiple intersections with the sketched path, and if it its shape is prolonged in the sense of the path or not.

Acknowledgements

I go, and it is done; the bell invites me.

William Shakespaere
Macbeth (II,i,62)

At this point, I would like to send my thanks to everyone who accompanied me during my studies and helped me to carry out this thesis.

First of all to the *Vis-group* at the *Institute of Computer Graphics and Algorithms* for a nice environment, and especially to *Stefan Bruckner* and *Meister Eduard Gröller* for supervision and proofreading of this thesis. Second, to the partners at the *Institute of Molecular Pathology* for their constructive input, providing the data sets and their feedback, and especially to *Jai Yi Yu* for explanatory documents concerning their data sets and workflow.

I also want to thank to my sister *Barbora* who helped me to understand the biological background and who lended me her books on genetics.

A great thanks you goes also to my parents who supported and financed me during my studies.

I thank also to *Herbert Grasberger* and *Michael Schwärzler* for proofreading of the thesis and corrections of its german part.

Last but not least, I would like to thank to all my friends and also to the *CG-Club* who helped to create moments which made my studies funny and unforgettable, to all my past teachers and lecturers who brought me where I am now, and to those people who influnced my future :-).

Bibliography

[1] Amira. `www.amira.com` (Acessed June, 2009).

[2] Flybrain. `http://flybrain.neurobio.arizona.edu` (Accessed May, 2009).

[3] Christopher Ahlberg, Christopher Williamson, and Ben Shneiderman. Dynamic queries for information exploration: An implementation and evaluation. In *Proceedings of ACM CHI*, pages 619–626, 1992.

[4] Michael Ashburner, Kent K. Golic, and R. Scott Hawley. *Drosophila: A Laboratory Handbook*. Cold Spring Harbor Laboratory Press, 2nd edition, 2005.

[5] Ricardo S. Avila, Lisa M. Sobierajski, and Arie E. Kaufman. Visualizing nerve cells. *IEEE Computer Graphics and Applications*, 14(5):11–13, 1994.

[6] Barry G. Becker. Volume rendering for relational data. In *Proceedings of IEEE Symposium on Information Visualization*, pages 87–90, 1997.

[7] Louise Bertrand and Jonathan Nissanov. The neuroterrain 3D mouse brain atlas. *Frontiers in Neuroinformatics*, 2, 2008.

[8] Johanna Beyer, Markus Hadwiger, Stefan Wolfsberger, and Katja Bühler. High-quality multimodal volume rendering for preoperative planning of neurosurgical interventions. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1696–1703, 2007.

[9] Gleb Bezgin, Andrew T. Reid, Dirk Schubert, and Rolf Kötter. Matching spatial with ontological brain regions using java tools for visualization, database access, and integrated data analysis. *Neuroinformatics*, 7(1):7–22, 2009.

[10] Jan G. Bjaalie. Localization in the brain: New solutions emerging. *Nature Reviews Neuroscience*, 3:322–325, 2002.

[11] Stefan Bruckner and Meister Eduard Gröller. Volumeshop: An interactive system for direct volume illustration. In *Proceedings of IEEE Visualization*, pages 671–678, 2005.

[12] Stefan Bruckner and Meister Eduard Gröller. Instant volume visualization using maximum intensity difference accumulation. *Computer Graphics Forum (to appear)*, 28(3), 2009.

[13] Stefan Bruckner, Dieter Schmalstieg, Helwig Hauser, and Meister Eduard Gröller. The inverse warp: Non-invasive integration of shear-warp volume rendering into polygon rendering pipelines. In *Proceedings of Vision, Modeling, and Visualization*, pages 529–536, 2003.

[14] Gully A.P.C. Burns, Wei-Cheng Cheng, Richard H. Thompson, and Larry W. Swanson. The neuart II system: A viewing tool for neuroanatomical data based on published neuroanatomical atlases. *BMC Bioinformatics*, 7(531):431–459, 2006.

[15] Marina Chicurel. Databasing the brain. *Nature*, 406:822–825, 2000.

[16] Wim C. de Leeuw, Robert van Liere, Pernette J. Verschure, Roel van Driel, Astrid E. Visser, and Erik M. M. Manders. Visualization of time dependent confocal microscopy data. In *Proceedings of IEEE Visualization*, pages 473–476, 2000.

[17] Wim C. de Leeuw, Pernette Verschure, and Robert van Liere. Visualization and analysis of large data collections: A case study applied to confocal microscopy data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1251–1258, 2006.

[18] Winfried Denk and Heinz Horstmann. Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure. *PLoS Biology*, 2(11):1900–1909, 2004.

[19] Mark Derthick, John Kolojejchick, and Steven F. Roth. An interactive visual query environment for exploring data. In *Proceedings of ACM UIST*, pages 189–198, 1997.

[20] Barry Dickson. Research institute of molecular pathology. `www.imp.ac.at/research/barry-dickson/` (Accessed May, 2009).

[21] Rachel Drysdale and the FlyBase Consortium. Flybase - a database for the drosophila research community. In *Drosophila. Methods and Protocols*, volume 420 of *Methods in Molecular Biology*, pages 45–59. Humana Press, 2008. `http://flybase.org`.

[22] California Faculty of Chemistry, University of Santa Cruz. `www.chemistry.ucsc.edu/~lokey/108A/images/neuron.png` (Accessed May, 2009).

[23] Jesper Fredriksson. Design of an internet accessible visual human brain database system. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, pages 469–474, 1999.

[24] Volker Gaede and Oliver Günther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, 1998.

[25] William J. Gilbert. A cube-filling hilbert curve. *The Mathematical Intelligencer*, 6(3):78, 1984.

[26] Mark A. Harrower and Cynthia A. Brewer. ColorBrewer.org: An online tool for selecting color schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003.

[27] Donald Hearn and M. Pauline Baker. *Computer Graphics with OpenGL*. Prentice Hall, 3rd edition, 2004.

[28] David Hilbert. Über die stetige Abbildung einer Linie auf ein Flächenstück. *Mathemathische Annalen*, 11(38):459–460, 1891.

[29] MedicineNet Inc. Medical dictionary. `www.medterms.com` (Acessed April, 2009).

[30] Alfred Inselberg and Bernard Dimsdale. Parallel coordinates: A tool for visualizing multi-dimensional geometry. In *Proceedings of IEEE Visualization*, pages 361–378, 1990.

[31] Lundbeck Institute. Brain explorer. `www.brainexplorer.org` (Accessed May, 2009).

[32] Hosagrahar V. Jagadish. Linear clustering of objects with multiple attributes. *ACM SIGMOD*, 19(2):332–342, 1990.

[33] Sei-ichiro Kamata and Yukihiro Bandoh. An address generator of a pseudo-hilbert scan in a rectangle region. In *Proceedings of International Conference on Image Processing*, page 707, 1997.

[34] Sei-ichiro Kamata, Richard O. Eason, and Yukihiro Bandou. A new algorithm for n-dimensional hilbert scanning. *IEEE Transactions on Image Processing*, 8(7):964–973, 1997.

[35] Arie E. Kaufman, Roni Yagel, Reuven Bakalash, and Ilan Spector. Volume visualization in cell biology. In *Proceedings of IEEE Visualization*, pages 160–167, 1990.

[36] Daniel A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 7(1):100–107, 2002.

[37] Joe Kniss, Simon Premoze, Milan Ikits, Aaron Lefohn, Charles Hansen, and Emil Praun. Gaussian transfer functions for multi-field volume visualization. In *Proceedings of IEEE Visualization*, pages 497–504, 2003.

[38] Steven H. Koslow and Shankar Subramaniam, editors. *Databasing the Brain: From Data to Knowledge (Neuroinformatics)*. Wiley, 2002.

[39] Marc Levoy. A hybrid ray tracer for rendering polygon and volume data. *IEEE Computer Graphics and Applications*, 10(2):33–40, 1990.

[40] Benjamin Lewin. *Genes IX*. Jones and Bartlett, 9th edition, 2007.

[41] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Computer Graphics*, 21(4):163–169, 1987.

[42] Thomas Luft, Carsten Colditz, and Oliver Deussen. Image enhancement by unsharp masking the depth buffer. *ACM Transactions on Graphics*, 25(3):1206–1213, 2006.

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

[44] Allen R. Martin and Matthew O. Ward. High dimensional brushing for interactive exploration of multivariate data. In *Proceedings of IEEE Visualization*, pages 271–278, 1995.

[45] Alexander Maye, Thomas Hermann Wenckebach, and Hans-Christian Hege. Visualization, reconstruction, and integration of neuronal structures in digital brain atlases. *International Journal of Neuroscience*, 116(4):431–459, 2006.

[46] Bruce H. McCormick. Knife-edge scanning microscope. `http://research.cs.tamu.edu/bnl/kesm.html` (Accessed May, 2009).

[47] Zeki Melek, David Mayerich, Cem Yuksel, and John Keyser. Visualization of fibrous and thread-like data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1165–1172, 2006.

[48] Richard J. Millar, M. Elizabeth C. Hull, and John H. Fraser. The millar polyhedron and its use in the construction of octrees. *The Compuer Journal, Oxford Journals*, 36(2):186–194, 1993.

[49] Guy M. Morton. A computer oriented geodetic data base and a new technique in file sequencing. *Technical Report: IBM Ltd.,Ottawa, Canada*, 1966.

[50] Keith O'Conor, Paul Voorheis, and Carol O'Sullivan. 3d visualisation of confocal flourescence microscopy data. In *Proceedings of EUROGRAPHICS*, pages 49–54, 2004.

[51] Chris B. Phelps and Andrea H. Brand. Ectopic gene expression in drosophila using Gal4 system. *Methods*, 14(4):367–379, 1998.

[52] Vikram Prasad, Denis Semwogerere, and Eric R. Weeks. Confocal microscopy of colloids. *Journal of Physics: Condensed Matter*, 19(11):113102 (25pp), 2007.

[53] William A. Press, Bruno A. Olshausen, and David C. Van Essen. A graphical anatomical database of neural connectivity. *Philosophical Transactions of the Royal Society*, 356:1147–1157, 2001.

[54] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, 1st edition, 1990.

[55] Joel Quinqueton and Marc Berthod. A locally adaptive peano scanning algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-3(4):409–412, 1981.

[56] Guido Reina and Thomas Ertl. Volume visualization and visual queries for large high-dimensional datasets. In *Proceedings of Joint Eurographics – IEEE TCVG Symposium on Visualization*, pages 255–260, 2004.

[57] Torsten Rohlfing and Carsten 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.

[58] Peter J. Russell. *Genetics*. Benjamin Cummings, 5th edition, 1998.

[59] Hans Sagan. A three-dimensional hilbert curve. *International Journal of Mathematical Education in Science and Technology*, 24(4):541–545, 1993.

[60] Georgios Sakas, Michael G. Vicker, and Peter J. Plath. Visualization of laser confocal microscopy datasets. In *Proceedings of IEEE Visualization*, pages 375–379, 1996.

[61] Hanan Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufman, an Imprint of Elsevier, 1st edition, 2006.

[62] Stephan Schmitt, Jan Felix Evers, Carsten Duch, Michael Scholz, and Klaus Obermayer. New methods for the computer-assisted 3D reconstruction of neurons from confocal image stacks. *NeuroImage*, 23(4):1283–1298, 2004.

[63] Denis Semwogerere and Eric R. Weeks. Confocal microscopy. In *Encyclopedia of Biomaterials and Biomedical Engineering*, Biomaterials, Biomedical Engineering. Informa Healthcare, 2005.

[64] Anthony Sherbondy, David Akers, Rachel Mackenzie, Robert Dougherty, and Brian Wandell. Exploring connectivity of the brain's white matter with dynamic queries. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):419–430, 2005.

[65] Qiqi Wang, Yinlong Sun, and J. Paul Robinson. Gpu-based visualization techniques for 3D microscopic imaging data. In *Proceedings SPIE*, page 64981H, 2007.

[66] Eric W. Weisstein. Zeckendorf representation. From MathWorld – A Wolfram Web Resource. `http://mathworld.wolfram.com/GaussianFunction.html` (Accessed May, 2009).

[67] Carl Zeiss MicroImaging Inc. `www.zeiss.com/micro` (Accessed April, 2009).

[68] Jian Zhang and Sei-ichiro Kamata. A pseudo-hilbert scan for arbitrarily-sized cuboid region. *International Symposium on Signal Processing and Information Technology*, pages 920–925, 2006.

List of Figures

List of Tables